EUROPEAN COMMISSION

# *CASE DMA.100220 - Alphabet - OS - Google Android - Art. 6(7) – SP – AI*

(Only the English text is authentic)

## Digital Markets Act

## Regulation (EU) 2022/1925 of the European Parliament and of the Council

Article 8(2) Regulation (EU) 2022/1925

Date: 16/07/2026

This text is made available for information purposes only. A summary of this decision is published in all EU languages in the Official Journal of the European Union.

Parts of this text have been edited to ensure that confidential information is not disclosed. Those parts are replaced by a non-confidential summary in square brackets or are shown as […].

EUROPEAN
COMMISSION

Brussels, 16.7.2026
C(2026) 5094 final

# COMMISSION IMPLEMENTING DECISION

**of 16.7.2026**

**pursuant to Article 8(2) of Regulation (EU) 2022/1925 of the European Parliament and of the Council on contestable and fair markets in the digital sector**

**(Case DMA.100220 – Alphabet - OS - Google Android - Art. 6(7) – SP – AI)**

(Only the English text is authentic)

# TABLE OF CONTENTS


LIST OF ABBREVIATIONS ..... 10

1. Introduction ..... 12

2. Procedure ..... 13

2.1. Chronology of the current proceedings ..... 13

2.2. Commission's engagement with Alphabet ..... 17

2.3. Commission's engagement with third parties ..... 19

2.4. Alphabet's participation in the public consultation ..... 20

3. Rationale for specification proceedings ..... 22

3.1. Importance of AI-powered services ..... 22

3.2. Alphabet's unique position in the AI value chain ..... 23

3.3. Importance of effective interoperability with Google Android for AI-based services ..... 25

3.4. Current degree of interoperability for AI-based services on Google Android ..... 27

3.5. Use of specification proceedings ..... 28

4. Legal framework pursuant to Regulation (EU) 2022/1925 ..... 30

4.1. Specification proceedings for Article 6(7) DMA ..... 30

4.2. Features and functionalities ..... 32

4.3. Interoperability with the same features as available to the gatekeeper and under equal conditions ..... 34

4.4. Features available to or used by the gatekeeper ..... 35

4.5. Interoperability must be effective ..... 37

4.5.1. Implementation of interoperability ..... 37

4.5.2. User journey and friction ..... 39

4.5.3. Use of interoperability solutions in the gatekeeper's first-party apps ..... 41

4.6. The interoperability obligation applies to designated gatekeepers, irrespective of the implementation of the operating system on their own or on third-party devices ..... 43

4.7. Eligibility of beneficiaries, apps and use cases ..... 47

4.8. Integrity justification ..... 48

4.8.1. Scope of integrity measures ..... 49

4.8.2. Implementation in compliance with applicable law ..... 52

4.8.3. Role of user authorisation ..... 56

4.8.4. Justification of integrity measures ..... 58

5. Background on Google Android and Gemini ..... 62

5.1. Overview of the Google Android ecosystem ..... 62

5.1.1. The Android Open Source Project (AOSP) ..... 62

5.1.2. Google Android middleware ..... 64
5.1.3. Relevant aspects of Alphabet's contractual relationship with third-party OEMs ..... 72
5.2. Access controls in Android ..... 74
5.2.1. The operating system decides which apps have access to features and Alphabet decides how feature access is implemented in the operating system ..... 75
5.2.2. General access control model on Google Android ..... 76
5.2.3. Conditioning access to features to apps holding certain permissions ..... 77
5.2.4. Conditioning access to features on roles, default applications, and services ..... 79
5.2.5. Conditioning access to features on app preinstallation by the OEM ..... 82
5.3. Implementation of Gemini and Google Assistant in Google Android ..... 84
5.4. OEMs have granted access to Gemini and its own AI assistants, but only rarely to third parties ..... 86
5.4.1. OEMs rarely grant access to third-party AI service providers ..... 88
5.4.2. Gemini is pre-installed by OEMs with relevant feature access ..... 91
6. Alphabet's obligations under Article 6(7) DMA ..... 91
6.1. Alphabet must take all necessary measures to ensure effective interoperability with all Google Android powered smart mobile devices ..... 91
6.1.1. Taking all necessary measures in practice ..... 91
6.1.2. No disproportionate impact on third-party OEMs ..... 93
6.2. The scope of the designated Google Android operating system ..... 95
6.2.1. Interpretation of Article 2(10) ..... 96
6.2.2. Designated middleware ..... 99
6.2.2.1. AICore and the Gemini Nano ODMs are part of the designated Android middleware ..... 100
6.2.2.2. The Google Hotword Engine is part of the designated middleware ..... 104
6.3. Integrity justification ..... 107
6.3.1. Delegating interoperability decisions to OEMs is not a necessary and proportionate integrity measure within the meaning of Article 6(7) DMA ..... 107
6.3.2. The risk of the features in the present proceedings ..... 113
7. Introduction to the features subject to the proceedings ..... 115
8. Measures on features for invocation ..... 116
8.1. Introduction ..... 116
8.2. Measures for Long-press Home/Long-press Navigation Handle contextual invocation feature ..... 117
8.2.1. Introduction ..... 117
8.2.2. Description of the feature ..... 118
8.2.2.1. Access points ..... 118

8.2.2.2. Contextual data surfaced through the access points are part of the Google Android feature ..... 120
8.2.3. Technical implementation of contextual invocation in Google Android ..... 121
8.2.4. Feature falls within the scope of Article 6(7) DMA ..... 122
8.2.5. Current access for Alphabet's own services and hardware ..... 122
8.2.6. Current access for third-party services and hardware ..... 123
8.2.7. Alphabet's response to the Preliminary Findings ..... 125
8.2.8. The Commission's assessment of Alphabet's claims ..... 125
8.2.8.1. Concerning the existing level of interoperability and integrity ..... 125
8.2.8.2. Concerning proportionality ..... 127
8.2.9. Measures that Alphabet should implement ..... 128
8.2.10. Timing for implementation of the measures ..... 130
8.3. Measures for always-on hotword detection ..... 130
8.3.1. Introduction ..... 130
8.3.2. Description of the feature ..... 132
8.3.3. Technical implementation of always-on hotword detection in Google Android ..... 134
8.3.3.1. Technical implementation of always-on hotword detection in the operating system ..... 134
8.3.3.2. Technical implementation of the always-on hotword detection on the digital signal processor ..... 136
8.3.4. Feature falls within the scope of Article 6(7) DMA ..... 137
8.3.5. Current access for Alphabet's own services and hardware ..... 137
8.3.6. Current access for third-party services and hardware ..... 140
8.3.6.1. Third-party services have very rarely gained the possibility to use hotword detection ..... 140
8.3.6.2. Amazon Alexa always-on hotword detection ..... 141
8.3.6.3. Samsung always-on hotword detection for Bixby and Perplexity ..... 143
8.3.6.4. [Third party's] initiative to implement an open always-on hotword detection API for third-party developers ..... 144
8.3.6.5. Alternative solutions do not provide equally effective interoperability ..... 146
8.3.6.6. Third-party services have no meaningful way for interoperability with always-on hotword detection on Android ..... 146
8.3.7. Commission's assessment of Alphabet's response to the Preliminary Findings ..... 147
8.3.7.1. Concerning the existing level of interoperability and integrity ..... 147
8.3.7.2. Generic hotword proposal ..... 151
8.3.7.3. Interoperable implementations of AOHD ..... 152
8.3.7.4. Concerning proportionality ..... 164
8.3.8. Measures that Alphabet should implement ..... 166

8.3.9. Timing for implementation of the measures ..... 166
9. Measures on features for context ..... 167
9.1. Introduction ..... 167
9.2. Measures for centralised access to apps' data stored on-device ..... 168
9.2.1. Introduction ..... 168
9.2.2. Description of the feature ..... 169
9.2.3. Technical implementation of centralised access to apps' data stored on-device in Google Android ..... 170
9.2.4. Feature falls within the scope of Article 6(7) DMA ..... 172
9.2.5. Current access for Alphabet's own services and hardware ..... 172
9.2.6. Current access for third-party services and hardware ..... 173
9.2.7. Alphabet's response to the Preliminary Findings ..... 174
9.2.8. The Commission's assessment ..... 175
9.2.8.1. Concerning the existence and description of the feature ..... 175
9.2.8.2. Concerning access to Alphabet's apps' data stored on-device ..... 176
9.2.8.3. Concerning the existing level of interoperability and integrity ..... 177
9.2.8.4. Concerning proportionality ..... 179
9.2.9. Measures that Alphabet should implement ..... 181
9.2.10. Timing for implementation of the measures ..... 182
9.3. Measures for context-aware intelligence ..... 182
9.3.1. Introduction ..... 182
9.3.2. Description of the feature ..... 185
9.3.3. Technical implementation of context-aware intelligence in Google Android ..... 188
9.3.3.1. Access by intelligence components ..... 188
9.3.3.2. Access to inputs and outputs ..... 190
9.3.3.3. Access to processing resources ..... 193
9.3.4. Feature falls within the scope of Article 6(7) DMA ..... 194
9.3.5. Current access for Alphabet's own services and hardware ..... 194
9.3.5.1. Access to the context-aware intelligence feature via ASI and PSI ..... 195
9.3.5.2. The use of the context-aware intelligence feature by Alphabet's services ..... 196
9.3.6. Current access for third-party services and hardware ..... 199
9.3.6.1. Access to inputs ..... 199
9.3.6.2. Access to processing resources ..... 200
9.3.6.3. Access to output channels and display surfaces ..... 201
9.3.6.4. Permissions ..... 201
9.3.7. Alphabet's response to the Preliminary Findings ..... 202

9.3.8. The Commission's assessment ..... 202
9.3.8.1. Concerning the existing level of interoperability and integrity ..... 202
9.3.8.2. Concerning proportionality ..... 204
9.3.9. Measures that Alphabet should implement ..... 206
9.3.9.1. The scope of services ..... 206
9.3.9.2. Access to apps ..... 207
9.3.9.3. Equal effectiveness ..... 210
9.3.10. Timing for implementation of the measures ..... 213
9.4. Measures for access to ambient data ..... 213
9.4.1. Introduction ..... 213
9.4.2. Description of the feature ..... 216
9.4.3. Technical implementation of access to ambient data in Google Android ..... 217
9.4.4. Feature falls within the scope of Article 6(7) DMA ..... 218
9.4.5. Current access for Alphabet's own services ..... 219
9.4.6. Current access for third-party services and hardware ..... 220
9.4.7. Alphabet's response to the Preliminary Findings ..... 221
9.4.8. The Commission's assessment ..... 222
9.4.8.1. Concerning the existing level of interoperability and integrity ..... 222
9.4.8.2. Concerning proportionality ..... 224
9.4.9. Measures that Alphabet should implement ..... 225
9.4.10. Timing for implementation of the measures ..... 227
10. Measures on features for actions on apps and the OS ..... 227
10.1. Introduction ..... 227
10.2. Measures for structured on-device integration ..... 228
10.2.1. Introduction ..... 228
10.2.2. Description of the feature ..... 229
10.2.3. Technical implementation of structured on-device integration in Google Android 229
10.2.3.1. App Actions ..... 230
10.2.3.2. App Functions ..... 231
10.2.4. Feature falls within the scope of Article 6(7) DMA ..... 233
10.2.5. Current access for Alphabet's own services and hardware ..... 233
10.2.5.1. App Actions ..... 234
10.2.5.2. App Functions ..... 234
10.2.6. Current access for third-party services and hardware ..... 234
10.2.6.1. App Actions ..... 234
10.2.6.2. App Functions ..... 234

10.2.7. Alphabet's response to the Preliminary Findings ..... 235
10.2.8. The Commission's assessment ..... 236
10.2.8.1. Concerning the existing level of interoperability and integrity ..... 236
10.2.8.2. Concerning proportionality ..... 238
10.2.9. Measures that Alphabet should implement ..... 240
10.2.10. Timing for implementation of the measures ..... 242
10.3. Measures for screen automation ..... 242
10.3.1. Introduction ..... 242
10.3.2. Description of the feature ..... 243
10.3.3. Technical implementation of screen automation in Google Android ..... 245
10.3.4. Feature falls within the scope of Article 6(7) DMA ..... 248
10.3.5. Current access for Alphabet's own services and hardware ..... 248
10.3.6. Current access for third-party services and hardware ..... 249
10.3.7. Alphabet's response to the Preliminary Findings ..... 251
10.3.8. The Commission's assessment ..... 252
10.3.8.1. Concerning scope and integrity ..... 252
10.3.8.2. Concerning proportionality ..... 256
10.3.9. Measures that Alphabet should implement ..... 256
10.3.10. Timing for implementation of the measures ..... 258
10.4. Measures for system integration ..... 258
10.4.1. Introduction ..... 258
10.4.2. Description of the feature ..... 258
10.4.3. Technical implementation of system integration in Google Android ..... 259
10.4.4. Feature falls within the scope of Article 6(7) DMA ..... 260
10.4.5. Current access for Alphabet's own services and hardware ..... 260
10.4.6. Current access for third-party services and hardware ..... 260
10.4.7. Alphabet's response to the Preliminary Findings ..... 261
10.4.8. The Commission's assessment ..... 262
10.4.8.1. Concerning scope, existing level of interoperability and integrity ..... 262
10.4.8.2. Concerning proportionality ..... 264
10.4.9. Measures that Alphabet should implement ..... 265
10.4.10. Timing for implementation of the measures ..... 266
11. Measures on features for access to resources ..... 266
11.1. Introduction ..... 266
11.2. Measures for system-level on-device models ..... 267
11.2.1. Introduction ..... 267

11.2.2. Description of the feature ..... 268
11.2.3. Technical implementation of system-level ODM in Google Android ..... 269
11.2.4. Feature falls within the scope of Article 6(7) DMA ..... 270
11.2.5. Current access for Alphabet's own services and hardware ..... 271
11.2.6. Current access for third-party services and hardware ..... 271
11.2.6.1. Private APIs ..... 271
11.2.6.2. Non-discriminatory conditions, including quotas ..... 273
11.2.6.3. Technological changes ..... 275
11.2.7. Commission's assessment of Alphabet's response to the Preliminary Findings ..... 277
11.2.7.1. Concerning designation, scope, existing level of interoperability and integrity ..... 277
11.2.7.2. Concerning proportionality ..... 278
11.2.8. Measures that Alphabet should implement ..... 281
11.2.9. Timing for implementation of the measures ..... 282
11.3. Measures for on-device model implementation ..... 282
11.3.1. Introduction ..... 282
11.3.2. Description of the feature ..... 284
11.3.2.1. Access to hardware resources ..... 285
11.3.2.2. Background execution ..... 285
11.3.2.3. Central availability ..... 285
11.3.3. Technical implementation of ODM implementation in Google Android ..... 286
11.3.4. Feature falls within the scope of Article 6(7) DMA ..... 289
11.3.5. Current access for Alphabet's own services and hardware ..... 289
11.3.6. Current access for third-party services and hardware ..... 289
11.3.6.1. System-privileged APIs ..... 289
11.3.6.2. RAM residency on Pixel devices and other preferential access to device resources 290
11.3.6.3. Alphabet is implementing new relevant functionality ..... 292
11.3.6.4. Uncertainty on future implementation ..... 293
11.3.7. Commission's assessment of Alphabet's response to the Preliminary Findings ..... 293
11.3.7.1. Concerning system functionality, current level of interoperability and integrity ..... 293
11.3.7.2. Concerning proportionality ..... 295
11.3.8. Measures that Alphabet should implement ..... 296
11.3.9. Timing for implementation of the measures ..... 298
11.4. Measures for background execution ..... 298
11.4.1. Introduction ..... 298
11.4.2. Description of the feature ..... 298
11.4.3. Technical implementation of background execution in Google Android ..... 300

11.4.3.1. Android's standard power restrictions: Doze and App Standby ..... 301
11.4.3.2. OEM proprietary background restrictions ..... 302
11.4.3.3. OEM exemptions from background restrictions ..... 304
11.4.4. Feature falls within the scope of Article 6(7) DMA ..... 306
11.4.5. Current access for Alphabet's own services and hardware ..... 306
11.4.6. Current access for third-party services and hardware ..... 308
11.4.7. Alphabet's response to the Preliminary Findings ..... 310
11.4.8. The Commission's assessment ..... 310
11.4.8.1. Concerning the existing level of interoperability and integrity ..... 310
11.4.8.2. Concerning proportionality ..... 313
11.4.9. Measures that Alphabet should implement ..... 315
11.4.10. Timing for implementation of the measures ..... 316
12. Measures for all features ..... 316
12.1. Implementation across the Google Android ecosystem ..... 316
12.2. User consent ..... 320
12.3. Integrity ..... 321
12.4. Eligibility of beneficiaries, applications and use cases ..... 324
12.4.1. Access to an OS feature should not be limited to specific use cases or beneficiaries ..... 324
12.4.2. Restricting feature access to a limited group of beneficiaries via OS-level eligibility conditions ..... 325
12.4.3. Alphabet's proposal to restrict the beneficiaries via an eligibility programme is not strictly necessary and proportionate to protect integrity and security under Article 6(4) and Article 6(7) DMA ..... 326
12.4.4. Exceptional circumstances that justify restricting access to a feature for effective interoperability ..... 329
12.4.5. Scope of restricted features ..... 331
12.4.6. Minimum safeguards that should apply when restricting feature access to eligible beneficiaries ..... 334
12.5. Equal effectiveness ..... 338
12.6. Free of charge ..... 344
12.7. Documentation and APIs ..... 344
12.8. Assistance and testing ..... 345
12.9. Future updates and new functionalities ..... 346
12.10. Reporting ..... 347
12.11. Waiver ..... 351
13. Proportionality ..... 351
14. Conclusion ..... 354

## LIST OF ABBREVIATIONS

| | |
|---|---|
| ACC | Android Compatibility Commitment |
| ACE | Android Context Engine |
| AI | Artificial Intelligence |
| APU | Accelerated Processing Unit |
| ASI | Android System Intelligence |
| AOHD | Always-On Hotword Detection |
| AOSP | Android Open Source Project |
| API | Application Programming Interface |
| APK | Android Package (file format) |
| ASR | Automatic Speech Recognition |
| BII | Built-In Intent |
| CHRE | Context Hub Runtime Environment |
| CDD | Compatibility Definition Document |
| CPS | Core Platform Service |
| CPU | Central Processing Unit |
| DMA | Digital Markets Act – Regulation (EU) 2022/1925 |
| DSP | Digital Signal Processor |
| EC | European Commission |
| EDPB | European Data Protection Board |
| EEA | European Economic Area |
| EMADA | European Mobile Application Distribution Agreement |
| ENISA | European Union Agency for Cybersecurity |
| EU | European Union |
| GDPR | General Data Protection Regulation – Regulation (EU) 2016/679 |
| GMS | Google Mobile Services |
| GPS | Global Positioning System |
| GPU | Graphics Processing Unit |
| HAL | Hardware Abstraction Layer |
| ICT | Information and Communication Technology |
| IPR | Intellectual Property Rights |
| LLM | Large Language Model |
| LPH | Long-press home |
| LPNH | Long-press navigation handle |
| MBA | Mobile Bundled Apps |
| ML | Machine Learning |
| NFC | Near Field Communication |
| NPU | Neural Processing Unit |
| ODM | On-Device Model |
| OEM | Original Equipment Manufacturer |
| OS | Operating System |
| PCC | Private Compute Core |

| | |
|---|---|
| PCS | Private Compute Services |
| PFs | Preliminary Findings |
| PHA | Potentially Harmful App |
| PSI | Pixel System Intelligence |
| QPR | Quarterly Platform Release |
| RAM | Random-Access Memory |
| RFI | Request for Information |
| SDK | Software Development Kit |
| TPU | Tensor Processing Unit |
| TFLM | Tensor Flow Lite for Microcontrollers |
| UI | User Interface |
| UWB | Ultra-wide band |
| VPN | Virtual Private Network |

**COMMISSION IMPLEMENTING DECISION**

**of 16.7.2026**

**pursuant to Article 8(2) of Regulation (EU) 2022/1925 of the European Parliament and of the Council on contestable and fair markets in the digital sector**

**(Case DMA.100220 – Alphabet - OS - Google Android - Art. 6(7) – SP – AI)**

(Only the English text is authentic)

THE EUROPEAN COMMISSION,

Having regard to the Treaty on the Functioning of the European Union,

Having regard to Regulation (EU) 2022/1925 of the European Parliament and of the Council of 17 September 2022 on contestable and fair markets in the digital sector and amending Directives (EU) 2019/1937 and (EU) 2020/1828 (Digital Markets Act),[1] and in particular Article 8 thereof,

After consulting the Digital Markets Advisory Committee,

Whereas:

## 1. INTRODUCTION

(1) In accordance with Article 8(2) of Regulation (EU) 2022/1925 ("DMA"), the Commission hereby specifies the measures that Alphabet Inc., together with all legal entities directly or indirectly controlled by Alphabet Inc. ("Alphabet"), has to implement in order to effectively comply with its obligations under Article 6(7) DMA as regards certain features of its operating system ("OS") Google Android as defined in the Commission Decision of 5 September 2023[2] and which are used by or available to Alphabet's artificial intelligence ("AI") services, such as Alphabet's Gemini.

(2) Pursuant to Article 8(1) DMA, it is up to the gatekeeper to ensure and demonstrate compliance with its obligations, and the measures implemented by the gatekeeper shall be effective in achieving the objectives of the Regulation. Recognising that for certain obligations gatekeepers may benefit from the Commission's guidance on the concrete implementation measures, the legislation enables the Commission to specify, in the specific circumstances of a gatekeeper and core platform service(s), such concrete implementation measures. On the basis of an extensive and continuous dialogue with Alphabet and third parties, the Commission lays down the measures annexed to this Decision.

[1] OJ L 265, 12.10.2022, pages 1-66.

[2] Decision C(2023) 6101 final.

## 2. PROCEDURE

(3) On 5 September 2023, the Commission adopted a decision designating Alphabet as a gatekeeper pursuant to Article 3 DMA (the "Designation Decision").[3] The Designation Decision lists eight core platform services (CPS) provided by Alphabet and which individually constitute an important gateway for business users to reach end users including Alphabet's operating system Google Android ("Google Android").

(4) Pursuant to Article 3(10) DMA, Alphabet has to comply, with the obligations laid down in the DMA in relation to all designated CPSs since 7 March 2024.

(5) On 7 March 2024, Alphabet submitted its compliance report pursuant to Article 11 DMA ("Alphabet 2024 Compliance Report").[4] On 6 March 2025, Alphabet submitted its second compliance report pursuant to Article 11 DMA ("Alphabet 2025 Compliance Report").[5] On 6 March 2026, Alphabet submitted its third compliance report pursuant to Article 11 DMA ("Alphabet 2026 Compliance Report").[6]

### 2.1. Chronology of the current proceedings

(6) Concerning Alphabet's compliance with Article 6(7) DMA specifically, the Commission has held an extensive regulatory dialogue with Alphabet since October 2023, conducted technical meetings[7] and sent requests for information ("RFIs").[8] Furthermore, the Commission had two meetings[9] with Alphabet to specifically discuss its approach to AI services on Google Android and sent five RFIs[10] concerning Alphabet's compliance with Article 6(7) DMA as regards AI services.

(7) On 26 January 2026, the Commission informed Alphabet of its intention to assist Alphabet in its compliance efforts in relation to certain features of Google Android for AI-powered services through the opening of specification proceedings pursuant to Article 8(2) DMA.

(8) On 27 January 2026, the Commission adopted a Decision opening proceedings pursuant to Article 20(1) DMA with a view to adopting an implementing act pursuant to Article 8(2) DMA specifying the measures that Alphabet has to implement in order to effectively comply with Article 6(7) DMA with respect to certain Google Android features for AI services (the "Decision opening the proceedings").[11]

(9) Following the Decision opening the proceedings, the Commission held a State of Play meeting with Alphabet on 5 February 2026 and seven in-depth technical meetings[12] based on an agreed agenda. The Commission sent 22 RFIs to Alphabet,

---

3 Decision C(2023) 6101 final.

4 Alphabet's 2024 Compliance Report, Doc ID […].

5 Alphabet's 2025 Compliance Report, Doc ID […].

6 Alphabet's 2026 Compliance Report, Doc ID […].

7 The Commission held technical meetings on 29 February 2024, 12 June 2024 and 19 June 2025.

8 The Commission sent the first RFI concerning Alphabet's compliance with Article 6(7) DMA to Alphabet on 20 October 2023. The Commission sent 5 other requests for information to Alphabet on 29 November 2023, 22 December 2023, 5 June 2024 and 7 August 2024, and 17 July 2025.

9 Technical meeting of 2 July 2025, Doc ID [...] and Technical meeting of 11 December 2025, Doc ID […].

10 Requests for information sent on 23 May 2025, 18 June 2025, 23 September 2025, 17 October 2025, 28 November 2025.

11 Decision C(2026) 585 final.

12 Prior to the issuance of the PFs, four technical meetings, each one lasting at least four hours, were held with Alphabet on 12 February 2026, 19 February 2026, 26 February 2026 and on 5 March 2026. At the request of Alphabet, the Commission agreed also to a meeting on 6 March 2026, which lasted three

including one RFI by decision on 27 January 2026 requesting specific internal documents.[13] Alphabet also made spontaneous submissions.[14]

(10) In the context of these proceedings, the Commission also engaged with third parties. On 5 February 2026, the Commission sent RFIs to nine Original Equipment Manufacturers ("OEMs") and three Mobile Network Operators ("MNOs") receiving respectively seven and three replies. On 20 February 2026, the Commission sent RFIs to eight third-party AI assistants, seven of which responded. In addition, the Commission held a total of 42 calls with third parties (e.g. OEMs, AI assistant, industry players) throughout these proceedings.

(11) On 24 April 2026, prior to the adoption of the Preliminary Findings ("PFs"), the Commission held with Alphabet a state-of-play conference call during which the Commission outlined the scope of the PFs.

(12) On 27 April 2026, the Commission communicated its PFs to Alphabet pursuant to Article 8(5) DMA,[15] and set out the measures that it considered Alphabet should implement in the Annex to these PFs ("Draft measures"). Pursuant to Article 34(1) DMA, the Commission provided Alphabet with the opportunity to comment on its findings and the proposed measures by 12 May 2026.

(13) On the day of the adoption of the PFs, Alphabet requested access to the Commission's file pursuant to Articles 8(1), 8(2) and 8(3) of the Commission Implementing Regulation (EU) 2023/814. On the same day and in reply to Alphabet's request, the Commission provided Alphabet with all documents mentioned in the PFs pursuant to Article 8(2) of the Commission Implementing Regulation (EU) 2023/814. On 28 April 2026, the Commission granted Alphabet's external legal counsel access to the case file at the Commission's premises via a data room, subject to the Terms of Disclosure laid down in the Commission Decision of 27 April 2026 pursuant to Article 8(3) of the Commission Implementing Regulation (EU) 2023/814.[16] Access to the data room was provided from 28 April until 30 April and from 4 May until 6 May 2026. Alphabet's external counsel prepared a confidential and non-confidential version of the data room report. The non-confidential version of the data room report ("Data room report") was shared with Alphabet on 6 May 2026.[17]

(14) On 27 April 2026, and in line with Article 8(6) DMA, the Commission published a non-confidential summary of the case and the draft measures to enable interested third parties to provide their comments, in particular as concerns their technical feasibility.[18]

(15) The public consultation was open until 13 May 2026, with extensions granted until 18 May 2026 upon request. The Commission received 53 contributions from third parties, end users and other interested third parties. All submissions were shared with

---

hours, and to a dedicated "Gemini: solutions workshop" meeting on 24 March 2026, which lasted four hours. On 8 April 2026, a follow-up two-hour meeting dedicated to solutions took place.

13 Decision C(2026) 604 final.

14 Alphabet submission of 3 March 2026, Doc ID […]; Alphabet submission of 10 March 2026, Doc ID […]; Alphabet submission of 17 March 2026, Doc ID […]; Alphabet submission of 17 March 2026, Doc ID […]; Alphabet submission of 23 March 2026, Doc ID […]; Alphabet submission of 3 April 2026, Doc ID […].

15 Decision C(2026) 2940 final.

16 Decision C(2026) 2948 final.

17 Data Room report, Doc ID […].

18 Public consultation available at https://digital-markets-act.ec.europa.eu/dma100220-consultation-proposed-measures-interoperability-google-android-article-67-dma_en

Alphabet on a rolling basis. The Commission held two calls with third parties following the public consultation, the minutes of these calls were shared with Alphabet.[19]

(16) On 1 May 2026, Alphabet requested an extension to respond to the PFs. On 7 May 2026, the Commission granted an extension until 26 May 2026, thereby granting Alphabet almost one full calendar month to submit its observations.[20]

(17) On 5 May 2026, Alphabet made a submission seeking clarifications from the Commission on the scope of the draft measures. The Commission provided clarifications on 8 May 2026.[21]

(18) On 11 May 2026, Alphabet submitted a paper with its observations on the draft measures.[22]

(19) On 19 May 2026, Alphabet made a submission on [Alphabet submission]. [23]

(20) On 20 May 2026, Alphabet submitted a mark-up of the draft measures in relation to the specific Google Android features.[24]

(21) On 13 May, 21 May, and 26 May 2026, the Commission held meetings with Alphabet to discuss Alphabet's feedback on the draft measures.

(22) On 26 May 2026, Alphabet provided its response to the PFs (Alphabet's response to the PFs).[25]

(23) On 29 May 2026, Alphabet submitted another mark-up of the draft measures in relation to the specific Google Android features.[26]

(24) On 4 June 2026, Alphabet submitted a paper on [Alphabet submission],[27] which the Commission discussed with Alphabet during a technical meeting on 5 June 2026.

(25) On 5 June 2026, the Commission shared with Alphabet updated measures on always on hotword detection.[28]

(26) On 7 June 2026, the Commission shared with Alphabet updated measures on context-aware intelligence.[29]

(27) On 9 June 2026, Alphabet submitted another paper [Alphabet submission], as a follow-up to the technical meeting of 5 June 2026.[30]

(28) On 9 June 2026, the Commission shared with Alphabet updated measures on restricted features, which were discussed in a meeting with Alphabet on 12 June 2026, together with the timelines for the implementation of the final measures.[31]

---

19 Agreed minutes of meeting with [Academics] of 29 May 2026, Doc ID […], Agreed minutes of meeting with [third party] of 26 May 2026, Doc ID […].

20 Alphabet's email of 1 May 2026 requesting a deadline extension for replying to the Commission's PFs and Commission's email of 7 May 2026 extending the deadline to reply to the PFs, Doc ID [...].

21 EC observations in response to Alphabet submission of 5 May 2026, Doc ID [...].

22 Alphabet submission of 11 May 2026, Doc ID [...].

23 Alphabet submission of 19 May 2026, Doc ID [...].

24 Alphabet submission "First Mark-up of the draft measure" of 20 May 2026, Doc ID […].

25 Alphabet's response to the PFs, Doc ID [...].

26 Alphabet submission "Alphabet Second Mark-up of the draft measure" of 29 May 2026, Doc ID [...].

27 Alphabet submission of 3 June 2026, [Alphabet submission], Doc ID [...].

28 Email of 5 June 2026, Doc ID [...].

29 Email of 7 June 2026, Doc ID [...].

30 Alphabet's submission of 9 June 2026, "Follow-up submission with respect to meeting of 5 June 2026", Doc ID [...].

(29) By way of courtesy, the Commission shared the draft final measures ("Draft final measures") with Alphabet on 12 June 2026.[32]

(30) On 15 June 2026, Alphabet submitted a mark-up of the restricted features measure and a [Alphabet submission].[33]

(31) On 16 June 2026, Alphabet submitted the priority issues on the draft final measures which were discussed in a technical meeting with the Commission on the same day.[34]

(32) On 17 June 2026, Alphabet submitted a mark-up version of the draft final measures on context-aware intelligence and ambient data, along with a submission on the [Alphabet submission].[35]

(33) On 18 June 2026, Alphabet submitted another paper on [Alphabet submission] .[36]

(34) On 22 June 2026, the Commission shared with Alphabet updated [Alphabet submission].[37] On 24 June 2026, Alphabet requested some clarifications on the updated measures [Alphabet submission], which were discussed in a meeting with Alphabet on 25 June 2026.[38]

(35) On 8 July 2026, the Commission held with Alphabet a courtesy conference call during which the Commission outlined the scope of the final measures annexed to this Decision, in particular focusing on the minor editorial changes compared to Draft final measures.

(36) On 10 July 2026, the Commission shared with Alphabet a courtesy copy of the final measures annexed to this Decision.[39]

(37) On 15 July 2026, the Digital Markets Advisory Committee took place.[40]

### 2.2. Commission's engagement with Alphabet

(38) Against the background of the regulatory dialogue with Alphabet concerning the implementation of Alphabet's obligation under Article 6(7) DMA, which has been ongoing since the designation of Alphabet's Google Android operating system, and intensified as of July 2025 in particular as concerns AI-powered services, the Commission conducted the specification proceedings in a diligent and transparent manner, in accordance with the DMA.

(39) *First*, prior to opening proceedings, the Commission had extensively engaged with Alphabet concerning Alphabet's approach to AI services on Google Android. The Commission sent an initial RFI concerning Alphabet's compliance with Article 6(7) DMA as regards AI services on 23 May 2025. The Commission subsequently sent

---

31 Email of 9 June 2026, Doc ID [...] and Technical meeting of 12 June 2026, lasting two hours, slide deck, Doc ID [...].

32 Email of 12 June 2026, Doc ID [...].

33 Alphabet submission of 15 June "Google restricted features measures", Doc ID [...], Alphabet email of 15 June [Alphabet submission], Doc ID [...].

34 Alphabet submission of 16 June "Draft measure – Priority Issues" and Technical meeting of 16 June 2026 to discuss Priority issues of the draft measures, Doc ID [...]. The meeting lasted one hour.

35 Alphabet submission "mark-up of the context aware intelligence measure", Doc ID [...], Alphabet submission "mark-up of the ambient data measure", Doc ID [...] and Alphabet submission [Alphabet submission] Doc ID [...].

36 Alphabet submission [Alphabet submission] of 18 June 2026, Doc ID [...].

37 Email of 22 June 2026, Doc ID [...] and Doc ID [...].

38 Email of 24 June, and follow-up meeting with Alphabet on 25 June, Doc ID [...].

39 Email of 10 July 2026, Doc ID [...] and Doc ID [...].

40 The opinion of the Digital Markets Advisory Committee can be found on the Commission's Comitology Register under meeting code CMTD(2026)1165: https://ec.europa.eu/transparency/comitology-register/screen/committees/C114400.

four additional RFIs and conducted two technical meetings with Alphabet on this topic.[41]

(40) *Second,* following months of extensive exchanges, the Commission informed Alphabet that technical meetings would be organised right after the opening of proceedings to make the process as constructive as possible. The Commission scheduled four technical meetings and, at Alphabet's request, agreed to hold three additional technical meetings prior to the adoption of the PFs. The timeline and agenda for these meetings were discussed well in advance with Alphabet in order to allow sufficient time to prepare the meetings and discuss each aspect within the scope of the proceedings. For most technical meetings, which lasted approximately four hours each, the Commission also scheduled fallback meetings on the following day with Alphabet to ensure sufficient time for discussion. Alphabet did not express an interest in using this additional time, even when explicitly asked. During these technical meetings, Alphabet presented slide decks setting out its views on the relevant features and on effective interoperability. In addition to these in-depth technical meetings, the Commission also agreed to hold weekly meetings with Alphabet to take stock of pending technical and organisational issues throughout the proceedings.

(41) *Third*, the Commission issued 19 RFIs to collect the relevant information to inform the Commission's assessment prior to the issuance of the PFs. Whilst acknowledging the short timelines imposed by the DMA, the Commission set appropriate deadlines in its RFIs. All of Alphabet's requests for extensions to reply to RFIs were accommodated.

(42) *Fourth*, the Commission engaged extensively with Alphabet regarding its compliance with the RFI by decision of 27 January 2026, which required Alphabet to produce internal documents from a limited number of custodians, identified by reference to keywords, as well as certain categories of documents defined by description. Alphabet's internal documents provided insight into Alphabet's intentions and the likelihood of new features, which are relevant to these proceedings, being developed by Alphabet. Contrary to Alphabet's claim that the PFs rely on hypothetical future developments, exploratory internal discussions, and speculative assertions,[42] the Commission did not base its assessment on hypothetical future functionalities mentioned in Alphabet's internal documents but used these documents to conduct a reasonable forward-looking analysis of the likelihood of Alphabet developing new functionalities or further integrating with features already available to its services, in order to ensure that the proposed measures do not become obsolete or ineffective as a result of Alphabet's future strategies.

(43) *Fifth*, Alphabet's six spontaneous submissions throughout these proceedings[43] and the discussions held with Alphabet informed the Commission's assessment and the measures specified in this Decision.

(44) *Sixth*, as described in Section 2.1 of this Decision, the Commission continued its engagement with Alphabet after the adoption of the PFs. In particular, the

---

41 Requests for information sent on 17 July 2025, 23 September, 17 October 2025, 28 November 2025 and Technical meeting Alphabet/Commission of 2 July 2025 which lasted two hours, Doc ID [...], and Technical meeting Alphabet/Commission of 11 December 2025 which lasted four hours, Doc ID [...].

42 Alphabet's response to the PFs, Section VIII, Doc ID [...], paragraph 388.

43 Alphabet submission of 3 March 2026, Doc ID [...]; Alphabet submission of 10 March 2026, Doc ID [...]; Alphabet submission of 17 March 2026, Doc ID [...]; Alphabet submission of 17 March 2026, Doc ID [...]; Alphabet submission of 23 March 2026, Doc ID [...]; Alphabet submission of 3 April 2026, Doc ID [...].

Commission reviewed several mark-ups of the draft measures, which Alphabet submitted after the deadline for replying to the PFs, and held meetings with Alphabet to discuss its feedback on the proposed measures. Numerous suggestions from Alphabet have been adopted either verbatim or in principle.

(45) *Seventh*, for the sake of completeness, the Commission sent Alphabet, on a rolling basis, all submissions made by third parties in the public consultation, as well as the minutes of all follow-up meetings with third parties including after the adoption of the PFs.[44]

(46) *Eighth*, in the spirit of good cooperation, the Commission sent the draft final measures to Alphabet on 12 June 2026, thereby granting Alphabet an additional opportunity to make observations. Following this, the Commission continued to engage with Alphabet regarding its observations on the draft final measures.

(47) Alphabet engaged with the Commission on potential solutions for interoperability gaps outlined in the technical meetings only at a very late stage, with the first meeting on that subject taking place on 8 April 2026, that is, only 19 days before the adoption of the PFs. Nevertheless, the Commission assessed those ideas and, where appropriate, used that assessment as an input in designing the measures annexed to this Decision.[45]

(48) In light of the above and Section 2.1 of this Decision, the Commission considers that it has discharged its duties under Article 41 of the Charter of Fundamental Rights of the European Union ("the Charter"), which guarantees the right to good administration. Alphabet's claim that it "*was deprived of a meaningful opportunity to address the relevant legal framework prior to the adoption of the PFs"* is therefore unfounded.[46] This is not only because the Commission duly considered all Alphabet's submissions throughout these proceedings, but also because Alphabet has been able to comment on the PFs and the proposed measures pursuant to Article 34 DMA. Moreover, it is established case-law that, even in adversarial procedures that may result in a fine (which is not the case here), the duty to state reasons does not require the Commission to reply to every individual submission or argument raised by an undertaking during the administrative procedure.[47] In its reply to the PFs, Alphabet was given the opportunity to respond to each of the reasons put forward by the Commission to justify the measures it intended to adopt. and the Commission took into account Alphabet's remarks in a number of cases leading to amendments of the measures originally envisaged.

### 2.3. Commission's engagement with third parties

(49) As set out in Section 2.1, the Commission engaged extensively with third parties both before the opening of these proceedings and throughout their duration. The Commission engaged with third parties through meetings, calls and written submissions. The input collected from third parties has been used to inform the specification by providing qualitative insights and considering, to the extent relevant,

---

44 Agreed minutes of meeting with [Academics] of 29 May 2026, Doc ID [...], Agreed minutes of meeting with [third party] of 26 May 2026, Doc ID [...]. Submission to the public consultation of [AI service provider] sent by email to Alphabet, Doc ID [...] and Doc ID [...].

45 For instance, the Commission tested with third parties prior the adoption of the PFs, the hotwords detection arguments submitted by Alphabet. Alphabet submission of 23 March 2026, "Hotword Detection on Connected Devices", Doc ID [...].

46 Alphabet's response to the PFs, Doc ID [...], paragraph 383.

47 See in that regard, judgment of 9 April 2019, Qualcomm v Commission, T-371/17, EU:T:2019:232, paragraphs 70-71 and case law cited. See also judgment of 24 May 2023, Meta v Commission, T-451/20, EU:T:2023:276, paragraph 160.

the responses independently, not as a statistical measure of consensus. Specifically, the Commission notes the following types of engagement:

(a) **Engagement with OEMs**: Given the complexities of the Google Android ecosystem and Alphabet's relationship with OEMs (see Section 5), the Commission sent RFIs to nine OEMs, representing almost the entire sales of Google Android mobile devices in the EU, and to three MNOs in order to assess the potential impact on OEMs of the measures envisaged in relation to features within the scope of these proceedings, in particular those whose implementation would require OEMs to engage with Alphabet. The Commission also held calls with OEMs and received submissions, notably from [OEM], which alone accounts for […] sales of Google Android mobile devices in the EU. The measures specified in this Decision, in particular in relation to ODM and hotword invocation, take OEM feedback into account. Contrary to Alphabet's claim that the PFs omit reference to stakeholders that would be impacted by the proposed measures,[48] such as OEMs, the Commission took OEMs' feedback into account and also explained the potential impact of the measures on OEMs in Section 6.1.2 of this Decision.

(b) **Engagement with AI Assistants:** The Commission sent RFIs to eight companies offering AI Assistants (i.e. all current major players in the AI mobile space) which are the potential beneficiaries of the measures to which these proceedings relate. The Commission also held several conference calls with them as they are best placed to identify – in practical terms – the interoperability gaps they face when competing with Google's Gemini AI assistant.

(a) **Engagement with academics:** The Commission also engaged with academics on highly technical matters arising in these proceedings, in particular the model for hotword detection.[49]

(b) **Public Consultation:** The Commission published the proposed measures for public consultation, thereby seeking feedback from the widest possible range of interested third parties. The Commission received 53 contributions from industry players, trade associations and academics. This engagement was also stimulated by Alphabet, who, as explained in Section 2.4, actively motivated certain third parties to take positions in the public consultation.

(50) In light of the above and taking into account the extent of the Commission's engagement with third parties, the Commission considers that it has discharged its duties under Article 41 of the Charter, which guarantees the right to good administration. Contrary to Alphabet's claim that the PFs do not reflect the input of any "*industry bodies such as EDPB*",[50] the Commission is not under a positive obligation to consult or engage with other EU bodies pursuant to the DMA. The DMA sets out a framework for the role of the various public stakeholders in the implementation of this Regulation.[51] Where proceedings may have a specific legal impact, the Commission proactively engages with the relevant bodies. For example, in the parallel specification proceedings concerning Alphabet's obligations pursuant

48 Alphabet's response to the PFs, Doc ID [...], paragraph 379.

49 Agreed minutes of meeting with [Academics] of 29 May 2026, Doc ID [...].

50 Alphabet's response to the PFs, Doc ID [...], paragraphs 378 and 383.

51 Article 37 DMA mentions that the Commission may consult national authorities where appropriate on any matter relating to the application of the DMA. Moreover, Article 40 DMA establishes the high-level group composed by European bodies and networks that may provide the Commission with advice and expertise in the areas falling within the competences of its members.

to Article 6(11) DMA, the Commission has engaged with the European Data Protection Board ("EDPB") in view of the specific interplay between the measures at stake and data protection rules. Alphabet's claims are unfounded because Alphabet has not demonstrated that the implementation of the measures will have any impact such as to require consultation with specific bodies, in addition to the public consultation.

### 2.4. Alphabet's participation in the public consultation

(51) During these proceedings, Alphabet also reached out to third parties in the context of the public consultation and outlined its position on the scope of these proceedings. Based on the information available to the Commission,[52] Alphabet reached out to at least 49 stakeholders,[53] including 11 OEMs,[54] eight mobile carriers,[55] two industry players,[56] 16 think tanks,[57] two academics,[58] four EU and international governmental bodies,[59] two national regulators,[60] four trade bodies,[61] and four organisations active in the security field.[62] Over a third of the stakeholders contacted by Alphabet responded to the public consultation.[63] One of the participants – a consultancy – had been financially supported by Alphabet for its response to the public consultation.[64]

(52) Alphabet's outreach consisted in sending out emails, setting up calls and in a few instances actively reviewing submissions of third parties before they were submitted to the Commission.[65] The content and the ask of Alphabet's outreach varied between

---

52 RFI PO 20 sent to Alphabet on 3 June 2026 concerning Alphabet's engagement with third parties in relation to the public consultation on the draft measures launched on 28 April 2026, Doc ID [...].

53 Alphabet's reply to RFI PO 20 of 3 June 2026, Document "Annex Q1", Doc ID [...].

54 OEMs contacted by Alphabet: [OEMs]. Alphabet's reply to RFI PO 20 of 3 June 2026, Document "Annex Q1", Doc ID [...].

55 Mobile carriers contacted by Alphabet: [Mobile carriers]. Annex Q1 of RFI 20 to Alphabet of 3 June 2026, Doc ID [...].

56 Industry payers contacted by Alphabet: [Industry payers]. Alphabet's reply to RFI PO 20 of 3 June 2026, Document "Annex Q1", Doc ID [...].

57 Think Tanks contacted by Alphabet: [Think Tanks]. Alphabet's reply to RFI PO 20 of 3 June 2026, Document "Annex Q1", Doc ID [...].

58 Academics contacted by Alphabet: [Academics]. Alphabet's reply to RFI PO 20 of 3 June 2026, Document "Annex Q1", Doc ID [...].

59 EU and international governmental bodies contacted by Alphabet: [EU and international governmental bodies]. Alphabet's reply to RFI PO 20 of 3 June 2026, Document "Annex Q1", Doc ID [...].

60 National regulators contacted by Alphabet: [National regulators] Alphabet's reply to RFI PO 20 of 3 June 2026, Document "Annex Q1", Doc ID [...].

61 Trade bodies contacted by Alphabet: [Trade bodies]. Alphabet's reply to RFI PO 20 of 3 June 2026, Document "Annex Q1", Doc ID [...].

62 Organisations active in the security field: [Organisations active in the security field]. Alphabet's reply to RFI PO 20 of 3 June 2026, Document "Annex Q1", Doc ID [...].

63 The respondents to the public consultation that were contacted by Alphabet were: OEMs: [OEM]. Think Tanks: [think tanks]. Academics: [academics]. Trade bodies: [trade bodies]. Organisations active in the security field: [organisations active in the security field].

64 Submission to the public consultation of [third party] includes the following statement: "Google provided financial support for this submission", Doc ID [...].

65 See for instance, email sent from [third party's] legal counsel to Alphabet's external counsel [direct quotes from emails sent by Alphabet to third parties]" "Alphabet's reply to RFI PO 20 of 3 June 2026, Document "Annex Q1c.58", Doc ID [...], Email exchange between Google's employee and [third party] includes "[direct quotes from emails sent by Alphabet to Think Tank]"Alphabet's reply to RFI PO 20 of 3 June 2026, Document "Annex Q1c.17", Doc ID [...], Email exchange between Alphabet and [Trade bodies] , Alphabet's reply to RFI PO 20 of 3 June 2026, Document "Annex Q1c.10", Doc ID [...] and email exchange between [trade bodies] and Alphabet, Alphabet's reply to RFI PO 20 of 3 June 2026, Document "Annex Q1c.39", Doc ID [...].

stakeholders. When contacting OEMs, [details of Alphabet's OEM engagement strategy], [direct quotes from emails sent by Alphabet to third parties]".[66]

(53) Alphabet sent the following email to [third party]: [direct quotes from email from Alphabet about their submissions to the public consultation].[67] [Third party] followed the brief and [details of Alphabet's engagement with the third party] before submitting its response to the public consultation.[68]

(54) Similarly, Alphabet sent the following email to consultants: [direct quotes from email from Alphabet to third parties about their submissions to the public consultation]. One consultant replied asking Alphabet to provide their input before officially submitting the response to the public consultation: [direct quotes from email from Alphabet to consultant about their submissions to the public consultation].[69]

(55) Alphabet also reached out to trade bodies and think tanks of which it is a member or which it financially supports (e.g. [trade bodies contacted by Alphabet]).[70]

(56) In its reply to the PFs, Alphabet quoted very frequently 14 submissions from third parties that it had contacted in the context of the public consultation. The Commission took this into account in its assessment of the replies received from these participants, as well as the weight accorded to their contributions in the overall outcome of the public consultation.

## 3. RATIONALE FOR SPECIFICATION PROCEEDINGS

(57) The purpose of these proceedings is to determine the measures that Alphabet is to implement in order to ensure effective interoperability pursuant to Article 6(7) DMA with respect to some features that are used by or available to Alphabet's services, including, but not limited to, some AI-powered services such as Gemini and Google Assistant (hereafter together referred to as "Google AI Assistants").

(58) This includes aspects such as the concrete interoperability solutions, their technical and contractual implementation, and the modalities of access to the operating system to be provided to third parties.

### 3.1. Importance of AI-powered services

(59) AI has undeniably resulted in one of the largest shifts in recent consumer technology history. Used across multiple sectors, AI-powered services facilitate, among others, productivity, communication, e-commerce, customer support, professional services, and workflow automation. The rapid adoption of related technologies reflects their broad societal and economic importance, as well as their role in shaping digital ecosystems. According to publicly available forecasts, the overall European AI sector is projected to reach EUR 120.19 billion in 2026 and is projected to grow to

66 Alphabet's reply to RFI PO 20 of 3 June 2026, Document "Annex Q1c.43", Doc ID [...].

67 Email sent from Alphabet's […] to [third party], Alphabet's reply to RFI PO 20 of 3 June 2026, Document "Annex Q1c.59", Doc ID [...], page 1.

68 Email sent from Alphabet's […] to [third party], Alphabet's reply to RFI PO 20 of 3 June 2026, Document "Annex Q1c.59", Doc ID [...], page1 and [third party] submission to the public consultation, Doc ID [...].

69 Email exchange between Alphabet's employee and [consultant], Alphabet's reply to RFI PO 20 of 3 June 2026, Document "Annex Q1c.17", Doc ID [...], page 2 and page 4.

70 Trade bodies and Think Thanks contacted by Alphabet: [trade bodies and think thanks]. Alphabet's reply to RFI PO 20 of 3 June 2026, Document "Annex Q1", Doc ID [...].

EUR 435.81 billion by 2032, with a predicted annual growth rate of approximately 27%.[71]

(60) AI-powered services have gradually permeated mobile ecosystems, with various kinds of AI assistants becoming a crucial entry point to use digital services on smart mobile devices, for both consumers and businesses. These AI assistants are capable of handling tasks on a user's behalf, through complex and natural-language queries, sometimes voice-based, that trigger various actions, often highly personalised. As technology advances, AI assistants will become more independent, acting autonomously as "agents", taking actions, planning tasks and operating tools on behalf of end users. This is generally referred to as "agentic AI". For example, agentic AI can, with a single prompt, execute multi-step tasks, such as searching flights, comparing prices, summarising the options available and booking the chosen flight, all fully automated.

(61) While the rise of AI-powered services has attracted considerable attention and resources by the largest companies in the technology space, it has also provided the incumbent players the opportunity to use AI services to further attract and lock consumers into their ecosystem, especially when those AI services have exclusive access to "complementary assets", i.e. platforms and services that support the capabilities of AI services.[72]

### 3.2. Alphabet's unique position in the AI value chain

(62) Alphabet is currently one of the most vertically integrated companies in the AI space, becoming a strong, if not a key, player across every layer of the AI value chain. Alphabet owns different components of the AI value chain, in particular:[73]

(a) **Infrastructure**: Alphabet designs its own AI chips, the Tensor processing unit (TPU), in-house and uses them in its Google Cloud infrastructure and (mobile versions of them) in its Pixel Android mobile devices.

(b) **Foundation models:** Alphabet owns the Gemini "family" of Large Language Models (LLMs), including the Gemini Nano on-device model. These are used in Alphabet's ecosystem, with e.g. Gemini Nano being pre-installed on Android mobile devices. In addition, some models even power other ecosystems, such as Apple's Siri AI assistant.[74]

(c) **Platform integration**: Alphabet's numerous products and services (e.g. Gemini) leverage AI integrations with its operating system Google Android and its browser Chrome, which it controls.

(d) **Distribution**: Alphabet distributes and propagates the use of its AI services through first-party (1P) consumer apps such as Gmail, Google Search, Google Photos, and YouTube as well as Google Android features.

---

71 See https://www.statista.com/outlook/tmo/artificial%20intelligence/europe?currency=EUR, Doc ID [...], accessed on 9 April 2026.

72 See Azoulay et al. "Old Moats for New Models: Openness, Control, and Competition in Generative AI", National Bureau of Economic Research, Working Paper 32474, May 2024, DOI 10.3386/w32474. https://www.nber.org/system/files/working_papers/w32474/w32474.pdf, Doc ID [...].

73 See Alphabet, "The OS designed for AI", March 2025. https://www.android.com/static/pdf/gemini-whitepaper.pdf, Doc ID [...].

74 See "Joint statement from Google and Apple", 12 January 2026, https://blog.google/company-news/inside-google/company-announcements/joint-statement-google-apple/, Doc ID [...], accessed on 9 April 2026.

(63) Alphabet's internal documents confirm that it is [direct quote describing Alphabet's commercial and strategic discussions].[75]

(64) Alphabet's flagship AI service Gemini is a prime example of the global uptake of AI services. According to Alphabet's own data, the Gemini app now has over 900 million monthly active users worldwide.[76]

(65) Gemini's rapid uptake is supported by its deep integration with Alphabet's ecosystem of services, across general search (Google Search), email (Gmail), productivity tools (Google Workspace), messaging (Google Messages) and other consumer-facing services within Alphabet's ecosystem. Such deep integration enables Alphabet to expose its end users to Gemini, and conversely, Gemini end users to Alphabet's ecosystem of different services.

(66) Alphabet is well aware of this comparative advantage: [direct quote describing Alphabet's commercial and strategic discussions].[77]

(67) The rapid growth of AI-based services, combined with Alphabet's unique position in the AI value chain is contributing to further entrenchment of the already strong position of Alphabet's ecosystem, in particular in the mobile space. The key role of the operating system, and in particular Google Android, in this transformation is well understood at Alphabet: in the agentic era, [direct quote describing Alphabet's commercial and strategic discussions].[78] The central role of Google Android in Alphabet's vision for its AI services is perhaps best revealed by the introduction of "Gemini Intelligence" for Android 17 in mid-2026, where AI is becoming even more deeply integrated into Google Android as, in Alphabet's own words, "*Android transitions from an operating system into an intelligence system*".[79] Gemini Intelligence prominently appears on the Android website and in communications about Android, underlying the extent to which this Android integration is meant to heavily favour use of Gemini.[80] Internal slide decks further reveal this seeming equivalence of Gemini and AI on Android: [direct quote describing Alphabet's commercial and strategic discussions].[81]

---

75 Alphabet internal document, "Alphabet internal document" will be used to refer to any Google internal documents obtained in response to RFI by decision of 27 January 2026 (Commission Decision C(2026) 604 final), amended on 5 February 2026 (Commission Decision C(2026) 857 final) or to RFI of 5 August 2025, amended on 10 September 2025. Alphabet internal document, Document [Alphabet internal document], dated 4 January 2026, Doc ID […], page 2.

76 See "I/O 2026: Welcome to the agentic Gemini era", published on 19 May 2026, https://blog.google/intl/en-nz/company-news/sundar-pichai-io-2026/#momentum, Doc ID [...]. The reported number is very likely a conservative estimate of active user numbers, since it counts only users of AI assistant Gemini app and does not reflect all those users that are exposed to and use Gemini when integrated into other first-party services, such as for instance Google Search or Google Maps.

77 [Alphabet internal document], Doc ID […].

78 [Alphabet internal document], Doc ID […], page 2.

79 "A smarter, more proactive Android with Gemini Intelligence", Google, 12 May 2026, https://blog.google/products-and-platforms/platforms/android/gemini-intelligence/, accessed on 1 June 2026, Doc ID [...]. See also Alphabet internal document, Document [Alphabet internal document] of 24 February 2026, Doc ID […], page 5.

80 Android.com home page, showing a prominent headline for Gemini Intelligence, https://www.android.com/, accessed 1 June 2026, Doc ID [...]. The information page for Gemini Intelligence is also hosted under the Android.com website instead of a dedicated Gemini website, https://www.android.com/gemini-intelligence/, accessed 1 June 2026, Doc ID [...]. "A smarter, more proactive Android with Gemini Intelligence", Google, 12 May 2026, https://blog.google/products-and-platforms/platforms/android/gemini-intelligence/, accessed on 1 June 2026, Doc ID [...].

81 Alphabet internal document, Presentation [Alphabet internal document] of January 2026, Doc ID [...]. Alphabet internal document, Presentation [Alphabet internal document] of January 2026, Doc ID [...],

(68) Alphabet already benefits from a large userbase for its Gemini AI service being embedded in the Google app which includes Google search and Gemini (see Section 5.3)[82] on virtually all Google Android powered smart mobile devices. The Google app comes pre-installed on all Pixel devices (which are distributed by Alphabet) and virtually all third-party OEM smartphones and tablets running Google Android (see Section 5.4.2). Alphabet requires that all such OEMs grant the Google app extensive permissions, giving Gemini broad access to Android features, sometimes on an exclusive basis (see Sections 5.2 and 5.4). On recent devices, Gemini is set as the default and automatically enabled AI assistant (see Section 5.4.2). In older devices, Gemini is still readily accessible via a "hotword" ("Hey Google") activation or by-long pressing the power button, enabled by the pre-installation of the Google app (Section 5.3).

(69) Alphabet recognises that, due to the access to Google Android features, Gemini users benefit from a more integrated and improved user experience, and Alphabet has been building Google Android in function of Gemini. A white paper about Android and AI states that "*Android infuses Gemini directly into its core*".[83] [Direct quote describing Alphabet's commercial and strategic discussions].[84] [Direct quote describing Alphabet's commercial and strategic discussions].[85] Alphabet even appears to see this as a way to restrict the use of competing services: even though they recognise that [direct quote describing Alphabet's commercial and strategic discussions].[86] The race to be the best AI assistant is fierce, and Alphabet's strategy voices an ambition to have [direct quote describing Alphabet's commercial and strategic discussions].[87]

### 3.3. Importance of effective interoperability with Google Android for AI-based services

(70) With a high number of end users in the European Union, mobile devices powered by Google Android represent a critical access point for providers of stand-alone AI services to reach end users. Around 60% of mobile users in Europe own a Google Android smart mobile device.[88] According to publicly available data, over 67% of smartphone users at global level interact with voice commands at least once per month, while 38% use voice assistants daily, which suggests that usage is not merely occasional but part of regular mobile behaviour.[89] Finally, there is a clear consumer

---

page 2. Alphabet internal document, Presentation [Alphabet internal document] of December 2025, Doc ID [...]Alphabet internal document, Presentation [Alphabet internal document] of December 2025, Doc ID [...], page 4.

82 The Google app is different from the traditional Alphabet consumer app services such as for instance Gmail or YouTube. The Google app is a "central hub" that can be used by end users to access through one single "door" Google search, AI-powered tools (Gemini), news and personalised content.

83 Alphabet, "The OS designed for AI", March 2025. https://www.android.com/static/pdf/gemini-whitepaper.pdf, Doc ID [...], accessed on 9 April 2026.

84 Alphabet internal document, Document [Alphabet internal document] of November 2025, Doc ID […], page 1.

85 [Alphabet internal document], Doc ID […], page 2.

86 [Alphabet internal document], Doc ID […], page 4.

87 [Alphabet internal document], Doc ID […], page 4.

88 See "Mobile operating system market share in Europe" from gs.statcounter.com for March 2026, accessed on 9 April 2026, Doc ID [...].

89 See "Voice Assistant Market | Global Industry Analysis 2034", https://www.marketreportsworld.com/market-reports/voice-assistant-market-14722266, Doc ID [...], accessed on 10 April 2026.

demand for AI-powered devices in 2025[90] supported by an industry-wide push toward AI.[91]

(71) Interoperability with Google Android is therefore critical for third-party AI service providers to compete with Alphabet's AI-based services, such as Gemini, in this space. For example, any AI provider that wants to offer its services to users of smart mobile devices powered by Google Android needs its AI assistant to be able to interact with the device hardware, have access to invocation mechanisms such as voice-based hotword activation, the user's context and data, integration with other apps, and underlying system services. Such access ensures that alternative AI assistants can provide at least comparable performance, responsiveness, visibility, functionality, and utility to end users than Alphabet's own AI assistant Gemini and thereby compete with the vertically integrated service. This includes access to end users' information contained in Alphabet services with which the end users interact via Gemini. Without this level of ecosystem access, third-party AI assistants will not be able to compete with Gemini.[92]

(72) Providing access to the same Google Android features as the ones that are used by or are available to Alphabet's own AI services is necessary to ensure contestability of AI services in the Google Android ecosystem. Without effective interoperability and equal access to the relevant OS features there is a risk of significant barriers to market entry and expansion, and ultimately limited user choice. Lowering such barriers to entry and expansion increases the incentives of AI service providers to innovate and improve their products. Increasing user choice allows end users to use the AI services that suit their personal interests best. Specialised AI assistants (e.g., in education, legal, regional languages, enterprise verticals) should also have the capabilities to integrate into the user experience that Android device owners expect, ultimately improving innovation.

(73) Alphabet has the ability to control and restrict the access of third-party AI services to Google Android features that are available to or used by Alphabet's own AI services and to grant an advantage to its own services, including AI services (see Section 5.2.1). Indeed, Alphabet internally establishes as [direct quote describing Alphabet's commercial and strategic discussions].[93] Many third parties are developing innovative AI services, which could become crucial points for users to access and communicate with Google Android and apps running on it. These third-party AI services should enjoy the same close integration with Google Android as Alphabet's AI services, including the close integration with other services within Alphabet's ecosystem.

---

90 See "Global smartphone shipments rise in third-quarter on AI-driven upgrades | Reuters", https://www.reuters.com/business/global-smartphone-shipments-rise-third-quarter-ai-driven-upgrades-2025-10-13/, Doc ID [...], accessed on 10 April 2026.

91 See "Report from the Mobile World Congress 2025 in Barcelona reporting the view of OEMs, chip and wearable manufactures", https://www.idc.com/resource-center/blog/mwc-2025-the-future-of-mobile-devices-unfolds/, Doc ID [...], accessed on 10 April 2026.

92 Access to this type of data allows a form of interoperability that the Crémer, de Montjoye and Schweitzer report on Competition Policy for Digital Era refers to as "data interoperability", which generally relies on access to features of the relevant services hosting the data. European Commission: Directorate-General for Competition, *Competition policy for the digital era*, Publications Office, 2019, DOI 10.2763/407537, https://data.europa.eu/doi/10.2763/407537, Doc ID [...], pages 8, 9, 28, 31, 58-60 and 125.

93 [Alphabet internal document], Doc ID […], page 1.

### 3.4. Current degree of interoperability for AI-based services on Google Android

(74) According to the Alphabet Compliance Reports (see recital (5) above), to comply with Article 6(7) DMA, Alphabet has created a portal that developers can access to submit interoperability requests concerning the interoperability with Google Android. In addition, in the Compliance Reports, Alphabet claims that the Android Open Source Project ("AOSP") is "*fully interoperable for both first-party and third party products*" as it is based on open-source code.[94]

(75) None of the measures Alphabet outlined in its Compliance Reports to demonstrate compliance with Article 6(7) DMA specifically refers to or implements an interoperability solution for AI-based services.

(76) Furthermore, several third parties that offer AI services expressed concerns about the level of interoperability with respect to Alphabet's features for AI services. For instance:

(a) [AI service provider] indicated that "*third-party (3P) developers of AI assistants have limited access to privileged operating system features and core Google services that are at OS level. These issues have become apparent since the appearance of Gemini, in particular as it is set as the default AI assistant for Google Android.*".[95]

(b) [AI service provider] submitted that "*Google subtly controls and integrates features to funnel users into its ecosystem, such as restricting access to critical APIs and hooks*".[96]

(c) [Mobile operator] reported that "*it is currently very difficult, if not impossible, to distribute AI assistants with equal access and permissions compared to Gemini on Android* ".[97]

(d) [AI service provider] indicated that "[AI service provider]*expects that Gemini will be set as the default assistant in all or almost all cases, and only Google can therefore effectively use those exclusive features for its assistant*".[98]

(e) [AI service provider] expressed concerns that "*hyperscalers [are] integrating their own AI offerings with their first-party apps and services, meaning that users are continuously steered to using the hyperscalers' AI solutions rather than third-party ones*".[99]

(f) A business user pointed out that while Android generally makes the necessary features available "*however, there are restrictions […] often due to the contractual relationship between OEMs and Android* ".[100]

(g) [AI service provider] confirmed that "*there is still a lack of interoperability with respect to Alphabet's features for AI services and AI assistants*".[101]

---

94 Alphabet 2026 Compliance Report, Doc ID [...], page 193, Google Android Chapter, Section XVII.B, paragraph 95.

95 Agreed minutes of the meeting with [AI service provider] of 18 December 2025, Doc ID [...].

96 Agreed minutes of the meeting with [AI service provider] of 28 October 2025, Doc ID [...], paragraph 26.

97 Agreed minutes of the meeting with [Mobile operator] of 17 November 2025, Doc ID [...], paragraph 1.

98 Agreed minutes of the meeting with [AI service provider] of 2 March 2026, Doc ID [...], paragraph 3.

99 Agreed minutes of the meeting with [AI service provider] of 13 February 2026, Doc ID [...], paragraph 2.

100 Agreed minutes of the meeting with Business User of 5 September 2025, Doc ID [...], paragraph 1.

101 Agreed minutes of the meeting with [AI service provider] of 20 March 2026, Doc ID [...], paragraph 1.

(h) Finally, the [trade body] stressed that "*the power to customise the AI-based services integrated into the device lies with the OS provider*".[102]

(77) For all the reasons set out above, achieving effective interoperability pursuant to Article 6(7) DMA is necessary to ensure contestability and fairness for services, software and hardware operating or connected with Google Android, which includes the interoperability of AI services.[103]

(78) Alphabet is aware of the existence of an interoperability gap and the risks to its business strategy should this gap fully disappear. Consistently, Alphabet's internal documents indicate that Alphabet's incentives and plans to enable effective interoperability are limited: "[direct quote describing Alphabet's commercial and strategic discussions][104] Aware of its obligations under the DMA, Alphabet set out as goal for 2026 to [direct quote describing Alphabet's commercial and strategic discussions] for third-party services, and deployed a strategy to do so by [direct quote describing Alphabet's commercial and strategic discussions].[105]

### 3.5. Use of specification proceedings

(79) Google Android constitutes a complex and ever evolving operating system, characterised by multiple layers of abstraction, granular permission systems, proprietary system Application Programming Interfaces ("APIs"),[106] and tight coupling with core applications and hardware components (see Section 5). In this context, AI services are by design deeply integrated with the OS, requiring persistent access to OS functionalities such as those listed in Section 7. Minor technical asymmetries in access to these features may materially affect user experience and competitive outcomes.

(80) There is an inherent information asymmetry between Alphabet and third-party developers when it comes to understanding the intricacies of Google Android. Alphabet not only controls the design choices, internal dependencies, and evolution of Google Android features but is also at the same time the competitor of some of its customers, be it OEMs or developers of services, such as AI services. This dual role of Alphabet makes the relationship with third parties unbalanced.

(81) The Google Android ecosystem also exhibits differences due to features and APIs spread across multiple versions of Google Android, device manufacturers and support libraries (see Section 5.1). Alphabet even provides a dedicated suite of libraries (i.e. Jetpack)[107] to reduce differences in the accessible APIs. Due to such differences in the Google Android ecosystem, third-party AI services can face restrictions or inconsistent behaviour because of different access to system-level

---

102 Agreed minutes of the meeting with [Trade body] of 12 January 2026, Doc ID [...], paragraph 7.

103 Recital 31 DMA provides that "*[t]he obligations in this Regulation should not only ensure contestability and fairness with respect to core platform services listed in the designation decision, but also with respect to other digital products and services into which gatekeepers leverage their gateway position, which are often provided together with, or in support of, the core platform services*."

104 [Alphabet internal document], Doc ID […].

105 [Alphabet internal document], Doc ID […], page 5.

106 Application Programming Interfaces (APIs) are software interfaces that allow two or more pieces of software to communicate with each other. APIs are typically implemented in one piece of software to offer (or 'expose') capabilities to other software, including applications. An API generally comes together with an API specification, i.e. a document describing how to use the API. The term API is sometimes used to refer to its specification, rather than its actual implementation.

107 Jetpack is a suit of libraries, tools and architectural guidance provided by Google to help developers to build Android apps more easily and consistently. Developers can rely on Jetpack libraries instead of only on core Android SDK. See "Android Jetpack Dev Resources, Android Developers", https://developer.android.com/jetpack, Doc ID [...], accessed on 10 April 2026.

actions controlled by private and public APIs. These differences also cause a variation of services within Google Android, which can affect the implementation of Article 6(7) by Alphabet.

(82) AI and AI-based services, notably AI assistants and agentic AI, are new technologies that have experienced massive growth recently. The capacity of AI assistants and agentic AI to act autonomously, process vast volumes of data, and continuously adapt through user interaction creates novel operational challenges.

(83) All the above aspects make Alphabet's compliance with Article 6(7) DMA, as well as stakeholders' ability to assess that compliance and benefit from it, more complex. Therefore, specification proceedings are particularly suited to establish a compliant and transparent technical framework that will enable third-party AI-based services to effectively interoperate with Google Android and compete on equal terms with Alphabet's services, before Alphabet's position in AI-based services becomes further entrenched. Swift intervention is important because several relevant Android features are scheduled to be released by mid-2026. Setting out clear rules through specification is therefore essential to ensure consistent application of Article 6(7) DMA across the entire Google Android ecosystem and provide legal certainty for the gatekeeper and the relevant market participants.

(84) This is all the more the case in a situation where a gatekeeper has integrated its own AI assistant across multiple services – including email, calendar, and productivity applications – because the risk of favouring its own vertically integrated services and foreclosure of other participants is heightened. Specification is therefore required to ensure that interoperability obligations apply consistently across all relevant use cases and integration points and that the entire Android ecosystem benefits from competing AI services.

(85) Indeed, implementation of Google Android across mobile devices of several manufacturers, combined with frequent updates of this operating system, and the evolving nature of AI services and AI technology necessitates effective specification measures which are suitable to prevent the gradual weakening of compliance with interoperability obligations through incremental technical changes that result in a situation where providers of services and hardware do not enjoy equally effective access.

(86) Specification pursuant to Article 6(7) DMA is necessary to ensure that Alphabet's interoperability obligations are clear, enforceable, and effective, and that AI services can compete on a level playing field in relation to users of Google Android.

## 4. LEGAL FRAMEWORK PURSUANT TO REGULATION (EU) 2022/1925

### 4.1. Specification proceedings for Article 6(7) DMA

(87) Article 6(7) DMA is an obligation that is susceptible of being further specified pursuant to Article 8(2) DMA.

(88) Pursuant to Article 6(7) DMA, gatekeepers shall:

(a) Allow providers of services and providers of hardware, free of charge, effective interoperability with, and access for the purposes of interoperability to, the same hardware and software features accessed or controlled via the operating system or virtual assistant listed in the designation decision pursuant to Article 3(9) DMA as are available to services or hardware provided by the gatekeeper.

(b) Furthermore, allow third parties and alternative providers of services provided together with, or in support of, CPSs, free of charge, effective interoperability with, and access for the purposes of interoperability to, the same operating system, hardware or software features, regardless of whether those features are part of the operating system, as are available to, or used by, that gatekeeper when providing such services; and

(c) Not be prevented from taking strictly necessary and proportionate measures to ensure that interoperability does not compromise the integrity of the operating system, virtual assistant, hardware or software features provided by the gatekeeper, provided that such measures are duly justified by the gatekeeper.

(89) Article 2(29) DMA defines interoperability as the ability to exchange information and mutually use the information which has been exchanged through interfaces or other solutions, so that all elements of hardware or software work with other hardware and software and with users in all the ways in which they are intended to function.

(90) Pursuant to Article 8(1) DMA, the gatekeeper shall ensure and demonstrate compliance with the obligations laid down in Article 6(7) DMA. The measures implemented by the gatekeeper must be effective in achieving the objectives of the DMA and of Article 6(7). The DMA empowers the Commission to enforce gatekeepers' compliance with the obligations set out in that Regulation.

(91) While the obligation to achieve effective interoperability is clearly set out in Article 6(7) DMA, gatekeepers may need guidance as to how to implement the obligation in relation to specific features of their designated operating systems. Such guidance may include aspects such as the exact interoperability solutions in relation to specific features, aspects of their technical implementation, and the modalities of access to be provided to third parties.

(92) Pursuant to Article 8(2) DMA, the Commission may, on its own initiative or at the request of a gatekeeper, adopt an implementing act specifying measures that the gatekeeper should implement in order to effectively comply with Article 6(7) DMA ("specification decision"). The Commission may decide to adopt such implementing act following a dialogue with the gatekeeper concerned and after enabling third parties to make comments.[108]

(93) Pursuant to Article 8(2) DMA, this implementing act shall be adopted within six months from the opening of the proceedings pursuant to Article 20 DMA. Pursuant to Article 8(5) DMA, with a view to adopting such an implementing act, the Commission shall communicate its PFs to the gatekeeper concerned, including the measures it is considering taking or that it considers the gatekeeper concerned should take, within three months from the opening of the proceedings.

(94) Pursuant to Article 8(6) DMA, to effectively enable interested third parties to provide comments, the Commission shall, when communicating its PFs to the gatekeeper or as soon as possible thereafter, publish a non-confidential summary of the case and the measures that it is considering taking or that it considers the gatekeeper concerned should take. The Commission shall specify a reasonable timeframe within which such comments are to be provided.

(95) Pursuant to Article 8(7) DMA, in specifying the measures that the gatekeeper concerned is to implement in order to effectively comply with its obligations, the

108 Recital 65 DMA.

Commission shall ensure that the measures are effective in achieving the objectives of the DMA and the relevant obligations. In addition, the measures shall be proportionate to the specific circumstances of the gatekeeper and the relevant service.

(96) Pursuant to Article 26(1) DMA, the Commission shall take the necessary actions to monitor the effective implementation of and compliance with, inter alia, the obligations laid down in Article 6(7) DMA and the decisions taken pursuant to Article 8(2) DMA.

(97) Pursuant to Article 8(9) DMA, the Commission may, upon request or on its own initiative, decide to reopen specification proceedings where (a) there has been a material change in any of the facts on which the decision was based, (b) the decision was based on incomplete, incorrect or misleading information, or (c) the measures as specified in the decision are not effective.

(98) Pursuant to Article 8(4) DMA, specification proceedings are without prejudice to the powers of the Commission to adopt a decision establishing non-compliance by a gatekeeper with any of the obligations laid down in that Regulation pursuant to Article 29, a decision imposing fines pursuant to Article 30 or a decision imposing periodic penalty payments pursuant to Article 31.

(99) Article 8(2) DMA therefore confers on the Commission regulatory powers to ensure that gatekeepers effectively comply with the obligations in Articles 5, 6 and 7 DMA and achieve the objectives of this Regulation and the relevant obligation. The legislator recognised that gatekeepers may implement compliance solutions which are ineffective in practice, which is liable to delay effective compliance. There is also a risk that a gatekeeper implements successive ineffective solutions, leading to further delays. The legislator therefore included specification proceedings in the DMA as an instrument allowing the Commission to specify, *ex officio* or at the request of a gatekeeper, the measures that the gatekeeper must implement to effectively comply with the objectives of the Regulation and the relevant obligation. Given the technical nature of the obligations, these measures may include technical details of compliance solutions taking account of the specific circumstances of the gatekeeper and the relevant service. This is to allow gatekeepers to integrate those measures into the technological design of their services. The existence of a public consultation underlines the regulatory character of specification proceedings and contributes to achieving the legal certainty sought by the legislator. Specification proceedings are non-adversarial: they do not entail a finding of non-compliance nor can they lead to the imposition of a fine.

### 4.2. Features and functionalities

(100) Article 6(7) DMA refers to hardware and software features accessed or controlled via the operating system. The scope of specific features must be assessed in light of the objective of this legal provision which, according to recitals 55, 56 and 57, is to ensure that alternative service and hardware providers have access to the same features under equal conditions and as effectively as the gatekeeper's own services or hardware.

(101) Hardware and software features within the meaning of Article 6(7) DMA include capabilities that the operating system makes available to the gatekeeper's services or hardware. Such capabilities include technical functionalities of the device on which the operating system runs, such as for instance near-field communication technology,

secure elements and processors, authentication mechanisms and the software used to operate those technologies.[109]

(102) A feature may consist of one or more functionalities. Access to a feature may require a functionality alone or in combination with other functionalities. Therefore, denying or undermining access to a functionality may result in denying or undermining access to a feature.[110] For a feature consisting of several functionalities, effective interoperability with that feature requires effective interoperability with all those functionalities. Withholding access to a functionality would allow gatekeepers to withhold access to a feature and would thus not amount to effective interoperability with that same feature – which would be contrary to the objective of Article 6(7) DMA to create a level playing field.

(103) A functionality of a feature may be a feature in its own right, meaning that operating system features can be visualised as forming a 'stack' of functionalities sitting at different levels, where the features at each level (or 'layer') are built by making use of features located below, which act as 'building blocks'.[111] In certain cases, interoperability may be needed both at a high level, where access is provided to user-facing and feature-rich operating system functionalities, and at lower levels, where access is provided to the functionalities implemented in the underlying operating system components. This is because interoperability at different levels can enable very different types of applications and business models, and can therefore be critical for contestability, fairness, and innovation.

(104) For instance, an operating system may offer a media casting feature (to cast the content of the device screen onto an external device such as a TV). Interoperability with that feature allows third parties to rapidly build on top of existing features, quickly bringing to market services that rely on the same level of access as the gatekeeper's services, but reliant on the gatekeeper's implementation choices for the advanced feature. Interoperability with the features used by the media casting feature allows third parties to recombine the 'building blocks' to use them to offer media casting or other apps. From the perspective of third parties, both sets of features provide building blocks to build their services using operating system functionality.

(105) Many apps use features from both levels and operating systems make features at different layers interoperable. While lower-level interoperability is likely to be more elaborate and thus slower in implementation, it offers the possibility for third parties to use software functionalities that are used by or available to the gatekeeper's services, in a way that enables a third party to recombine those functionalities in a novel way to offer services that the gatekeeper does not offer or offers in a different way.[112] This creates new opportunities and incentives for innovation. Depending on the situation, several related features (even those at "different layers") may need to

---

109 Recital 56 DMA.

110 For example, the notifications feature that allows smartwatches to display and interact with notifications received on the smartphone includes at least two functionalities: a functionality to read notifications and a functionality to reply to them. Undermining access to any of them would undermine access to the notifications feature as a whole.

111 This is similar to how intermediate products in a manufacturing supply chain are produced from other intermediate products in earlier steps of the manufacturing process, but also serve as inputs for subsequent steps. For operating systems, these layered components are collectively called 'software stack'. Third-party applications are built using components at different levels of the software stack.

112 See also Section 4.4.

be made interoperable to meet the objectives of the DMA[113] and ensure effective interoperability in individual situations.

(106) Alphabet argues that Article 6(7) DMA only covers features which are available to services outside the operating system, i.e. interfaces exposed to apps.[114] Specifically, Alphabet argues that operating system features which are only used by an operating system component are not covered by interoperability obligations, because this would lead to every layer of the operating system becoming accessible to third parties. According to Alphabet, "intra-operating system" competition was not the policy intent, and Article 6(7) does not aim to allow third parties to supply alternative operating system components such as the kernel, CPU drivers or file systems, as that could break upper-level interfaces.[115] Similarly, Alphabet notes that certain very low-level functions are not designed to be generally available interfaces (e.g. Android sandboxes apps from accessing each other's resources for security purposes).[116]

(107) Alphabet's arguments are inconsistent with the text and aim of Article 6(7) DMA and the scope of modern operating systems. The wording of Article 6(7) does not limit its scope to features "used by" the gatekeeper's software applications, since it uses the broader concept of "service", and also refers to features which are "available to" the gatekeeper's service.[117] Article 6(7) seeks to enable third parties to contest the gatekeeper on the digital platform, including with services the gatekeeper does not yet offer, and enable third parties to innovate on the platform.[118] Operating system features at different "layers" can be highly relevant to enable such contestability and innovation. For instance, modern operating systems typically offer a range of base and utility services which users value and expect. In many cases, third parties offer similar services with additional functionality and require interoperability with the underlying operating system features. Similar considerations apply if a third party seeks to offer functionality not offered by a gatekeeper's software application, or not at all offered by the gatekeeper. A gatekeeper can use operating system functionality at different layers as building blocks to offer attractive services based on its own decisions so that end users can benefit from innovation on the platform. Alphabet's argument that access to related features (or at different 'layers') is not required is contradicted by reality because operating systems already grant app-level access at different layers enabling developers to build feature-rich apps.[119]

(108) Alphabet's specific concerns on requiring access at several layers are only relevant to two of the features in this Decision, concerning on-device models. For both features Alphabet already enables interoperability to a significant extent at different layers of the operating system as further discussed in Sections 11.2 and 11.3. Alphabet's other examples are theoretical and unfounded. Article 6(7) does not require gatekeepers to allow third parties to replace the operating system or parts thereof. Alphabet is therefore not required to allow third parties to replace the kernel, CPU drivers, file

---

113 Recitals 2, 3, 32 and 33 DMA.

114 Alphabet's response to the PFs, Doc ID [...], paragraphs 34-35.

115 Alphabet's response to the PFs, Doc ID [...], paragraph 35, first bullet.

116 Alphabet's response to the PFs, Doc ID [...], paragraphs 34-35.

117 If an operating system feature is directly used by an operating system component, which is then directly used by a gatekeeper's service, then it means that the gatekeeper's service (indirectly) uses the operating system feature, and the operating system feature is available to that gatekeeper's service. See also Section 4.4.

118 See Section 4.4.

119 Examples for Google Android are on-device models (see Sections 11.2 and 11.3 below) and the location APIs (Google Android exposes location APIs through Google Play Services and lower level location APIs as part of AOSP – both are accessible to all developers as public APIs).

systems or change its sandboxing design. Nor is Alphabet's concern about *"having [the operating system's] internal layered architecture peeled back for access at every level"* justified. Access to such a low layer may not be required if the access granted at higher layers is sufficient to provide effective interoperability.

(109) A level playing field can exist only if third-party providers of services and hardware obtain effective access to feature updates, including new functionalities, at the same time as the gatekeeper and under equal conditions. This includes updated and new functionalities of features which may have already been made interoperable. This is in line with recital 65 DMA, according to which gatekeepers should ensure compliance with that Regulation by design. Therefore, the necessary measures should be integrated as much as possible into the gatekeepers' design of the operating system and of the features, including with respect to updates and new functionalities. If such updates are not made effectively accessible and interoperable, the effective interoperability that was previously granted will regress, or worse, can be easily circumvented. In practice, this means that third parties also need to be able to test any planned updates or new functionalities and obtain access to them once they are available to the gatekeeper's own services or hardware.

### 4.3. Interoperability with the same features as available to the gatekeeper and under equal conditions

(110) Article 6(7) DMA requires access to the same features as are available to the gatekeeper. The language of Article 6(7) DMA reflects the legislator's aim to ensure legal certainty without the need for lengthy discussions about equivalent features and to facilitate effective and timely *ex ante* compliance so that business users and end users can benefit from the DMA without delay. The legislator achieved this goal by requiring interoperability with the same features in a way that is equally effective[120] as the solution available to the gatekeeper, whatever the feature at hand and independently of the specific situation of any potentially interested or affected third party.

(111) Pursuant to Article 6(7) DMA, effective interoperability and access for the purposes of interoperability should be granted with the "*same*" features as are available to or used by the gatekeeper's services or hardware (the "what" of the interoperability). Hence, providing access to features that are only similar and not the same is insufficient.

(112) In addition, recitals 55 and 57 DMA clarify that the actual interoperability solution – the "how" of the interoperability – needs to be "*equally effective*" to the solution available to the gatekeeper and should be made available "*under equal conditions*". This does not mean that the interoperability solution must be exactly the same. By mandating an equally effective interoperability solution the legislator acknowledged that the implementation of interoperability does not always need to (and potentially cannot always) be the same for the gatekeeper and third parties, but interoperability must be granted to the same features under equal conditions.

(113) Equal effectiveness and equality of conditions are assessed by comparison to how the gatekeeper implements interoperability and access to hardware and software features for its own services and hardware. It includes both user-facing and non-user-facing aspects, such as the end user journey, ease of use, device and software set-up, data transmission speed, and energy consumption. Indeed, these may be critical properties of the feature itself.

---

120 Recital 55 DMA.

(114) Providing interoperability with the same feature under equal conditions creates a level playing field, allowing third parties to offer their services and hardware and innovate on an equal footing with the gatekeeper. Access to a partial, degraded or barren feature would not create the level playing field required under Article 6(7) DMA and would undermine the goals of contestability and fairness of the DMA.

### 4.4. Features available to or used by the gatekeeper

(115) Pursuant to Article 6(7) DMA, interoperability should be provided with the same features "*as are available to, or used by*" the gatekeeper's services and hardware.

(116) The interoperability obligation encompassed in Article 6(7) DMA is not limited to features that are already used by existing services or hardware of the gatekeeper, but also encompasses features that are available to the gatekeeper's services, irrespective of their current use by the gatekeeper. If there was a requirement for the gatekeeper to first develop its own use case based on an operating system feature that is available to the gatekeeper for that feature to fall within the obligation of Article 6(7), this could significantly undermine innovation by alternative providers and create a risk of leveraging, market tipping and the gatekeeper further entrenching its position. The gatekeeper would always enjoy a first-mover advantage and would be able to decide if and when new services or hardware could enter the market and thus restrict innovation to its own level and the moment in time when it decides to use the available operating system features to offer a specific service or hardware – if ever. Pursuant to the DMA, third parties must be allowed interoperability to provide novel and innovative products and services, irrespective of whether they are part of the gatekeeper's offer of products and services, as to be able to "*overcome barriers to entry*" and "*challenge the gatekeeper on the merits of their products and services*".[121]

(117) The concept of a feature being "available to" is distinct from the concept of a feature being "used by". This follows from the wording of Article 6(7) DMA and is consistent with the objective to enable innovation from third parties irrespective of whether the gatekeeper uses the feature for its own purposes as long as the feature exists.

(118) A feature is "used by" a gatekeeper if a service or hardware of the gatekeeper makes use of it. The type of use depends on the feature. For instance, if a feature is exposed by the operating system via an API, a service calling into the API uses the feature. In the case of a feature consisting in device access points, such as hardware buttons, a service that is assigned to such an access point, for example a digital wallet or AI assistant, uses the feature. For a feature consisting in a service being displayed on operating system surfaces, for instance user data or information being displayed on the lock screen, the service being displayed makes use of that feature.

(119) It is not required that a service of the gatekeeper uses a feature on all devices running a specific designated operating system for the feature to be used by that service. For instance, a service is "using" a feature even when it does so in the devices last released on the market but not in previously released devices comprising the existing installed base. Similarly, some features, such as synchronisation of notifications from the smartphone to a smartwatch, may be used only when the user owns both the phone and the smartwatch.

(120) A feature is "available to" a gatekeeper's service or hardware if the gatekeeper can determine fundamental factors such as the feature's inclusion in the operating system, its design and implementation, as well as the conditions and requirements

[121] Recital 32 DMA.

governing access to and use of the feature, to the advantage of its services. The gatekeeper's decision-making power with regards to such fundamental factors allows it to determine whether and how to make features available (to itself and others). The gatekeeper thereby decides on the feature's release, accessibility and operation, also after a device's release.

(121) A feature of an operating system can be "available to" a gatekeeper when the gatekeeper, which controls the basic conditions governing access and use of the feature, sets these conditions at the operating system level in a way that restricts access to existing features to services meeting specific conditions. By setting such conditions, Alphabet decides which services have access to a feature and can make a feature accessible to its services. It is therefore entirely within the gatekeeper's control to decide, at the operating system level, which services get access to existing features. Such cases are covered by the obligation of Article 6(7) DMA. This is in line with the objective to enable third-party innovation on devices running a designated operating system irrespective of whether the gatekeeper's services or hardware use all existing features of the operating system.

(122) Alphabet argues that the Commission's approach to the features that fall within the scope of Article 6(7) DMA is expansive because it relies on an untenable interpretation of what is "available to" Alphabet. *First,* Alphabet claims that the interoperability obligation is triggered only if a feature exists and does not require the creation of new features.[122] *Second,* Alphabet argues that Article 6(7) DMA covers features that "are available" and not features that "could be available if technical alterations or policy changes were made" because the intention of the language was to cover features that are "active on devices" even if a Google service is not yet using them. *Third,* Alphabet claims that the Commission's reasoning on what features are available to Alphabet leads to the entirety of Google Android and even features that are not in the operating system but that Google could decide to include in the operating system to be subject to Article 6(7) DMA. As a result, parts of the statute are redundant because the "qualifying effect" of the term "available to" is removed.[123]

(123) Alphabet's arguments are misplaced and theoretical for the following reasons. *First,* this Decision does not cover "the entirety of the Android OS", but 11 specific features which exist on Google Android mobile devices in the European Union. To the Commission's knowledge, implementations of 10 of these features are even used by Alphabet's software applications in the European Union. The sole exception, Computer Control, an implementation of the screen automation feature, is currently only used outside the European Union,[124] but it is available via an API on devices in the European Union and Alphabet's software applications could use it without any policy change or technical alteration in Android. A discussion of the meaning of "available to" in all conceivable scenarios is therefore theoretical. It is unclear what Alphabet means by a feature being "active" on devices and this concept does not exist in Article 6(7). *Second,* Alphabet does not contest that it has, as the operating system provider, full control over the conditions governing access to a feature that exists on the device and is controlled by the operating system, and that Alphabet could set conditions granting its own services access (see Section 5.2.1 for details). This control over features existing on the device makes a feature available to Alphabet's services and is consistent with the legislator's aim to enable innovation

122 Alphabet's response to the PFs, Doc ID [...], paragraphs 29 to 31.
123 Alphabet's response to the PFs, Doc ID [...], paragraphs 32 and 33.
124 See recital (989).

from third parties irrespective of whether a gatekeeper service uses the feature itself. *Third,* this Decision does not require Alphabet to create new features.[125] *Fourth,* as explained particularly in Sections 4.2 and 4.8, the scope and application of Article 6(7) are limited in important aspects. This Decision addresses Alphabet's substantiated concerns in that respect insofar as they are relevant to this Decision, e.g., in Sections 6.3 and 11.4.8.2).

### 4.5. Interoperability must be effective

(124) Interoperability pursuant to Article 6(7) DMA needs to be effective, meaning that it must enable the desired result to be achieved in practice. As such, the interoperability solution must be granted in a manner that is technically sound, stable, and workable in practice for third parties without unnecessary hurdles, be they on the side of third-party providers or on the side of end users.

#### *4.5.1. Implementation of interoperability*

(125) What an effective interoperability solution requires depends on the gatekeeper's choices regarding the design and use of its operating system. In the case of some features, a mere lifting of a contractual or technical restriction might be sufficient. In other cases, the gatekeeper might need to create and/or implement the prerequisites – including software components – that are required to provide effective interoperability. An effective interoperability solution may therefore require additional engineering work by the gatekeeper to make a feature interoperable.

(126) This is in line with the wording and aim of Article 6(7) DMA, which require effective interoperability with the same features under equal conditions, but not necessarily with the same software implementation (see Section 4.3 above). Interoperability is a results-based concept, as is apparent from its definition in Article 2(29) DMA which includes: "*all elements of hardware or software work with other hardware and software and with users in all the ways in which they are intended to function*". The definition also provides that information shall be exchanged "*through interfaces or other solutions*".

(127) Recital 57 DMA states that the gatekeepers should be "*required to ensure [...] effective interoperability*" and should "*allow competing third parties to interconnect through interfaces or similar solutions to the respective features*".[126] The DMA therefore makes it clear that a gatekeeper may have to undertake certain engineering work to make an effective interoperability solution available to third parties.[127]

(128) The gatekeeper must enable third parties to interconnect smoothly to the relevant features through interfaces or similar solutions. For this purpose, the gatekeeper may have to technically develop and implement such interfaces (e.g. APIs or other solutions), to ensure that the features subject to Article 6(7) DMA can work with the hardware and software of third parties in all the ways in which they are intended to function. This means that for an interoperability solution to be effective, the solution

---

125 Alphabet's arguments on the user-customisable hotword and the centralised access to apps' data stored on-device features are further addressed in the corresponding sections below.

126 The verb "*allow*" does not intrinsically refer only to a passive action but defines an endpoint of "*make it possible for something to be done or to happen*", in this case effective interoperability with the same features as available to the gatekeeper. See Cambridge Dictionary, https://dictionary.cambridge.org/dictionary/english/allow, Doc ID [...], accessed on 9 April 2026.

127 This interpretation is also in line with the judgment *Alphabet v AGCM*, where the Court of Justice held that granting interoperability can entail the adoption of additional work, such as the development of a template to implement a requested interoperability for certain third-party apps. See judgment of 25 February 2025, Alphabet v AGCM, C-233/23, ECLI:EU:C:2025:110, paragraphs 73-74.

that a gatekeeper must make available to third parties may in some cases go beyond merely opening an interface to them. This might require, for instance, the use of technical standards, the provision of technical documentation or assistance, or the provision of software development kits ("SDKs"), which are commonly used to achieve interoperability.[128]

(129) In addition, for interoperability to be effective, gatekeepers cannot circumvent it through practices that may render an interoperability solution ineffective, such as imposing conditions that unduly differentiate between third parties; imposing conditions that are not equal to those that apply to the gatekeeper's own services and hardware; failing to properly consider the needs of third parties that will make use of the solution (e.g. by implementing limitations that prevent certain use cases); providing implementations that are not properly tested for bugs or other shortcomings (e.g. design gaps, performance and stability issues); providing implementations that are less stable or consistent over time than the feature used by or available to the gatekeeper; failing to provide adequate and up-to-date documentation; or failing to provide adequate and timely assistance to third parties that report issues (e.g. by submitting bug reports).

(130) For an interoperability solution to be effective, the solution must take account of how third parties are making use of the solution, how third parties make their services available to their end users, and the possible involvement of other third parties. The interoperability solution should not impose undue costs – including development costs – for third parties directly benefitting from the interoperability solution or for other third parties that are involved in the use of the relevant feature.[129] If an interoperability solution for some third parties (the beneficiaries) were to require other third parties to adapt their products or services (e.g. their apps) to make the solution workable in practice[130] this would de facto shift the burden of compliance from the gatekeeper to those third parties, in breach of the obligation for the gatekeeper to ensure effective interoperability under equal conditions.

*4.5.2. User journey and friction*

(131) On the side of end users, the effectiveness of interoperability solutions must not be undermined by introducing unnecessary "friction" when an end user uses third-party services or hardware. Friction refers to any obstacle, difficulty, or inefficiency that hinders or affects the end user's ability to complete a task or achieve their goal in the shortest possible time and with the least effort. For instance, friction has an impact on the ease, convenience and speed of using services or hardware from the end user perspective. Friction is unnecessary if it is imposed by the operating system only on end users of third-party services and hardware, but not on end users of the gatekeeper's services and hardware. Gatekeepers are therefore required not to add,

---

128 For instance, Alphabet makes available the Android SDK, which app developers use for software development for Google Android devices. See e.g. https://developer.android.com/tools: "*The Android SDK is composed of multiple packages that are required for app development*", Doc ID [...], accessed on 23 April 2026.

129 These third parties may not necessarily be business users of the third parties that benefit from the interoperability solution. For example, when a notification from a messaging app is displayed on a connected device, this is made possible by the technical implementation of the operating system, which acts as an intermediary. It is not required – and normally it is not the case – that the app developer and the connected device provider are in a professional relationship, or even that they are aware of each other's product.

130 For instance, continuing with the example above, if every messaging app provider was required to develop custom support to display notifications on every third-party connected device.

directly or indirectly, friction with regards to third parties that does not exist for their own services and hardware.

(132) This follows directly from "*effectiveness*" and "*equal conditions*". Indeed, it is well known that even a small amount of friction can have an outsized impact on end user behaviour and ultimately on commercial success.[131] A user experience without unnecessary friction is essential to enable a level playing field between third-party and gatekeeper's services and hardware. End users expect a frictionless experience. Unnecessary friction undermines effective interoperability, as end users may not enjoy the full functionality of a third-party service or hardware, in turn reducing the commercial attractiveness of those services or hardware.[132] The avoidance of friction and the seamlessness of a user journey is critical to improving the user experience, i.e. "*the overall experience users have […], which includes the perceived utility, ease of use and efficiency of interacting with it*".[133]

(133) Friction is introduced by measures that make the "user journey" – i.e. "*the series of actions or steps for users to perform in order to reach their goal*"[134] – slower or more complicated and frustrating than necessary.[135] Friction can be caused by any behavioural techniques or interface design elements that force an end user to take several, potentially confusing actions before the end user can use a service or complete a task. Examples of such techniques are successive and excessive permission prompts, or unnecessary requirements to switch apps to complete an operation.

(134) Friction can also be caused by offering choices to the end users in a non-neutral manner that steers the end users towards making certain choices or by limiting the ability to exercise their choice effectively and easily. Examples of such techniques are misrepresenting or exaggerating the risks of using a service or hardware, communicating risks based on the mere theoretical possibility that such risks might materialise, or informing users of risks only in certain contexts without informing them in other contexts where the risk is similar or comparable. Another example is the practice of disabling by default a permission that is necessary for the proper and full functioning of the service or hardware and requiring the end user to actively search for the option in settings in order to change it.

---

131 For example, Alphabet explains that when a website "*reduce[d] load times from nine seconds to 1.4 seconds, ad revenue increased three percent, and page views-per-session went up 17 percent*" and that "*[w]hen you speed up service, people become more engaged - and when people become more engaged, they click and buy more*". See The Google Gospel of Speed; https://www.thinkwithgoogle.com/_qs/documents/77/the-google-gospel-of-speed-urs-hoelzle_articles.pdf, Doc ID [...], accessed on 23 April 2026.

132 For instance, with respect to AI services, friction makes it more likely that an end user will abandon a user journey that is necessary to use some functionalities of AI services, e.g. setting up, setting as default or using the service on their devices.

133 See "Guidelines 03/2022 on Deceptive design patterns in social media platform interfaces: how to recognise and avoid them," European Data Protection Board, version 2.0, 2023, https://www.edpb.europa.eu/system/files/2023-02/edpb_03-2022_guidelines_on_deceptive_design_patterns_in_social_media_platform_interfaces_v2_en_0.pdf, Doc ID [...], accessed on 9 April 2026.

134 See "Guidelines 03/2022 on Deceptive design patterns in social media platform interfaces: how to recognise and avoid them," European Data Protection Board, version 2.0, 2023, https://www.edpb.europa.eu/system/files/2023-02/edpb_03-2022_guidelines_on_deceptive_design_patterns_in_social_media_platform_interfaces_v2_en_0.pdf, Doc ID [...], accessed on 9 April 2026.

135 Digital services providers often employ teams and tools to study user behaviour and minimise or, in some cases, maximise friction: see https://www.thealien.design/insights/ux-metric, Doc ID [...], accessed on 9 April 2026.

(135) The DMA aims to address the harmful impact of friction and its detrimental effect on contestability and fairness. Recital 37 DMA states that "*gatekeepers should not design, organise or operate their online interfaces in a way that deceives, manipulates or otherwise materially distorts or impairs the ability of end users to freely give consent*". Further, it states that "*not giving consent should not be more difficult than giving consent*" and that the gatekeeper "*should proactively present a user-friendly solution to the end user to provide, modify or withdraw consent in an explicit, clear and straightforward manner*". Articles 13(4) and 13(6) DMA address the issue that friction undermines effective compliance by prohibiting behavioural techniques or interface design that make the exercise of the rights or choices of end users unduly difficult, including by offering choices to the end user in a non-neutral manner, or by subverting end users' or business users' autonomy, decision-making, or free choice via the structure, design, function or manner of operation of a user interface or a part thereof.

(136) Equally effective interoperability requires that end users can set preferences (i.e. settings) in relation to third-party services and hardware as they can for the gatekeeper's services and hardware. To allow the end user to configure a specific setting, there are generally two options for the location of these settings: to include the setting in system-level settings, or to have it inside the app provided by the third party (or both). If the end user is required to switch between a third-party app and the system-level settings in order to change relevant settings, while not being required to do so in respect of the gatekeeper's services or hardware, this may add significant friction compared to the use of the gatekeeper's services or hardware, in particular if not clearly directed to the relevant setting.[136] Therefore, when the gatekeeper designs the interoperability solution, it should ensure that the location and design of the relevant settings: (i) do not make the user experience more burdensome for end users of third-party services and hardware; and (ii) are sufficiently flexible to meet reasonable differentiation needs of different third parties.

*4.5.3. Use of interoperability solutions in the gatekeeper's first-party apps*

(137) Some operating system features are "multi-sided", i.e. designed to allow integration between different apps of potentially different developers. For such features, the effectiveness of interoperability for a developer crucially relies on other developers using the other "side" of the feature.

(138) For example, in the context of connected devices, interoperability with the notification feature of mobile operating systems implies that third-party smartwatches can display the notifications generated by a mobile device (such as a smartphone), in the same way as smartwatches provided by the gatekeeper. Here, the "donating" side of the feature allows apps (e.g. messaging apps) to generate notifications, while the "receiving" side of the feature allows smartwatches to access and display those notifications. In this case, the obligation of equal effectiveness of the interoperability solution requires that all notifications from all donating apps are supported, including from apps of the gatekeeper.[137] If the gatekeeper was allowed to

136 It is often difficult for the end user to locate and change the setting in system-level settings, if the relevant setting is not highlighted, and the end user is not guided back to the third-party app.

137 This is required by the first sentence of Article 6(7), and in particular by the effectiveness requirement. The same requirement would also follow from the second sentence of Article 6(7), which requires interoperability with the features of services provided together with, or in support of, the operating system – regardless of whether those features are part of the operating system. In the case of notifications, the relevant features under the second sentence of Article 6(7) would be the features of the first-party apps that allow the gatekeeper's smartwatch to access the notifications generated by those

reserve notifications related to its own first-party apps (e.g. web browser app, messaging app, map and navigation app, smart home app, health app, etc.) for its own smartwatches while withholding those notifications from the interoperability solution available to third-party smartwatches, the solution would not be equally effective – and would therefore not meet the requirements of Article 6(7).

(139) Similarly, in the context of agentic AI, third-party AI assistants must be able to interact with other apps, as effectively as the gatekeeper's AI assistant. In this case too, equal effectiveness requires that the gatekeeper does not prevent third-party assistants from interacting with its own first-party apps, while allowing its own first-party assistant to do so. For example, with respect to the screen automation feature, the gatekeeper should not prevent third-party AI assistants from automating first-party apps, while allowing its own AI assistant to do so. This applies regardless of whether the gatekeeper's apps and third-party apps use a different interoperability solution: as explained above in Section 4.3, while the implementations may be different, the result must be equally effective. In particular, the gatekeeper may not develop a "parallel" implementation of the same multi-sided operating system feature that is reserved only to first-party apps, as this would seriously undermine the effectiveness of the interoperability solution made available to third parties. This is especially important since many first-party apps often have high adoption among end users, also due to the inherent advantage stemming from pre-installation of many of these apps. Otherwise, end users will be much more likely to revert to the gatekeeper's services, not on their merits but because they are the only ones that integrate with the other gatekeeper services that end users are likely to be using (and want to continue using).[138]

(140) For this reason, the measures for some of the features[139] in scope of this Decision include requirements that apply to Alphabet's first-party apps and that ensure that Alphabet cannot use or develop a parallel implementation of the feature that reserves integration only to its own apps – as this would be against the text and the objectives of Article 6(7), thereby defeating the intended effect of this obligation.

(141) Alphabet contested these measures, claiming that they mischaracterise Article 6(7) as a data-sharing obligation and wrongly treat data as a feature.[140] These claims are misplaced as they misconstrue both the scope and the nature of the measures. *First,* none of the measures in this Decision impose a broad obligation for Alphabet's first-party apps to share any data with third-party apps. For example, if Alphabet simply

---

first-party apps. In any event, the legal basis for the measures in this Decision that relate to first-party apps is the first sentence of Article 6(7) for all features. This is without prejudice to the fact that the same or equivalent measures might (also) be required under the second sentence.

138 By contrast, third parties are generally not in a position to gain an advantage by following the same behaviour, because: (i) they often do not have as many "donor" apps that are as popular as the gatekeeper's; (ii) they typically do not have a comparable ecosystem of "receiver" apps and products (iii) they generally do not control the operating system, meaning that any "parallel" implementation of an operating system feature would not be equally integrated with the rest of the ecosystem or, in many cases, technically feasible at all. Therefore, third parties are more likely to act as "donors" (e.g. by donating their data) in order to participate in the gatekeeper's ecosystem. For example, a third party providing both a messaging app and a smartwatch would not have sufficient incentives to withhold notifications from the messaging app and display them only on its own smartwatch, because that would mean that the messaging app would be less functional for the many users who use other smartwatches, including the gatekeeper's smartwatches.

139 These are: Long-press Home/Long-press Navigation Handle contextual invocation feature, centralised access to apps' data stored on-device, context-aware intelligence, access to ambient data, structured on-device integration, screen automation.

140 Alphabet's response to the PFs, Doc ID [...], paragraphs 73-84.

stores data in an app, none of the measures require that app to share any data. The measures are narrowly scoped to each specific feature and tailored to simply avoid blatant attempts to defeat the purpose of interoperability such as the ones described above, and are justified in the respective feature Sections. *Second,* Alphabet's claim that data is not a feature[141] again misconstrues the measures as a broad data sharing obligation. As acknowledged by Alphabet,[142] the DMA defines interoperability precisely in terms of the "*ability to exchange information and mutually use the information*", where "information" means data.[143] While data is indeed not a feature in itself, interoperability with a feature may well require access to some data and such data may be indispensable for the feature to be made interoperable. For example, interoperability with the notification feature described above obviously requires the third-party smartwatch to access the content of the notifications, which is data. If the gatekeeper was allowed to withhold that data from the interoperability solution – while at the same time making it available to its own services via the same or a "parallel" implementation – any interoperability solution to display notifications on third-party smartwatches would not be effective.

### 4.6. The interoperability obligation applies to designated gatekeepers, irrespective of the implementation of the operating system on their own or on third-party devices

(142) Where an undertaking is designated as a gatekeeper in relation to a CPS, it is that undertaking's responsibility, pursuant to Article 8(1) DMA, to ensure and demonstrate effective compliance with its obligations.

(143) It is clear from the wording and the objective of Article 6(7) DMA that allowing actual access and actual interoperability is incumbent on designated gatekeepers.

(144) The text of that provision requires "effective interoperability with, and access for the purposes of interoperability to" certain features. As explained above (recital (126)), the definition of interoperability in Article 2(29) DMA is results-based, requiring that "*all elements of hardware or software work with other hardware and software and with users in all the ways in which they are intended to function*". The gatekeeper must therefore ensure actual technical access, not an "opportunity" to obtain such access under certain conditions.

(145) In other words, the mere hypothetical opportunity to obtain access, e.g., via negotiations, or access under certain conditions imposed directly or indirectly by the gatekeeper, e.g., the pre-installation of an app, fall short of the gatekeeper's responsibility to ensure effective compliance with its obligation under Article 6(7) DMA. Opportunities may, by definition, not materialise and conditions, especially if dependent on third parties' intervention, add complexity and impose additional hurdles, which are liable to limit the quality of access and effective interoperability. The legislator sought to ensure straightforward access to and effective interoperability with operating system features to enable, amongst other, clarity and legal certainty for business users as to the scope of the available features and the modalities of access to these features as well as means of effective interoperability.[144]

---

141 Alphabet's response to the PFs, Doc ID [...], paragraphs 76-79.

142 Alphabet's response to the PFs, Doc ID [...], paragraph 80.

143 Article 2(29) DMA.

144 Article 6(7) DMA therefore reflects the legislative decision to ensure interoperability and access on a level playing field instead of a system whereby app providers have the "opportunity" to obtain access (see Alphabet's submission on 1P/3P of 17 March 2026, Doc ID [...], page 5, paragraphs 18-20).

(146) Article 6(7) DMA therefore requires gatekeepers to unconditionally ensure effective interoperability (subject only to strictly necessary and proportionate integrity measures) and does not allow them to subject effective interoperability to conditions that are contrary to the wording or the spirit of Article 6(7) DMA. Otherwise, the gatekeeper could render interoperability *de facto* ineffective by imposing conditions, such as, for instance, allowing a third party to charge for the effective interoperability that the gatekeeper is required to ensure free of charge.[145]

(147) The DMA does not prevent different implementations depending on the business model operated by the designated gatekeeper. Article 6(7) DMA applies equally to "closed" and to "open" ecosystems. While Article 6(7) DMA mandates effective interoperability, it does not preclude the pre-installation of apps or default setting agreements, which are means to make end users aware of those apps. However, pre-installation cannot be a pre-condition for effective interoperability or for access to features pursuant to Article 6(7) DMA.[146] Indeed, when the gatekeeper makes access to features conditional on concluding pre-installation agreements with OEMs, this limits interoperability for business users as it impedes them from effectively reaching end users. It also limits end user choice. If interoperability is granted only to pre-installed apps, an end user's preferred app (or several apps from a certain app provider) may not be available on the device or be available only in a degraded form (e.g. with fewer features).[147]

(148) This applies regardless of whether the gatekeeper only uses the designated operating system on its own devices or also makes it available to licensed third parties for integration into their devices. Recitals 56 and 57 DMA recognise the dual role that gatekeepers *can* have as developers of operating systems and device manufacturers and state that gatekeepers that are manufacturers of devices should provide effective interoperability with the same operating system, hardware or software features that are available or used in the provision of their own complementary and supporting services. Operating system providers can have a dual role, and in practice typically do, as developers of the operating system and of software applications running on its operating system. It is clear, in light of these recitals, that the interoperability standard required pursuant to Article 6(7) DMA is the same irrespective of whether the gatekeeper has a dual role (i.e. provides the operating system and the device) or not.[148]

---

145 Compare with Alphabet's submission on 1P/3P of 17 March 2026, Doc ID [...], page 3, paragraphs 11-12.

146 See Alphabet's submission on 1P/3P of 17 March 2026, Doc ID [...], page 2, paragraphs 6-9, claiming that Article 6(7) DMA is not concerned with differences in interoperability between pre-installed and user-downloaded apps.

147 Allowing users to select the apps and app providers of their choosing is foundational to the DMA, which seeks to allow business users to offer their services to end users, see, e.g., recital 15. This is also relevant in the context of integrity measures as allowing users to decide about the appropriate level of risk can be a less restrictive measure. Such a condition preserves access, while a pre-installation requirement is liable to limit access (compare with Alphabet's submission on 1P/3P of 17 March 2026, Doc ID [...], pages 2 to 4, paragraphs 10-12).

148 The extent to which the gatekeeper allows the OEM to modify the designated operating system does not change the interoperability obligations of the gatekeeper. An operating system remains a designated CPS and an important gateway, regardless of whether some modifications are permitted, and the gatekeeper remains responsible under Article 6(7) DMA to ensure effective compliance. The Alphabet Designation Decision designates Alphabet as a gatekeeper for Google Android, and Alphabet notified Google Android across all mobile devices, both its own Pixel devices and third-party devices (cf. Alphabet's submission on OEM of 17 March 2026, Doc ID [...], pages 3 to 8, paragraphs 9-16), without any differentiation at any point between the two.

(149) Similarly, a gatekeeper cannot escape the obligations laid down in Article 6(7) DMA by delegating its responsibilities to third parties. A gatekeeper may licence third parties to use the designated operating system on their devices but it is the gatekeeper's responsibility to take all actions – including legal, contractual, judicial and technical actions – necessary to ensure that the implementation of the operating system on such device complies with Article 6(7) DMA.[149] Otherwise, a gatekeeper could use third-party device manufacturers to escape its own obligations pursuant to that Article or such third-party device manufacturers could frustrate the objectives of the DMA (e.g. by degrading, withholding, or charging for such interoperability or access) to further their own interests.

(150) In Alphabet's view, if a gatekeeper provides an open-source operating system which is also implemented by third-party OEMs, the gatekeeper can comply with Article 6(7) DMA by making the operating system available to OEMs and allowing OEMs to decide which services obtain feature access on OEM devices.[150] Alphabet argued that as the OEM is responsible to decide which services – Alphabet's, the OEMs', and third parties' services – can access the operating system features, there is access for all services under "equal conditions" and the operating system is therefore "interoperable by design".[151] According to Alphabet it is irrelevant whether an OEM ultimately grants access to the feature and on what terms: "*the mere opportunity to obtain access is sufficient*".[152] In Alphabet's view, it suffices that it "*takes the steps required to 'make it possible' for interoperability 'to be done or to happen'*" and what OEMs subsequently do when modifying or customising Android code has no bearing on Alphabet's compliance.[153] Alphabet makes this argument across the entire ecosystem of OEMs, that is, also including when it is itself an OEM for its Google Pixel devices – meaning that Alphabet considers that it is free, under Article 6(7), to reserve operating system features to its own services on Google Pixel devices.

(151) In support of its view, Alphabet submitted that the wording of Article 6(7) DMA requires to "allow" effective interoperability, not "ensure", "provide" or "guarantee",[154] and that it does not require that interoperability be unconditional but rather (as reflected in recital 57) it requires that any conditions imposed on third-party access to a feature be the same as those that are imposed on the gatekeeper's own services (subject to integrity and proportionality).[155] Furthermore, Alphabet submitted that nothing in the DMA prevents third parties who are not gatekeepers from frustrating its objectives through their independent actions. In particular, Alphabet claims that Android's fundamental design – which leaves discretion to OEMs in areas where they have legitimate interest as the manufacturer of the device – cannot amount to "circumvention". A different conclusion would disadvantage gatekeepers with an open-source model, as OEMs could create direct liability for

149 In particular, a gatekeeper's interoperability obligations do not end when third-party manufacturers modify the operating system and deploy it on their devices (see Alphabet's submission on OEM of 17 March 2026, Doc ID [...], page 3, paragraph 9). Such an interpretation is inconsistent with the nature of a "*core platform service*", the fact that the operating system CPS becomes a gateway only upon integration on the actual device, and the fact that Alphabet maintains Google Android-related requirements on third-party OEMs throughout the lifetime of the mobile device.

150 Alphabet's response to the PFs, Doc ID [...], paragraphs 4-7, 22-27, Section IV.

151 Alphabet's response to the PFs, Doc ID [...], paragraphs 7, 24-27, 123 et seq.

152 Alphabet's response to the PFs, Doc ID [...], paragraphs 3, 126.

153 Alphabet's response to the PFs, Doc ID [...], paragraph 27.

154 Alphabet's response to the PFs, Doc ID [...], paragraph 27.

155 Alphabet's response to the PFs, Doc ID [...], paragraphs 123, 124.

them in potentially limitless ways that could never fully be accounted for through measures like Alphabet's Compatibility Definition Document (CDD).[156]

(152) Alphabet's position is incompatible with Article 6(7) DMA. As explained above (recitals (142)-(145)), Article 6(7) DMA requires actual "interoperability" and "access", not the "mere possibility" of obtaining feature access based on the unilateral decision of the OEM.

(153) Alphabet is required to take all necessary steps to ensure compliance of its operating system on the device of third-party OEMs. It is incorrect that Alphabet must in all circumstances "guarantee" compliance with respect to such devices. The gatekeeper decides to whom and on what terms it distributes its operating system. Accordingly, Alphabet can set conditions requiring third-party OEMs to not take decisions with respect to the Google Android operating system which would render it non-compliant with Article 6(7) DMA. In these terms, Alphabet can also provide for verification mechanism, sanctioning mechanisms, etc. Liability under the DMA is limited to acts or omissions of the gatekeeper,[157] and fines may only be imposed if the gatekeeper failed to comply negligently or intentionally.[158] Third-party OEMs are well aware that Google Android is designated under the DMA and that Alphabet is therefore subject to specific obligations.[159]

(154) This does not require Alphabet to have "*an unlimited capability to control OEMs*",[160] but merely to make use of the means at its disposal. This interpretation is consistent with Alphabet's practice to tightly regulate third-party OEMs' use of Google Android today (as explained in Section 6.1.1)). It imposes numerous obligations on OEMs on how they should implement Google Android on their mobile devices. For instance, Alphabet has the right to deny approval of a third-party Google Android device – i.e., the third-party OEM is contractually prohibited from marketing such device – if this device does not implement detailed measures to ensuring compliance with certain regulatory obligations under the DMA. Alphabet also decides which features are only accessible if the OEM decides so, while it actually prevents OEMs from making such a decision for the vast majority of features, which OEMs are contractually obliged to expose to all apps, including downloaded apps (see Section 5.2.1).

(155) This interpretation is consistent with Alphabet's submissions in the context of the designation proceedings, where it argued that the fact that [quotes from Alphabet's designation submission relating to the implementation of Android OS on Android-compatible devices];[161] ;[162] .[163] Alphabet thereby recognised that it is Alphabet who

---

156 Alphabet's response to the PFs, Doc ID [...], paragraph 120. The CDD is part of the contractual requirements that Alphabet imposes on OEMs (see Section 5.1).

157 Contrast with Alphabet's response to the PFs, Doc ID [...], paragraph 120, fourth bullet.

158 Article 30(1) DMA.

159 The [OEM] quote in paragraph 120 1st bullet in full does not suggest that [OEM] would oppose terms limiting its ability to modify Google Android: "*Contrary to some perceptions, the term "OEM" is somewhat misleading when applied to* [OEM] *and other Android device manufacturers. We are not a company that simply follows Google's orders or instructions to manufacture Android-powered devices. Instead, we develop, manufacture and sell our own branded devices, leveraging Google's Android OS but reflecting our unique innovations and value*." See [OEM] submission in response to the Commission's public consultation, 13 May 2026.

160 Alphabet's response to the PFs, Doc ID [...], paragraph 119.

161 Final Form GD, 3 July 2023, paragraph 177.

162 Final Form GD, 3 July 2023, paragraph 134(b).

163 Final Form GD, 3 July 2023, paragraph 325.

provides the "*important gateway*"[164] (i.e. the Google Android CPS), not OEMs – and this is why the Commission designated Alphabet as the gatekeeper for Google Android, not OEMs. Alphabet itself therefore recognised its responsibility to comply with the obligations in the DMA, including Article 6(7) thereof, both with respect to its own Pixel and third-party OEM smart mobile devices, without expressing any reservations with respect to its obligations as to the latter.

(156) The assessment is, in principle, the same irrespective of whether a gatekeeper operates an open-source operating system.[165] Alphabet claims that [direct quote on liability considerations in the context of relationships between OEMs and Alphabet],[166] but any liability is a direct consequence of Alphabet's actions and omissions. It is within Alphabet's control to design and distribute its operating system in such a way[167] that third-party OEMs have no or only very limited ability or incentive to modify the Google Android operating system in a way that is non-compliant with Article 6(7) DMA. As explained in Section 5.4.1, the evidence shows that OEMs make decisions which render Alphabet's Google Android CPS non-interoperable, as OEMs rarely grant feature access and if they do they require compensation. This illustrates more broadly that Alphabet's proposed interpretation risks rendering Article 6(7) DMA *de facto* ineffective.

(157) Finally, as regards its own Google Android mobile devices, Alphabet is able to ensure compliance as it retains full control over the implementation of the operating system and over the contractual conditions under which it is provided to OEMs.

### 4.7. Eligibility of beneficiaries, apps and use cases

(158) Article 6(7) DMA does not provide for any limitations as to the beneficiaries, apps, services, products, and use cases for interoperability with operating system features, insofar as they are available to, or used by, the gatekeeper. If a business user or end user is eligible as a beneficiary within the scope of application of the DMA under Article 1(2), then Article 6(7) DMA applies.

(159) Indeed, one of the objectives of the DMA is to foster and promote innovation in the digital sector and remove barriers that could prevent market participants from innovating.[168] Specifically, Article 6(7) DMA aims to allow third parties to develop and provide innovative services or hardware complementing or supporting the designated operating system offered by a gatekeeper.[169] Such innovation can be fully ensured only if interoperability is not limited to a select group of beneficiaries, apps, services, products or use cases. Gatekeepers do not have discretion to decide which

---

164 Article 1 DMA.

165 Alphabet's generic claim that Google Android is open-source is inaccurate for several reasons, some of which are directly relevant to this Decision. While Alphabet publishes some parts of Google Android (specifically, AOSP) under open-source licenses, Alphabet adds many restrictive conditions when it provides Google Android to OEMs, most notably via the CDD and the GMS Requirements. First, while the AOSP licenses alone would allow anyone to customise AOSP in virtually unconstrained ways (producing a so-called "Android forks"), the CDD and the GMS Requirements tightly regulate third-party OEMs' modification of the Google Android operating system, which is incompatible with requirements on derived works for any open-source license (see The Open Source Definition, https://opensource.org/osd, Doc ID [...]. Second, a significant part of the Google Android operating system is implemented through proprietary Android middleware, whose source code is not public, and which is therefore not open-source in any way. These facts are further explained in Section 5.1 of this Decision.

166 Alphabet's response to the PFs, Doc ID [...], paragraph 120, last bullet.

167 Alphabet's response to the PFs, Doc ID [...], paragraph 120.

168 See, for example, recitals 4, 32 and 107 DMA.

169 Recital 57 DMA.

third parties, apps, services, products and use cases can benefit from the interoperability mandated by Article 6(7) DMA, whether through discriminatory restrictions of any nature or the outright exclusion of beneficiaries, apps, services, products or use cases.

(160) Subjecting feature access to pre-installation, resulting in a lack of effective access, constitutes a restriction of beneficiaries and apps, as it denies the end user full choice in selecting and using services that enjoy broad feature access. Therefore, beneficiaries of the interoperability obligation pursuant to Article 6(7) DMA include *user-downloaded apps* (also called *user-installable apps*), i.e. apps that the end user chooses to download and install (through the gatekeeper's app store or any available distribution channel[170]).

(161) Alphabet submitted that Article 6(7) DMA permits it to restrict use cases or beneficiaries for the following reasons. First, Alphabet can place conditions on interoperability provided they equally apply to its own services, such as a condition making access conditional on being a *"type of app (e.g., AI virtual assistant, search engine, etc.)*". Second, the definition of interoperability in Article 2(29) DMA supports the ability to make "functional" distinctions. Third, the framework in the Apple Connected Devices Decision should not apply to Alphabet. According to Alphabet, in that case the Commission took an extreme position that Article 6(7) allows for no bounds on use cases or beneficiaries and it has transplanted that proposition into the PFs.[171]

(162) Alphabet's arguments are without merit for the following reasons.

(163) *First,* the argument that interoperability can be subject to conditions (such as the type of app) as long as they formally apply equally to Alphabet's and third-party services is not supported by the text and spirit of Article 6(7) DMA (see Section 4.6). Nor does the text and spirit of this provision support restricting interoperability to specific use cases or beneficiaries (see above in this Section). Indeed, should that be the case, the gatekeeper would always be in control of how third parties use a feature, enabling the gatekeeper to control and limit third parties' ability to develop innovative solutions, which is contrary to the objectives of Article 6(7).

(164) *Second,* the definition of "interoperability" in Article 2(29) DMA states that software/hardware elements work with other software/hardware elements in all the ways in which they are intended to function. This means that interoperability must technically work ("function") as intended and does not mean that the intentions of gatekeepers determine the availability of interoperability for certain use cases and services.

(165) *Third,* Alphabet's argument that interoperability can be limited to a type of app as this "*leaves ample room for innovation because competitive forces that drive innovation are most acute between apps that undertake similar functions*"[172] would limit interoperability to competing apps and specific app types. This is liable to cap innovation at the level and scope of the gatekeeper's services and tie innovation to the moment in time when the gatekeeper decides to use the available operating system features in a certain type of service – if the gatekeeper were to ever offer such

170 In line with Article 6(4) DMA.

171 Alphabet's response to the PFs, Doc ID [...], paragraphs 38 to 40.

172 Alphabet's response to the PFs, Doc ID [...], paragraph 40, first bullet.

a service – rather than allowing third parties to "overcome barriers to entry" and "challenge the gatekeeper on the merits of their products and services".[173]

### 4.8. Integrity justification

(166) Pursuant to Article 6(7) DMA, gatekeepers shall not be prevented from taking strictly necessary and proportionate measures to ensure that interoperability does not compromise the integrity of the operating system, virtual assistant, hardware or software features provided by the gatekeeper, provided that such measures are duly justified by the gatekeeper.

(167) Recital 50 DMA states that the integrity of the hardware or the operating system includes any design options that need to be implemented and maintained in order for the hardware or the operating system to be *protected against unauthorised access*, by ensuring that security controls specified for the hardware or the operating system concerned cannot be compromised.

#### *4.8.1. Scope of integrity measures*

(168) Within the architecture of the DMA, integrity has a distinct meaning from end users' privacy and security. While a measure to ensure that interoperability does not compromise the integrity of the operating system, virtual assistant, hardware or software features provided by the gatekeeper ("integrity measure") may have positive effects on the privacy and security of end users, the legislator clearly distinguished these concepts in the context of the DMA, with privacy and security appearing in some provisions, namely Article 6(4) and Article 7(9), but not in Article 6(7). Article 6(4) and recital 50 refer separately to integrity and security, indicating that they are different concepts. Indeed, the reference to "*end users' security*" in recital 50 indicates that security can be pertinent to end users. Similarly, recital 72 DMA refers to privacy (and data protection) as "*interests of end users*".

(169) By contrast, the DMA does not use the term integrity with respect to end users, but only as a property of services and their features. A pertinent definition of integrity of a service or feature is the state of being unimpaired – that is, functional and not damaged or corrupted. Recital 50 DMA confirms that integrity should be regarded as the property ensuring that security controls cannot be compromised: compromised security controls are necessarily impaired. Regulation (EU) 2024/2847[174] (Cyber Resilience Act) links integrity to the absence of manipulation or modification not authorised by the user, which is again a form of compromise, consistently with the DMA.[175] The records of the legislative process that led to the adoption of the DMA indicate that the legislator considered but ultimately rejected the position that cybersecurity and end user data protection may serve as a justification to impose additional measures when providing interoperability. Instead, it decided in favour of a justification grounded on integrity only.[176]

---

173 Recital 32 DMA.

174 Regulation (EU) 2024/2847 of the European Parliament and of the Council of 23 October 2024 on horizontal cybersecurity requirements for products with digital elements and amending Regulations (EU) No 168/2013 and (EU) 2019/1020 and Directive (EU) 2020/1828 (Cyber Resilience Act).

175 Annex I Part I paragraph 2 of Regulation (EU) 2024/2847 provides that "*products with digital elements shall: […] (f) protect the integrity of stored, transmitted or otherwise processed data, personal or other, commands, programs and configuration against any manipulation or modification not authorised by the user, and report on corruptions*."

176 During the legislative process, the European Parliament's position before the Trilogue included a justification allowing for integrity, end user data protection and cyber security grounds, see P9_TA(2021)0499, Amendments adopted by the European Parliament on 15 December 2021 on the

(170) Therefore, the concept of integrity in the context of Article 6(7) DMA encompasses only risks that threaten to impair the correct functioning of the gatekeeper's operating system, or hardware or software features provided by the gatekeeper, including of security controls designed to prevent unauthorised access to the operating system or these features.[177]

(171) The absence of privacy and security justifications from Article 6(7) is logical considering the scope of the obligation, which applies to features that are *already* available to the gatekeeper's services and hardware. This is because any appropriate security and privacy measure should have already been implemented when the gatekeeper decided to make those features available to its own services or hardware – in line with best practices of security and privacy engineering and to comply with existing obligations stemming from other legislation.[178] For instance, the Android Platform Security Model explicitly includes *"malicious insiders at the [Android] platform vendor (i.e., Google)"* as a threat against which to protect with technical measures.[179] Similarly, Android's Insider Attack Resistance ensures that users *"do not need to take [Google's] word for it"*, because it is *"technically impossible [to] abuse user trust"*,[180] including for *"a potentially malicious sub-part of Google itself"*.[181] As another example, Alphabet developed Private Compute Core (Section 9.3.3.3) to process sensitive data in a way that not even Alphabet could have access to such data.[182]

(172) Opening interoperability to third parties would only increase the attack surface insofar as it could lead to *unauthorised* access. In that case, under Article 6(7) the gatekeeper is always able to implement integrity measures before providing interoperability; hence the explicit recognition in Recital 50 DMA that integrity measures can be applied to ensure that any access is authorised.

(173) In some circumstances, protecting integrity can bring benefits for end users' security and privacy.[183] Indeed, integrity measures are used to ensure that security and privacy controls are not manipulated without the end user's authorisation. For

---

proposal for a regulation of the European Parliament and of the Council on contestable and fair markets in the digital sector (Digital Markets Act) (COM(2020)0842 – C9- 0419/2020 – 2020/0374(COD)). This position was ultimately rejected, and only the integrity justification was maintained.

177 See, by analogy, judgment of 17 September 2007, Microsoft v Commission, T-201/04, EU:T:2007:289, paragraphs 1165 and 1220.

178 Recital 50 DMA explains that integrity measures exist in part to protect such controls, hence making obvious that those controls may exist and be implemented by the gatekeeper in the first place.

179 See The Android Platform Security Model (2023), https://arxiv.org/pdf/1904.05572v3, Doc ID [...], accessed on 9 April 2026.

180 This is why, for example, Android enforces end-to-end encryption before uploading backups to Google Drive, thereby ensuring that the Google Drive service does not have access to backup data. See Insider Attack Resistance in the Android Ecosystem (2019), https://www.usenix.org/conference/enigma2019/presentation/mayrhofer, Doc ID [...].

181 See https://www.mayrhofer.eu.org/post/leaving-google/, Doc ID [...], accessed on 10 July 2026.

182 Alphabet internal document, Presentation [Alphabet internal document] of 2021, Doc ID […], page 13.

183 Alphabet raises that "*the interpretation of "integrity" in the PFs does not allow for any consideration of data privacy*" (Alphabet's response to the PFs, Doc ID [...], paragraph 147), referencing the Joint Guidelines, which state that integrity measures are "consistent" with GDPR obligations to ensure security of personal data (Joint EDPB and Commission guidelines on the interplay between DMA and GDPR, version for public consultation, Doc ID [...], paragraph 94. Said version of the Joint Guidelines is available at: https://digital-markets-act.ec.europa.eu/document/download/8ba0913f-2778-4a6d-9c58-10f8c7ead009_en.) Alphabet's argument conflates two matters. Integrity indeed does not comprise privacy (or security). However, this does not mean that the PFs at large do not allow for any consideration of privacy or security. The Joint Guidelines then simply recognise the compatibility and potential synergy of integrity measures and privacy or security measures, without equating them as such.

example, integrity measures can ensure that no app can unduly disable an end user's passcode to unlock the screen, preventing unauthorised access and increasing security. Similarly, when a map app triggers a system prompt to obtain the end user's permission for access to GPS location, integrity measures may prevent a malicious actor from manipulating the prompt, to ensure that it is the end user who makes the actual choice about the end user's privacy.[184] The same applies to every security and privacy control ensuring that choices made by end users are respected, such as access to camera, microphone, or photos depending on the app; enforcing automatic VPN connections on certain Wi-Fi networks; or use of biometrics to unlock the device. Integrity ensures that these controls function without manipulation or corruption, including by malware.

(174) However, the Commission notes that some privacy and security aspects fall outside the scope of integrity within the meaning of the DMA. In particular, the concept of integrity in the DMA does not allow gatekeepers to impose their own model of security and privacy on third-party services. Indeed, nothing in the DMA precludes competition or differentiation in relation to models of security and privacy, as long as they are compliant with applicable legislation. For example, the gatekeeper should not prevent third-party apps from accessing the device's camera if they have obtained the user's consent – as access to the camera is necessary for many legitimate use cases, such as video conferencing apps. This is why camera access is already available to third parties on all major operating system and is not considered as sufficient to compromise integrity, despite the privacy and security implications that camera access comes with. Therefore, given that third-party services and hardware remain subject to applicable legislation, including on data protection and cybersecurity, the choice whether to use a service should be the prerogative of the end user and not the gatekeeper controlling the operating system.

(175) Alphabet contests the Commission's interpretation of "integrity" within the meaning of the DMA as too narrow and dangerous in light of the "sensitive" Android OS features at issue and the nature of the broad range of AI services that will purportedly be able to access them. Alphabet considers that there should be no distinction of integrity from privacy and security, with either pertaining to both the OS and the end user, and that *"nothing in the DMA prevents"* a broader interpretation of integrity.[185]

(176) The Commission notes that Alphabet's notion of integrity would in practice allow a gatekeeper to act as a universal regulator with respect to everything that happens when a business user provides a service to an end user on a device that comes with the operating system of the gatekeeper. Alphabet implies that it is its responsibility to prevent *"nefarious"*, *"illegal"*, and even *"unethical uses"* of operating system features,[186] but there is no basis in law for Alphabet to have such a general role (see Section 4.8.2 below). Pursuant to Art. 8(1) DMA, Alphabet should ensure that the implementation of its own measures complies with applicable law and not that third parties do so. Alphabet's position is also at odds with Alphabet's practice to exonerate itself from responsibility in the terms and conditions of various services it operates. For instance, Alphabet's Google Play Terms of Service explicitly state that

---

184 In this example, the permission prompt itself is therefore a privacy measure, not an integrity measure. On the other hand, a measure preventing manipulation of the system prompt would be an integrity measure ensuring that the integrity of the operating system and of the feature is not compromised. Modern operating systems employ many such measures, such as hardware-backed integrity, to ensure that system files – including those enforcing privacy prompts – are not manipulated.

185 Alphabet's response to the PFs, Doc ID [...], paragraphs 144-148.

186 Alphabet's response to the PFs, Doc ID [...], paragraph 153, fifth bullet.

*"Google is not responsible for and does not endorse any Content made available through Google Play that originates from a source other than Google"*.[187] Similarly, the terms and conditions that apply to developers using the Android SDK similarly declare that the developer is *"solely responsible for (and that Google has no responsibility to [the developer] or to any third party for) any breach of [the developer's] obligations under […] any applicable law or regulation"*.[188] Alphabet's interpretation would therefore allow Alphabet to exercise full discretion on when and how to regulate and enforce any business activity that takes place via Android – while at the same time not taking legal responsibility for it – reinforcing the power imbalance that the DMA seeks to address.

(177) Finally, Alphabet considers that the PFs apply analysis undertaken for the Apple *Connected Devices* case[189] and that these proceedings require a different approach.[190] Alphabet argues that the different approach is required due to: (i) Android's openness; (ii) the sensitivity of the features in scope; (iii) the intended beneficiaries; (iv) the fact that the AI sector is fast-moving and therefore requires a cautious approach.[191] *First,* Alphabet does not substantiate how these observations relate to integrity or the specific measures in this case. To the extent that Alphabet's comments should be interpreted to say that Android's openness and the nature of the features covered by this Decision should exempt Alphabet from complying with Article 6(7), this position has no basis in the DMA. *Second,* Alphabet appears to consider that the Commission should apply a different notion of integrity depending on the gatekeeper. However, this would lead to discriminatory outcomes in the application of the DMA. *Third,* in any case, gatekeepers can (and do) take different approaches in how integrity is protected on their platforms – a possibility that is grounded in Article 6(7) DMA. All measures specified in this Decision carefully take into account the specificities of the Android ecosystem and are explicitly justified for this specific context.

*4.8.2. Implementation in compliance with applicable law*

(178) Pursuant to Article 8(1) DMA, gatekeepers shall ensure that the implementation of their compliance measures complies with other applicable laws, such as Regulation (EU) 2024/2847 (Cyber Resilience Act), Regulation (EU) 2016/679 (General Data Protection Regulation), and Regulation (EU) 2022/2065 (Digital Services Act). These other laws are aimed at setting the standard of protection when it comes to cybersecurity, privacy and content moderation and they are not aimed at exonerating gatekeepers from their obligations under Article 6(7) DMA. Any measures taken to comply with other laws must be non-discriminatory and not impair the objectives of Article 6(7) DMA to create a level playing field and enable third-party innovation.

(179) While gatekeepers are required to implement integrity measures in relation to the interoperability solutions, this does not exclude that gatekeepers may apply strictly necessary and proportionate measures enabling end users to effectively protect security in relation to third-party software applications, pursuant to Article 6(4) DMA. Furthermore, software application store providers (including a software application store in relation to which a gatekeeper may have been designated, subject

187 See https://play.google.com/intl/en_us/about/play-terms/index.html, Doc ID [...].

188 See https://developer.android.com/studio/terms, Doc ID [...], version of 28 April 2026.

189 Case DMA.100203 (Apple – Operating systems – iOS – Article 6(7) – SP – Features for Connected Physical Devices).

190 Alphabet's response to the PFs, Doc ID [...], paragraph 149-150.

191 Alphabet's response to the PFs, Doc ID [...], paragraph 150.

to the DMA and in particular Article 6(12) DMA) may introduce safeguards[192] to prevent abuse by malicious actors, protect the security and privacy of end users, and comply with applicable law (in particular Regulation (EU) 2022/2065,[193] Regulation (EU) 2016/679, Directive 2002/58/EC, legislation on cybersecurity, consumer protection and product safety, as well as with accessibility requirements) for apps distributed on their stores.

(180) As regards cybersecurity, Alphabet and some third-party contributors to the public consultation claim that the measures conflict with the aims of cybersecurity regulations.[194] The Commission notes that under Regulation (EU) 2024/2847, manufacturers should take measures to address cybersecurity risks for products with digital elements, including operating systems, based on the "intended purpose" of the product.[195] For an operating system, in particular one that must comply with Article 6(7) DMA, interoperability is a clear part of that intended purpose. This is why Regulation (EU) 2024/2847 specifically acknowledges interoperability requirements from other Union law, which should be taken into account in devising measures to address cybersecurity risks.[196]

(181) Moreover, in the context of enabling software applications to run, only the software application store provider as a distributor bears responsibilities under Regulation (EU) 2024/2847 on the (third-party) applications it makes available on the market (i.e. supplies in the course of a commercial activity),[197] including ceasing to make the application available on the market when it is considered not in conformity with the essential cybersecurity requirements.[198] The operating system provider does not act as a distributor and therefore has no such responsibilities.

(182) Therefore, there is no conflict between the objectives of the two Regulations, counter to Alphabet's claim.[199] Gatekeepers can and should address the relevant cybersecurity risks, including by design, in general and in the context of developing interoperable solutions, as is expressly provided for by the DMA, while not impairing the achievement of the objectives of Article 6(7) DMA to create a level playing field and enable third-party innovation. For instance, the essential cybersecurity requirement to "*limit attack surfaces, including external interfaces*"[200] cannot be read in such a way that a gatekeeper can deny making such external interfaces available,[201] as this would undermine the requirement for effective interoperability under Article 6(7) DMA. Informing users about potential risks would be, amongst others, an example of a less restrictive means to protect against

---

192 Such safeguards should not undermine effective interoperability with features as prescribed in Article 6(7) DMA and implementing acts specifying measures that a gatekeeper should implement to effectively comply with the obligation in that Article.

193 Regulation (EU) 2022/2065 (Digital Services Act).

194 Alphabet's response to the PFs, Doc ID [...], paragraph 151, first bullet. [third party's] contribution to the Commission's public consultation, Doc ID [...], page 4. [third party's] contribution to the Commission's public consultation, Doc ID [...], pages 20-21.

195 Article 3(13) of Regulation (EU) 2024/2847 defines a manufacturer, and Article 3(23) of Regulation (EU) 2024/2847 defines the "intended purpose".

196 Recital 55 of Regulation (EU) 2024/2847.

197 Article 3(17) of Regulation (EU) 2024/2847 defines a distributor, and Article 3(22) of Regulation (EU) 2024/2847 defines "making available on the market".

198 Article 20(3) of Regulation (EU) 2024/2847.

199 Alphabet's response to the PFs, Doc ID [...], paragraph 151, first bullet.

200 Part I of Annex I of Regulation (EU) 2024/2847, point (2)(j). As referenced in Alphabet's response to the PFs, Doc ID [...], footnote 86.

201 Alphabet's response to the PFs, Doc ID [...], paragraph 151, first bullet.

cybersecurity risks while providing effective interoperability (see Section 4.8.3), as recognised by Regulation (EU) 2024/2847.[202]

(183) As regards data privacy, Alphabet claims that the measures conflict with Regulation (EU) 2016/679 (General Data Protection Regulation), as they constitute a "*regulatory mandate that […] risks unauthorised, large-scale access of personal data*".[203] First, the Commission notes that the measures in this Decision do not require unauthorised access to personal data in any way. To the contrary, the measures expressly provide for Alphabet's ability to request user authorisation before third-party apps can access sensitive features and any data available through them (see Section 12.2). In this respect, the measures are fully consistent with the current situation on Google Android. For example, when the user decides to change their default assistant, the user first consents that the new assistant will gain access to personal data, "*including information visible on your screen or accessible within the apps*".[204] Second, it is not the gatekeeper's responsibility as the provider of the operating system to ensure that third parties comply with the GDPR.[205]

(184) Alphabet further argues that, it cannot assume that all third parties will comply with applicable legislation such as data protection law and that the PFs seem to delegitimise any attempt at OS-level security protections beyond user prompts by stating that the choice to use a service should be the prerogative of the end user and not the gatekeeper.[206] In this respect, the Commission notes that Alphabet retains the ability to implement non-discriminatory OS-level security protections beyond user prompts.[207] However, this does not mean that Alphabet can restrict the beneficiaries that can access a feature and for which use cases, as this is not Alphabet's competence. Monitoring of compliance with specific obligations in the areas of data protection and security falls within the competence of the public authorities in charge of those respective sectors. In principle, it is not for the gatekeeper to enforce those obligations on the authority's behalf, beyond gatekeepers' obligations per applicable laws. Indeed, all providers of services or hardware including potential beneficiaries of interoperability under Article 6(7) DMA are subject to legal obligations applicable to their activities concerning, *inter alia*, security or privacy.

(185) As regards content safety, Alphabet submits that access for all use cases and beneficiaries would lead to dangerous consequences, and that it should be possible to place use case restrictions on apps and use of APIs in general, e.g. restricting usage for fraud, terrorism or other illegal activities.[208] Alphabet's argument cannot be accepted for the following reasons.

---

202 Recital 55 of Regulation (EU) 2024/2847.

203 Alphabet's response to the PFs, Doc ID [...], paragraph 151, third bullet.

204 Quote from the prompt shown when the user changes its default assistant on Android. A screenshot of this prompt is available here: https://beebom.com/how-set-copilot-default-assistant-android/, accessed 08 June 2026, Doc ID [...], accessed 08 June 2026.

205 See Joint EDPB and Commission guidelines on the interplay between DMA and GDPR, version for public consultation, Doc ID [...], paragraph 93: *"Gatekeepers should take account of the division of responsibility and liability between themselves and third parties providing apps and app stores […]. Gatekeepers should pay particular attention to any technical or contractual measures they seek to impose which may be intended to prescribe the way a third party, such as an app developer, complies with the GDPR. In this regard, as a separate and independent controller, the third party remains responsible and liable for its own processing and should therefore be free to choose how it ensures that such processing complies with the GDPR."*

206 Alphabet's response to the PFs, Doc ID [...], paragraph 153, seventh bullet.

207 For example, the gatekeeper may implement measures such as encryption, sandboxing, on-device processing, scoped access to storage, trusted execution, memory protection, and many others.

208 Alphabet's response to the PFs, Doc ID [...], paragraph 153, fifth bullet.

(186) *First,* Alphabet substantiates this argument only with respect to one feature, namely system-level ODMs, which is addressed in Section 11.2.7. Importantly, this concerns the specific circumstance of access to Alphabet's own general-purpose AI model, to which dedicated Union law applies.[209] Alphabet does not substantiate its claims for any other feature. Indeed, no other feature that is relevant to the current proceedings involves an AI model. The other features covered by these proceedings are only to be used *by* services relying on AI models, where the duty to adhere to applicable law lies with the AI developer, not Alphabet as the OS provider.

(187) *Second,* beneficiaries of the features may be subject to various legislation at Union and Member State level with respect to the dissemination of illegal content. This legislation also provides for different enforcement mechanisms, including by public and private actors. Examples of such relevant Union law that addresses the dissemination of illegal content and aims to ensure a safe, predictable and trusted online environment, are, in general, Regulation (EU) 2022/2065 (Digital Services Act), which Alphabet references,[210] and as regards terrorism, Regulation (EU) 2021/784 on addressing the dissemination of terrorist content online and Directive (EU) 2017/541 on combating terrorism.

(188) Union law places the responsibility to remove illegal or terrorist content on hosting service providers, not operating system providers. For instance, Regulation (EU) 2021/784 obliges hosting service providers to remove terrorist content within one hour of receipt of a removal order from the competent authority.[211] The DMA also clarifies that it should be without prejudice to the ability of providers of software application stores, online search engines and online social networking services to take the required responsibility in the fight against illegal and unwanted content as set out in Regulation (EU) 2022/2065.[212] Several Alphabet services, including YouTube and Google Play, are designated under Regulation (EU) 2022/2065 as very large online platforms,[213] and are therefore subject to the relevant obligations under that Regulation. Google Android has not been designated as a very large online platform or very large online search engine, and Alphabet does not claim that Google Android is an online platform, online search engine or a hosting service for that matter under Regulation (EU) 2022/2065. In any case, Alphabet may cooperate closely with the competent authorities and make use of notice and actions mechanism under this legislation.[214]

(189) *Third,* while the operating system can apply measures such as encryption and user prompts, it is unable to distinguish whether the use case for which a feature is accessed is legitimate or not. For instance, access to a device's microphone can be subject to user permission, but once the user grants access it can be used both for legal and illegal purposes,[215] which the operating system cannot distinguish. This possibility is simply accepted by the security model of Google Android,[216] as well as

---

209 Regulation (EU) 2024/1689 (Artificial Intelligence Act).

210 Alphabet's response to the PFs, Doc ID [...], paragraph 151, second bullet.

211 Article 3(3) of Regulation (EU) 2021/784.

212 Recital 62 DMA, referencing Regulation (EU) 2022/2065.

213 Decision C(2023) 2725 final (designating YouTube as a very large online platform), and Decision C(2023) 2738 final (designating Google Play as a very large online platform).

214 For example, Article 16 of Regulation (EU) 2022/2065.

215 The illegal purposes could be carried out by the user or by the app. For example, the user could use a recorder app to record other people without their consent, or the app could send the user's recording to the developer without the user's awareness.

216 The Android security model acknowledges *"abuse of APIs supported by the OS with malicious intent, e.g. spyware"* as an attack vector in Android – see The Android Platform Security Model (2023),

any other major operating system. Alphabet does not explain how use case restrictions in the context of these proceedings would be implemented via the operating system to prevent illegal uses while permitting legal ones. Use case restrictions could apply at another level, e.g., in Alphabet's software application store, where app developers have a commercial relationship and where Alphabet can at least review the apps before they are distributed.[217]

(190) In any event, in addition to Alphabet's freedom to implement non-discriminatory integrity measures to the extent they are strictly necessary and justified, the measures specified in this Decision provide for additional safeguards in relation to the access to restricted features, where targeted eligibility conditions are warranted under the exceptional circumstance of novel features used by AI services to preserve integrity and effectiveness of interoperability. The safeguards also include the option for Alphabet to suspend a service if its provider has been engaging in practices that can cause severe and immediate harm to users.

*4.8.3. Role of user authorisation*

(191) User authorisation may in certain cases be sufficient to address an integrity concern.[218] This is consistent with the goal of Article 6(7) DMA to enable user choice and innovation by alternative providers of services,[219] and is also consistent with the functioning of operating systems as platforms for third-party apps and services. Indeed, Article 6(7) DMA aims at providing access to features that are important for third-party innovation and therefore only allows gatekeepers to take measures that are strictly necessary and duly justified to ensure that interoperability does not compromise the integrity of the operating system, hardware and the features at stake. Such third-party innovation may include novel types of services and hardware that the gatekeeper does not (yet) provide, but that rely on access to features accessed or controlled via the operating system.

(192) Alphabet submits that user permission alone is inadequate given the risks of uninformed consent and social engineering, and that this is particularly the case when the primary harm may not be to the user that granted the permission.[220] Some third-party contributors to the public consultation (including some contacted by Alphabet) expressed similar comments.[221] Alphabet's argument cannot be accepted for the following reasons.

(193) *First,* Alphabet's comments misconstrue the position of the PFs. Neither the PFs nor this Decision claim that user permission is, in isolation, always sufficient in every

---

https://arxiv.org/pdf/1904.05572v3, Doc ID [...], page 6. This attack vector is mitigated by various technical measures that Alphabet has gradually implemented over the years (see page 24 of the same paper).

217 In any case, even use case restrictions applied at the level of the app store can often be easily circumvented. For example, an app might first upload user data to a remote server for legitimate purposes and then process the data for other illegal purposes. Neither the OS nor the app store can prevent this possibility. Alphabet therefore already accepts this risk when it provides Google Android and when it distributes third-party apps through the Google Play Store.

218 Annex I Part I paragraph 2 of Regulation (EU) 2024/2847 provides that "*products with digital elements shall: […] (f) protect the integrity of stored, transmitted or otherwise processed data, personal or other, commands, programs and configuration against any manipulation or modification not authorised by the user, and report on corruptions*."

219 Recital 57 DMA.

220 Alphabet's response to the PFs, Doc ID [...], paragraph 153, first through fourth bullet.

221 [third party's] contribution to the Commission's public consultation, Doc ID [...], pages 25-28. [Third party's] contribution to the Commission's public consultation, Doc ID [...], page 7. [Third party's] contribution to the Commission's public consultation, Doc ID [...], page 9.

context. However, the gatekeeper is not entitled to ignore or supersede user choice for the simple reason that user choice may entail some risks for the user. Alphabet already accepts that Android users are exposed to many risks today, including from illegal behaviour of some app developers – which is why Alphabet's Terms and Conditions for Android app developers exonerate Alphabet from legal responsibility in relation to the developers' actions (see Section 4.8.1).

(194) *Second,* granting agency to the user is generally adequate for most features, taking into account the circumstances of the feature and its implementation, and must be assessed in combination with other measures. This is what Alphabet already does today to mitigate the risks stemming from API abuse and from "dangerous permissions".[222] This Decision does not prevent Alphabet from implementing measures that give users the ability to consent in an informed manner to grant feature access to applications, including informing the user in detail about any risks and measures designed to prevent social engineering, as long as this happens in a non-discriminatory way that also applies to Alphabet's services, and without deceptive practices outlined in Article 13 DMA (Section 12.2).

(195) *Third,* the Commission notes that the risk of social engineering cited by Alphabet[223] is not novel, is not exclusive to certain features, and does not even fall in the domain of the operating system and feature access.[224] Any integrity measures taken in the context of the implementation of interoperability pursuant to this Decision must address risk related to the features at hand. As explained in Section 12.4, the introduction of novel features for use by AI systems may bring about exceptional situations, for which it may be appropriate, for a certain time, to implement measures that complement and reinforce user permission, e.g. eligibility conditions that certify that the AI system upholds user intent and permission.

(196) Alphabet states that sole reliance on user consent may not be allowed in the GDPR context, and that a controller remains subject to additional obligations on its processing operations.[225] The Commission agrees that the GDPR may require controllers to take measures in addition to user consent to comply with a variety of principles in the GDPR, such as data minimisation. However, Alphabet's argument is misplaced because the premise of that argument is wrong. Alphabet appears to assume that it is the relevant controller in this context, which would let it justify that it would be responsible for taking such additional measures. Alphabet quotes the Joint Guidelines on the Interplay between the DMA and the GDPR ("Joint Guidelines"), where it is explained that "*gatekeepers are responsible for ensuring that processing operations covered by Article 5(2) DMA at all times comply with*" such principles.[226] However, this is in the context of Article 5(2) DMA, which

---

222 Alphabet implemented *"runtime permissions"* to *"[move] the request for dangerous permissions from install to first use"* of the app – see The Android Platform Security Model (2023), https://arxiv.org/pdf/1904.05572v3, Doc ID [...], page 24. This means that, for example, an app that needs access to the camera will request the relevant permission from the user when it is actually needed for the first time, rather than when the user installs the app. Alphabet already accepts that a user's decision to grant a certain permission can impact another user. For example, a screenshot of an end-to-end encrypted messaging conversation with other users can already be shared or even accessed in the background by any app to which the user has granted permission to access photos.

223 Alphabet's response to the PFs, Doc ID [...], paragraph 153, first bullet.

224 For example, social network apps are a primary vector for social engineering threats – see Overview of Social Engineering Attacks on Social Networks (2022), https://www.sciencedirect.com/science/article/pii/S1877050921025412, Doc ID [...].

225 Alphabet's response to the PFs, Doc ID [...], paragraph 153, second bullet, and footnote 118.

226 Alphabet's response to the PFs, Doc ID [...], footnote 118, referencing the Joint EDPB and Commission guidelines on the interplay between DMA and GDPR, version for public consultation ("Joint

applies to the gatekeeper's own services where the gatekeeper is the relevant controller.[227] Alphabet omits that the Joint Guidelines explain that "*gatekeepers, as providers of operating systems, and app developers, should generally be considered as separate controllers under the GDPR*". Then, "*as a separate and independent controller, the third party remains responsible and liable for its own processing and should therefore be free to choose how it ensures that such processing complies with the GDPR*".[228] The Joint Guidelines therefore clarify that, if there were to be a responsibility to take measures in addition to user consent, this would lie with the third-party app provider as the relevant controller, not with the gatekeeper as OS provider.

*4.8.4. Justification of integrity measures*

(197) Under the DMA, the burden is on the gatekeeper to duly justify how the measures it intends to take to mitigate any integrity risk are necessary and proportionate in the context of the implementation of effective interoperability. It is the gatekeeper, having the full knowledge of its operating system, who is best placed to detect any risks to the operating system's integrity that might arise when allowing interoperability and to propose and duly justify specific measures to ensure that integrity is not compromised.

(198) In practice, to discharge this burden, the gatekeeper has to substantiate the specific integrity risks in the context of the operating system in question, the measures it intends to implement to mitigate those risks, how the measures will address the identified integrity risks, and why the measures are strictly necessary and proportionate in light of the extent to which the proposed measures may reduce effective interoperability.

(199) In doing so, the gatekeeper must demonstrate, in a verifiable way using data or other objective means, the existence and magnitude of the integrity risk.[229] In that respect, evidence regarding how other operating systems deal with the same or similar integrity risk can be relevant.

(200) To satisfy the standard of strict necessity and proportionality, the gatekeeper must, where there are several available integrity measures that are suitable to achieve the objective of mitigating the integrity risks, select the measure which is the least restrictive of effective interoperability pursuant to Article 6(7) DMA in view of its objectives and the objectives of the Regulation. This entails a balancing of those objectives on the one hand and the objective pursued by the integrity justification in

---

Guidelines"), Doc ID [...], paragraph 22. Said version of the Joint Guidelines is available at: https://digital-markets-act.ec.europa.eu/document/download/8ba0913f-2778-4a6d-9c58-10f8c7ead009_en.

227 Joint Guidelines, Doc ID [...], paragraph 18.

228 Joint Guidelines, Doc ID [...], paragraph 93. See in this context also paragraph 92 thereof, explaining the division of responsibility and liability between gatekeepers and third parties.

229 By analogy, judgments of 21 December 2023, International Skating Union, C-124/21 P, EU:C:2023:1012, paragraphs 137, 138 ("verifiable objectives"); of 26 September 2013, Ottica New Line di Accardi Vincenzo, C-539/11, EU:C:2013:591, paragraph 56. See also, by analogy, judgments of 20 March 2014, Commission v. Poland, C-639/11, EU:C:2014:173, paragraph 62; of 20 March 2014, Commission v. Lithuania, C-61/12, EU:C:2014:172, paragraph 67. The case law cited in this Section referring to the legal principle of proportionality (including in case law on the EU fundamental freedoms) is relevant since Article 6(7) DMA, second subparagraph, imposes a proportionality requirement and typically also refers to the invocation of a justification or exception.

the second subparagraph of Article 6(7) (proportionality in the strict sense) on the other hand.[230]

(201) An integrity measure may therefore not be appropriate if it disproportionately limits the attainment of the objective of Article 6(7) DMA. This should take into account, for instance, the nature of the integrity concern, the availability of alternative measures to mitigate these concerns, and the extent to which effective interoperability is affected.

(202) For example, asking users, via a permission prompt whether they would like to grant computing power to a specific app may be less restrictive than imposing strict limits on the use of such power resources for all third-party apps for integrity reasons. In this example, a permission prompt may be a sufficient protective measure in light of the objective of Article 6(7) DMA and of the DMA as a whole.[231]

(203) The legislator also clarified in recital 50 DMA, last sentence that "*[t]he gatekeeper should be prevented from implementing such measures as a default setting or as pre-installation*". An integrity measure that depends on pre-installation is therefore not strictly necessary and proportionate.

(204) Moreover, an integrity measure cannot be considered strictly necessary and proportionate if it seeks to achieve a higher level of integrity than the one that the gatekeeper requires or accepts in relation to its own services or hardware. Integrity measures can only be proportionate if they are based on transparent, objective, precise and non-discriminatory conditions that apply equally to the gatekeeper's and third parties' services and hardware.[232] These conditions must be clearly defined.[233] These requirements ensure that the integrity justification is not used arbitrarily.[234] They also ensure that conditions do not discriminate against innovative use cases and innovative types of services and hardware by third parties, including those that the gatekeeper is not yet providing.

(205) Similarly, to prevent arbitrary limitations on the obligation to allow for effective interoperability under Article 6(7) DMA, the gatekeeper should only apply conditions whose satisfaction is capable of being independently verified and are not exclusively within the gatekeeper's control. Absent verifiability, the gatekeeper retains broad discretion to abuse its power[235] and the affected third parties, other

---

230 By analogy, judgments of 12 February 2026, Vilniaus tarptautinė mokykla, C-48/24, EU:C:2026:83, paragraph 67; 23 December 2015, Scotch Whisky Ass 'n, C-333/14, EU:C:2015:845, paragraph 28.

231 See, by analogy, judgment of 20 February 1979, Rewe-Zentral AG v Bundesmonopolverwaltung für Branntwein (Cassis de Dijon), C-120/78, ECLI:EU:C:1979:42, paragraph 13, where informing customers by way of labelling was considered less restrictive than minimum alcohol content requirements set out by German law.

232 By analogy, judgments of 21 December 2023, Superleague, C-333/21, EU:C:2023:1011, paragraphs 135, 147, 254; of 28 February 2013, Ordem dos Técnicos Oficiais de Contas, C-1/12, EU:C:2013:127, paragraphs 84-86, 90, 91 and 99. These requirements – like every other requirement described in this section – apply to any other type of measure, condition, or design choice that the gatekeeper may apply in relation to the interoperability solutions – including those relating to privacy or security – since the interoperability must be equally effective and provided under equal conditions.

233 By analogy, judgments of 28 February 2023, Ordem dos Técnicos Oficiais de Contas, C-1/12, EU:C:2013:127, paragraph 99; of 16 December 2020, International Skating Union, T-93/18, EU:T:2020:610, paragraph 88.

234 By analogy, judgments of 21 December 2023, Superleague, C-333/21, EU:C:2023:1011, paragraph 255; of 22 January 2002, Canal Satélite Digital, C-390/99, EU:C:2002:34, paragraph 35; of 13 June 2019, TopFit and Biffi, C-22/18, EU:C:2019:497, paragraph 65.

235 By analogy, judgments of 21 December 2023, International Skating Union, C-124/21 P, EU:C:2023:1012, paragraphs 137, 138; of 16 December 2020, International Skating Union, T-93/18, EU:T:2020:610, paragraphs 88 ("*authorization criteria must be are clearly defined, transparent, non-*

independent third parties, the Commission, and courts would be incapable of assessing compliance. The gatekeeper could set conditions which are unachievable by third parties. Such conditions would be intrinsically liable to affect third parties more than the gatekeeper's own services and hardware, with a consequent risk that the gatekeeper could place third parties at a particular disadvantage.[236] Leaving the decision to deny or limit interoperability via the integrity justification entirely to the gatekeeper is liable to affect its objectivity and impartiality.[237] Similarly, leaving such a decision to a potential or actual competitor is liable to affect the objectivity and impartiality of such measures.[238]

(206) A gatekeeper cannot justify an integrity measure implemented in relation to third parties' services or hardware solely based on whether the gatekeeper controls or trusts such third parties. In particular, a gatekeeper may not justify integrity measures by the mere fact that third parties are not the gatekeeper, and therefore they cannot be trusted. This is because whether a gatekeeper trusts a third party is a subjective assessment exclusively within the gatekeeper's control. The requirement of "*gatekeeper's trust*" is a condition that is neither capable of being independently verified, nor objective or precise. A gatekeeper should set out the objective conditions which, in its view, mitigate or remove its integrity concerns and which should be met by its own and third-party services and hardware – for example, the requirement that apps include a "*manifest*".[239]

(207) In addition, the trust gatekeepers place into their own services and hardware may be misplaced, because the gatekeeper's own services or hardware may also pose integrity risks and risks for the end user. Indeed, the mere possibility to exercise control does not mean that services or hardware are fully protected against any integrity risk (or any other risk), such as unauthorised access or design implementation flaws[240] that invalidate the expected protections.[241] Indeed, vulnerability of apps, services, and operating systems is well documented, and providers regularly provide updates to address these vulnerabilities, indicating that despite their efforts, vulnerabilities existed.[242] Operating system providers also run

---

*discriminatory, reviewable and capable of ensuring the organisers of events effective access to the relevant market*"), 118 ("*Given the absence of objective, transparent, non-discriminatory and verifiable authorisation criteria, the applicant's broad discretion to authorise or reject such events was in no way limited.*"), and 129; judgment of 28 February 2023, Ordem dos Técnicos Oficiais de Contas, C-1/12, EU:C:2013:127, paragraph 99.

236 See, by analogy, judgments of 13 April 2010, Bressol and Others, C-73/08, EU:C:2010:181, paragraph 41; of 30 November 2000, Österreichischer Gewerkschaftsbund, C-195/98, EU:C:2000:655, paragraph 40; of 18 July 2007, Hartmann, C-212/05, EU:C:2007:437, paragraph 30.

237 By analogy, judgment of 10 March 2009, Hartlauer, C-169/07, EU:C:2008:478, paragraph 69.

238 By analogy, judgments of 10 March 2009, Hartlauer, C-169/07, EU:C:2008:478, paragraph 69; of 26 September 2013, Ottica New Line di Accardi Vincenzo, C-539/11, EU:C:2013:591, paragraph 53.

239 An app manifest is a document declaring to the operating system some important information about the app, such as which sensitive features are accessed by the app and the ways in which the app can interact with other apps. See https://developer.android.com/guide/topics/manifest/manifest-intro, Doc ID [...], accessed on 9 April 2026.

240 For example, Alphabet has recently agreed to pay $68 million to settle a class action lawsuit brought by users claiming that Google Assistant recorded private conversations after it was inadvertently triggered on their devices. See Case No. 5:19-cv-04286 (N.D. Cal.).

241 See, by analogy, judgment of 24 January 2023, Stanleybet International and Others, C-186/11, EU:C:2013:33, paragraphs 33-36, requiring that, for national legislation reducing opportunities for gambling to be lawful under Union law, it must be ensured that public authorities exercise effective and strict controls.

242 For instance, Alphabet (under the Google denomination alone) has self-reported 5,726 vulnerabilities of its products to the National Vulnerabilities Database maintained by the US National Institute of Standards and Technology:

programmes enabling third parties to identify such vulnerabilities and rewarding them.[243] Moreover, insider attacks coming from inside the operating system provider are explicitly considered as a risk to protect against by some operating system providers themselves, including Alphabet.[244]

(208) Finally, the fact that an end user has made the decision to buy a device running the gatekeeper's operating system does not mean that the end user can be assumed to automatically trust all of the gatekeeper's services and hardware that interoperate with that operating system. In fact, Article 5(2) DMA enshrines the principle that end user trust does not automatically extend to all different services provided by the same entity.[245]

(209) A condition or integrity measure is only suitable to achieve the objective of mitigating integrity risks if it genuinely reflects a concern to attain integrity in a consistent and systematic manner.[246] This requirement ensures that the condition or measure is suitable to attain the objective of protecting integrity rather than using the integrity justification as a means of arbitrary discrimination or a disguised restriction. Instances where this requirement may not be met include where the gatekeeper allows the particular integrity risk to persist in other areas, undermining the attainment of the goal[247] – either technically or due to lack of enforcement or monitoring[248] – or where the measure is not effective in reaching the goal,[249] which

---

https://nvd.nist.gov/vuln/search#/nvd/home?cnaSourceIdList=8,366,39,315,1049&resultType=records, Doc ID [...], accessed on 9 April 2026.

243 For instance, Alphabet's Android and Google Devices Security Reward Program. See https://bughunters.google.com/about/rules/android-friends/android-and-google-devices-security-reward-program-rules, Doc ID [...], accessed on 9 April 2026.

244 For example, Alphabet employees write: "*[Insider] attacks can occur at many more levels in the complex supply chain of hard- and software vendors, including […] malicious insiders at the platform vendor (i.e. Google). […] Another class of supply chain attacks are organizational attacks on a legal or political level. These may for example take the form of compelled technical insider attacks […]. The possibility of insider and/or organizational attacks at many levels is an effect of the ecosystem size, and such attacks need to be part of a realistic threat model.*" See The Android Platform Security Model (2023), https://arxiv.org/pdf/1904.05572v3, Doc ID [...], accessed on 9 April 2026.

245 Article 5(2) DMA, principles of data protection in EU law, and established data protection practices reject the idea that end user trust automatically extends to different services provided by the same entity. For example, the legal basis (including consent) for a certain processing operation does not automatically extend to any processing operation carried out by the same controller – even more so if the processing operations are carried out for different purposes by distinct services, even if provided by the same controller. Article 5(2) DMA enshrines the same legal principle by requiring specific consent for data combination and cross-use.

246 See by analogy judgments 21 December 2023, Superleague, C-333/21, EU:C:2023:1011, paragraph 251; of 8 September 2009, Liga Portuguesa de Futebol Profissional and Bwin International, C 42/07, EU:C:2009:519, paragraph 61; of 6 October 2020, Commission v Hungary (Higher education), C-66/18, EU:C:2020:792, paragraph 178; of 10 March 2009, Hartlauer, C-169/07, ECLI:EU:C:2009:141, paragraph 55.

247 See by analogy judgments of 17 July 2008, Corporación Dermoestética, C-500/06, EU:C:2008:421, paragraphs 39-40; of 10 March 2009, Hartlauer, C-169/07, EU:C:2008:478, paragraphs 60-63; of 23 December 2025, Hiebler, C-293/14, EU:C:2015:843, paragraphs 65 to 78.

248 See by analogy judgment of 24 January 2023, Stanleybet International and Others, C-186/11, EU:C:2013:33, paragraphs 33-36.

249 See by analogy judgments of 11 June 2015, Berlington a.o., C-98/14, EU:C:2015:386, paragraphs 71-72 in particular; of Jose Manuel Blanco Perez, C-570/07 & C-571/07, EU:C:2010:300, paragraphs 101-102.

requires assessing the effects of the conditions or integrity measures even after their adoption.[250]

(210) Alphabet submits that integrity must not only be considered in the course of the implementation of the measures, i.e. after the adoption of a decision in the present case, but must be also taken into account before the measures are imposed in the context of determining the measures during the administrative proceedings.[251]

(211) The Commission notes that, as a general point, the burden is on the gatekeeper to duly justify how the measures it intends to take to mitigate any integrity risk are necessary and proportionate in the context of the implementation of the effective interoperability (see recital (197) above). It is therefore, in principle, upon the gatekeeper to propose and substantiate integrity measures if it wishes the Commission to engage with such proposals in specification proceedings – including because integrity measures inherently depend on the actual implementation of the interoperability solution. This Decision, however, does not require a specific implementation of any interoperability solution, provided it complies with the requirements of Article 6(7) DMA. The gatekeeper is best placed to assess different implementation options and ultimately describe them in line with its reporting obligations relating to the measures specified in this Decision. This constitutes an in-built flexibility of Article 6(7) DMA. To the extent the implementation is known or integrity risks sufficiently identifiable, or an integrity measure has already been adopted or proposed and substantiated, the Commission can specify measures pertaining to integrity. In this Decision, the Commission specifies such measures in relation to novel features that may bring about exceptional situations and therefore might need to be subject to eligibility conditions to ensure integrity and effectiveness of interoperability, as per general measures specified in this Decision (see Section 12.4).

# 5. BACKGROUND ON GOOGLE ANDROID AND GEMINI

## 5.1. Overview of the Google Android ecosystem

(212) Sections 5.1.1 and 5.1.2 below set out, for the purpose of these proceedings, the relevant context in which Alphabet develops and currently enables access to its operating system, Google Android.

(213) Google Android is implemented by Alphabet across at least two interconnected components:

(a) AOSP (Android Open Source Project) provides the foundational framework for Google Android, described in Section 5.1.1.

(b) Google Android middleware, including for instance Google Play Services, described in Section 5.1.2.

(214) Section 5.1.3 explains the contractual framework under which Alphabet makes Google Android available to OEMs in the European Union, as this is relevant for the implementation of the measures specified in this Decision.

---

250 See by analogy judgments of 30 June 2016, Admiral Casinos, Case C-464/15, EU:C:2016:500, paragraphs 34, 36-37; of 23 December 2015, Scotch Whisky Ass 'n, C-333/14, EU:C:2015:845, paragraphs 60-65.

251 Alphabet's response to the PFs, Doc ID [...], para 155.

*5.1.1. The Android Open Source Project (AOSP)*

(215) The AOSP source code underlies Google Android, providing the foundational framework upon which Google Android powered devices operate. AOSP contains many APIs that apps use to interact with operating system services, such as hardware, networking, storage and the user interface. A subset of these APIs – called *Android SDK APIs* – are available to all apps in the same way, whether first-party or third-party apps, and whether user-downloaded or pre-installed.[252] Alphabet considers that the AOSP source code is fully included in the Google Android CPS.[253]

(216) Alphabet makes the AOSP source code available publicly and for free.[254] The AOSP source code is maintained, released, and controlled by Alphabet, who also oversees any modifications and contributions from third parties. Alphabet privately develops the next version of AOSP, including new features,[255] and releases updated source code "in bulk" in a half-yearly cadence.[256] This is less frequent than the quarterly release of new minor Android updates to the public,[257] meaning some Android features are only available in AOSP with a delay.

(217) As AOSP's project owner and maintainer, Alphabet has the final authority over which features are included in the official AOSP repository. Even though external contributors can submit code, patches and features proposals, all submitted code is reviewed and approved with Alphabet employees ("code owners") acting as the primary designated approvers.[258] Therefore, Alphabet has full control over which features ultimately form part of AOSP.

(218) The AOSP source code and documentation is licensed under an open-source, perpetual, worldwide, non-exclusive, no-charge, royalty-free license[259] by Alphabet to original equipment manufacturers (OEMs), such as its own wholly-owned subsidiary Google (which manufactures Pixel devices) or third party OEMs (e.g. Samsung or Xiaomi), which manufacture and distribute devices running on Google Android.[260]

(219) OEMs are "strongly recommended" to use the AOSP source code *"to the greatest extent possible"*,[261] but they may adapt it as is necessary to accommodate their

---

252 Alphabet's reply to the RFI of 17 July 2025, Doc ID [...], paragraphs 6-8. See also Alphabet's reply to the RFI of 18 June 2025, Doc ID [...], paragraph 17 (page 5).

253 Final Form GD, 3 July 2023, paragraph 125. Alphabet 2026 Compliance Report, Doc ID [...], Google Android Chapter, Section I, paragraph 1, and Section XVII.B, paragraph 95.

254 See https://source.android.com/, Doc ID [...], accessed on 9 April 2026.

255 In particular, Alphabet itself stresses that it is "*to focus attention on the current stable version of the Android source code while [Alphabet] create[s] the next version of the platform*". See https://source.android.com/docs/setup/about/faqs#how-is-the-android-software-developed, Doc ID [...], accessed on 26 February 2026, page 4.

256 Currently, AOSP source code is released in quarters 2 and 4 of each year. See https://source.android.com/docs/whatsnew/site-updates?year=2026#aosp-changes, accessed on 9 April 2026, Doc ID [...].

257 See https://developer.android.com/about/versions/16/qpr3/release-notes, Doc ID [...], accessed on 9 April 2026.

258 See https://source.android.com/docs/setup/contribute/submit-patches, Doc ID [...], accessed on 9 February 2026.

259 Under the Apache 2.0 license. See https://www.apache.org/licenses/LICENSE-2.0, Doc ID [...], accessed on 9 April 2026. For completeness, the Apache 2.0 license grants access free of charge, but most OEMs pay a per-device license fee for the licensing of Google Mobile Services, a bundle of Google proprietary apps and software including Google Play Services, as explained below.

260 See https://source.android.com/license, Doc ID [...], accessed on 9 April 2026.

261 Android 16 Compatibility Definition, https://source.android.com/docs/compatibility/16/android-16-cdd, Doc ID [...], Section 1.

device. However, to avoid significant differences, which is key for Google Android to be an attractive ecosystem for the developer community and to avoid any potential incompatibilities that could for example result in apps not running on certain devices, Alphabet requires OEMs to comply with Alphabet's Compatibility Definition Document (CDD). The CDD imposes requirements, including technical requirements,[262] on how OEMs can implement the AOSP source code on their device. The CDD alone contains around 2'000 mandatory requirements that OEMs must comply with. Only upon meeting CDD requirements, OEMs may market their device as "Android-compatible",[263] i.e. they obtain the right to use the Android brand,[264] and become eligible to enter into further commercial agreements with Alphabet.

(220) Every year, Alphabet releases to the market a new major release of Google Android, together with a new major release of the AOSP source code. The latest major release is Android 17, released on 17 June 2026. The next major release, Android 18, is planned to be released in Q2 of 2027.[265] In between major releases, a "Quarterly Platform Release" (QPR) or "Feature Drop" is released every three months, with feature updates and incremental improvements.[266]

(221) Pixel devices (manufactured by Alphabet's own OEM, Google) receive monthly updates, usually containing minor functional improvements[267] but sometimes with major new features, in which case those updates are also called "Pixel Drops".[268] For major releases and QPRs, Alphabet starts releasing public beta versions for Google Pixel devices a few months ahead.

(222) OEMs bear the responsibility, but not an obligation, to push the updates on their devices. Nevertheless, some OEMs commit vis-à-vis their customers to several years of Android updates for their Google Android mobile devices.[269]

---

262 As explained by Alphabet: "*This document enumerates the requirements that must be met in order for devices to be compatible with the latest version of Android.* ***To be considered compatible with Android, device implementations MUST meet the requirements presented in this Compatibility Definition****, including any documents incorporated via reference. For each release of the Android platform, a detailed CDD will be provided. The CDD represents the "policy" aspect of Android compatibility*." (emphasis added). See Android website: Android 16 Compatibility Definition, https://source.android.com/docs/compatibility/16/android-16-cdd, Doc ID [...].

263 As explained by Alphabet: "*An Android-compatible device is any device that can run any third-party app written by third-party developers using the Android SDK and NDK.* ***An Android-compatible device must adhere to the requirements of the compatibility definition document (CDD), and pass the compatibility test suite (CTS).*** *Android-compatible devices are eligible to participate in the Android ecosystem which includes potential licensure of the Android Play Store and the Google Mobile Services (GMS) suite of applications, and use of the Android trademark. Anyone is welcome to use the Android source code, but to be considered part of the Android ecosystem, your device must be Android-compatible*." (emphasis added) See Android website, Doc ID [...].

264 Alphabet's reply to RFI AI 2 of 23 September 2025, Doc ID [...], question 16.

265 [Alphabet internal document].

266 See https://developer.android.com/about/versions/16/qpr3/release-notes, Doc ID [...], accessed on 9 April 2026.

267 See https://source.android.com/docs/security/bulletin/pixel, Doc ID [...], accessed 9 April 2026.

268 See https://support.google.com/pixelphone/answer/4457705, Doc ID [...].

269 Samsung promises seven years of Android updates for their flagship devices (Galaxy S series) and up to four years for cheaper devices, see https://www.androidauthority.com/samsung-android-updates-1148888/, Doc ID [...]. Xiaomi promises at least two years of security updates for all phones and up to five years of Android updates for devices with the "Android enterprise recommended" seal, see https://trust.mi.com/misrc/updates/phone?tab=aerdata, Doc ID [...]. Fairphone promises eight years of Android updates for their newest Fairphone 6 model, see https://support.fairphone.com/hc/en-us/articles/9979180437393-Fairphone-s-Operating-System, Doc ID [...].

*5.1.2. Google Android middleware*

(223) On top of AOSP, Alphabet implements a large part of Google Android as middleware. Middleware lies between the software that is distributed as a core operating system and the applications running on it. This middleware embeds a range of features which are crucial to ensuring an effective functioning of Google Android as they contribute to controlling the basic functions of Google Android tablets and smartphones and to enabling software applications to run on them. As explained below (Section 6.2.2), parts of this middleware are part of Google Android within the meaning of the Designation Decision.

(224) To the Commission's knowledge, Alphabet distributes Google Android middleware within the Google Mobile Services (GMS) software package (also referred to as "GMS bundle"),[270] which also includes several Google applications (see Section 5.1.3 below). Most of Google Android middleware resides in the software components inside GMS called GMS Core Services.[271] A prominent part of GMS Core Services is Google Play Services, which is also referred to as GMSCore or GCore.[272] As Alphabet distributes GMS via a separate software package outside of AOSP, Alphabet can update the Google Android middleware, including GMS Core Services, much more frequently than AOSP. For example, Google Play Services is updated as often as every week.[273]

(225) Alphabet has full control over the features it includes in Google Android middleware, which is developed privately by Alphabet and licensed as closed-source proprietary software. As explained below, the addition of Google Android middleware is what turns AOSP into a fully functional modern mobile device operating system that can reliably run the millions of Google Android apps that exist today, as developers and users expect. Therefore, OEMs license and integrate GMS Core Services, including Google Play Services on almost all their Google Android mobile devices.[274]

(226) To be allowed to use Google Android middleware and Google Applications (including Google Search), OEMs must enter into the European Mobile Application Distribution Agreement (EMADA). Through the EMADA, Alphabet sets specific requirements on how third-party OEMs must implement Google Android (so-called "GMS Requirements"), which are particularly helpful for understanding the level of control Alphabet exerts over third-party OEMs.

(227) For instance, without Google Play Services, the basic functions of Google Android would not be accessible. Google Play Services enables apps to send push notifications to Google Android devices (through Alphabet's Firebase Cloud Messaging service),[275] the ability to protect the app from malware with Play Protect,

---

270 Alphabet's reply to RFI AI 2 of 23 September 2025, Doc ID [...], question 13, Annex Q.13.2, [Description of commercial agreements], Doc ID […].

271 [Alphabet internal document], Doc ID […].

272 [Alphabet internal document], Doc ID […].

273 See https://support.google.com/product-documentation/answer/14343500, Doc ID [...].

274 One exception to this is the [OEM]OEM. For [OEM], the devices without GMS only make up around 5-10%% of their direct sales. See [OEM's]reply to RFI of 5 February 2026, Doc ID [...], question 3.

275 Thousands of apps rely on Firebase Cloud Messaging, such as messaging apps, delivery tracking apps, social media apps, news apps etc. Alphabet's reply to RFI PO 11 of 09 March 2026, Doc ID [...], paragraph 17: *"It would be extremely inefficient for multiple apps on a phone to independently establish network connections so they can each simultaneously monitor and listen for such messages in parallel from each of their respective servers. This would result in unnecessary duplication, consuming battery and increasing network usage (and so cost) for the user. [...] This is a device feature app developers*

(which allows for malware scanning and runtime security checks), the ability to conduct device authenticity and anti-tampering verification with Google Play Integrity API (which detects rooted devices and reduces fraud risks), and power-efficient access to location data retrieved from device hardware (through the Google Play Services Location API).[276]

(228) Without these Google Play Services features, thousands of apps would lose access to secure payments, anti-fraud protection and efficient notifications, which Alphabet has been offering to developers as an integral part of the system experience. Some apps cannot serve their desired purpose without Google Play Services.[277] For example, without Google Play Services banking apps would refuse to launch as they rely on the Google Play Integrity API to check if the device has been manipulated,[278] messaging apps could not rely on Firebase Cloud Messaging to send notifications,[279] and ride-sharing apps could not offer reliable location updates by relying on the Google Play Services location API.

(229) The fact that Alphabet licenses Google Play Services to OEMs does not mean that it is fully interoperable for third parties. Indeed, Google Play Services includes some features that Alphabet prefers to keep closed, such as common features of Google apps.[280] According to an Alphabet's internal document, [Description of Alphabet's strategy for play services]. Some features are [direct quote describing Alphabet's commercial and strategic discussions].[281] This document discusses the following functionality in Google Play Services: [list of Google play functionalities].[282] [283]

(230) Similarly, other basic features within GMS Core Services (but outside of Google Play Services) include[284] [list of functionalities].[285] [286]

(231) The paramount importance of Android middleware for developers and OEMs was also highlighted in the context of the judicial proceedings in relation to the *Google*

---

*and users benefit from given the functionality it provides, its significant impact on mobile battery and data usage and the ability for any app to use it."*

276 Such location data is needed for a wide range of apps (e.g. apps providing navigation service). Apps need access to this API to obtain such accurate location data optimised by the device hardware. See https://developers.google.com/location-context, Doc ID [...], accessed on 9 April 2026.

277 CMA Market Study on Mobile Ecosystems, Final Report, 10 June 2022, paragraphs 8.45 and 3.168, Doc ID [...].

278 See https://appuals.com/run-bank-app-on-rooted-phone/, Doc ID [...], accessed on 10 March 2026.

279 See https://firebase.google.com/docs/android/android-play-services, Doc ID [...], accessed on 10 March 2026, which states that "*Some Firebase Android SDKs depend on Google Play services, which means they will only run on devices and emulators with Google Play services installed. These Firebase SDKs communicate with the Google Play services background service on the device to provide a secure, up-to-date, and lightweight API to your app*."

280 Alphabet submits that this is done for efficiency purposes to avoid code duplication – see Alphabet's reply to the RFI of 18 June 2025, Doc ID [...], paragraph 3 (page 11).

281 [Alphabet internal document], "Alphabet also considered that [direct quote describing Alphabet's commercial and strategic discussions]". Finally, Alphabet is concerned that including features from Google Play Services in Google Android [direct quote describing Alphabet's commercial and strategic discussions].

282 For each of these features, see [Alphabet internal document], Doc ID […].

283 See also Alphabet's reply to RFI PO 16 of 1 April 2026, paragraphs. 53-55.

284 All services listed are part of the GMS suite of apps and services. [Alphabet internal document] , Doc ID […], Case AT.40927, [Alphabet internal document], Doc ID […], [Alphabet internal document], Doc ID [...], [Alphabet internal document], Doc ID [...].

285 A large number of Android apps makes use of this functionality which is standard mobile operating system functionality. Google has separated this service from Google Android to allow for independent updates to fix potential security issues and provide an up-to-date browsing experience.

286 See also Alphabet's reply to RFI PO 16 of 1 April 2026, paragraphs 53-55.

*Android* antitrust case. The General Court held in the Android case that "[a]*ccess to Google's proprietary APIs* [i.e., its middleware, particularly Google Play Services][287] *was thus of strategic interest to developers and OEMs as, without those proprietary APIs, apps would not function correctly and the correction of those malfunctions would entail further and significant costs*."[288] The General Court also found that the technological gap between Google's proprietary APIs and the basic versions of the source code increased throughout the infringement period.[289] Alphabet submitted to the Court of Justice that OEMs have no incentive to develop their own operating system based on the AOSP (the open-source part of Google Android) unless Alphabet would license the Android middleware to OEMs.[290] In other words: the Android middleware is a must-have for OEMs for having a commercially viable operating system.

(232) Overall, Alphabet's middleware serves in principle the same purpose and operates in the same way as AOSP. Moreover, Alphabet arbitrarily decides whether to implement a functionality into AOSP or into middleware, and sometimes migrates features from one to the other, meaning that the distinction between the two has little informative value to determine as to whether a functionality (and its implementation) is part of the operating system or not. Alphabet also documents and recommends the use of middleware APIs just like it does with AOSP APIs. Indeed, around 70-80% of all Android apps distributed on the Google Play Store rely on Google Play Services APIs. Finally, Alphabet's middleware – specifically Google Play Services – is pre-installed on all certified Google Android devices distributed by OEMs. These considerations are further elaborated below.

(233) *First,* the main purpose of Google Play Services is to offer key system functionalities to apps. Alphabet itself recognises that [direct quote describing Alphabet's position on definition of Android OS].[291] Alphabet explains that Google Play Services serves as a "*core system software that enables key functionality on every certified Android device*".[292] AOSP provides thousands of APIs, some of which can be used by apps to access core features and basic functions as showing a user interface. In addition, Google Play Service adds developer APIs, as described by Alphabet: "*Google Play Services provides developers thousands of continually updated APIs that enable them to deliver high-quality experiences in their apps*",[293] offers *"core functionalities, like location services, authentication, and security updates"* and "[b]*ring the most secure and reliable Android experiences to users worldwide using the latest Google features and technologies*."[294] Further, Google Play services *"support[s] and improve[s] the Android ecosystem"* by *"provid[ing] a number of*

---

287 On the meaning of proprietary APIs and the importance of Google Play Services in particular, see Commission decision of 18 July 2018 in Case AT.40099, Google Android, recitals 576-577, 626, 631, 1133, 1164.

288 General Court, T-604/18, Google v Commission, EU:T:2022:541, paragraph 855; see also Court of Justice, C-738/22 P, Google v Commission, EU:C:2026:533, paragraphs 94, 337.

289 General Court, T-604/18, Google v Commission, EU:T:2022:541, paragraph 855; Court of Justice, C-738/22 P, Google v Commission, EU:C:2026:533, paragraph 94.

290 Court of Justice, C-738/22 P, Google v Commission, EU:C:2026:533, paragraphs 313 and 321; see also General Court, T-604/18, Google v Commission, EU:T:2022:541, paragraph 856.

291 [Alphabet internal document], Doc ID […].

292 See https://support.google.com/android/answer/10546414?hl=en, Doc ID [...], accessed on 9 February 2026.

293 See https://support.google.com/android/answer/10546414?hl=en, Doc ID [...], accessed on 9 February 2026.

294 See https://developers.google.com/android, Doc ID [...], accessed on 8 July 2026.

*APIs and core device services that enable Android to be a feature-rich, connected platform."*[295]

(234) An Alphabet internal document indicates that [description of Alphabet's position on scope of android OS]. The document states that [direct quotes describing Alphabet's position on scope of android OS] underscoring that both AOSP and GMS serve the same system level function on Google Android mobile devices.[296] Alphabet also highlights the device health aspect in stating that *"Google Play services helps to ensure the security and reliability of an Android device, and keep devices updated with the latest security features."*[297]

(235) *Second,* Google Play Services operates in a very similar, if not functionally identical, way to AOSP. Like the rest of the operating system, Google Play Services is always preinstalled on Google Android devices and therefore available out of the box on the user's device. However, unlike a preinstalled user-facing app, Google Play Services does not have an app icon on the home screen, no user interface, and is generally not acting like an app.[298] It acts as an essential system functionality, starting with the rest of the OS and always running in the background, and providing essential APIs that must be available at any time. In contrast to other apps that typically do not operate unless they are actively used (i.e. in the foreground). Google Play Services has access to many lower-level components of Google Android due to many pre-granted system privileges.[299] Indeed, to the Commission's knowledge, no other preinstalled software is granted the level of access which Google Play Services enjoys. Due to this deep integration, Google Play Services can offer developer APIs. This seems to be confirmed by Alphabet stating in its public documentation that *"Google Play services automatically obtains all permissions it needs to support its APIs."*[300] As further explained by Alphabet itself, [direct quote describing GMS core].[301] Alphabet describes Google Play Services publicly as part of the *"Android basics"*.[302]

(236) *Third,* Alphabet regularly decides whether to include a functionality into AOSP or Google Play Services. Implementing functionalities into Google Play Services gives Alphabet direct control even on third-party OEM devices. For instance, Alphabet can update Google Play Services at any time and Alphabet can add features in Google Play Services and have full control of which apps can access them, without OEMs intervention. Indeed, some of the features in Google Play Services used to be part of AOSP before Alphabet relocated them into Google Play Services.[303] Similarly, Alphabet's internal documents show [direct quote describing Alphabet's approach to

---

295 See https://support.google.com/android/answer/10546414?hl=en, Doc ID [...], accessed on 9 February 2026.

296 [Alphabet internal document].

297 See https://support.google.com/android/answer/10546414?hl=en, Doc ID [...], accessed on 9 February 2026.

298 Alphabet argued that by this measure any preinstalled app would be part of the OS. See Alphabet's response to the PFs, Doc ID [...], para 167, second bullet.

299 There permissions are set out in the GMS Requirements and linked websites, see Alphabet's reply to PO RFI 17, Annex Q18.2, [Alphabet internal document], Doc ID [...], [description of commercial agreements].

300 See https://developers.google.com/android/guides/permissions, Doc ID [...], accessed on 9 April 2026.

301 [Alphabet internal document], Doc ID […].

302 See https://support.google.com/android/topic/7355453?hl=en, Doc ID [...], accessed on 8 July 2026.

303 For example, Alphabet provided an API to implement and execute neural networks and machine learning models on Android as part of AOSP (NN API). This API has been deprecated and Google now refers to the LiteRT runtime that can run machine learning models as part of Google Play Services.

AOSP].[304] This underlines the close link between AOSP and Google Play Services and shared nature of being system software.[305]

(237) *Fourth,* Alphabet encourages app developers to use and rely on Google Play Services functionality in the same way as AOSP functionality. Thousands of APIs exposed by Google Play Services are publicly documented and advertised in a similar fashion as APIs exposed by the AOSP source code. Alphabet does not deprecate or change these APIs without prior announcement, i.e., the same reliable approach which Alphabet applies to APIs which are part of AOSP.[306]

(238) Differently, to other apps which may also offer APIs to make available features of the app to other apps on the device, e.g. Spotify allows apps to start playing music on Spotify via an API. Google Play Services offers Android developer APIs at the OS-layer. As such Google Play Service offers APIs to access the Google Android's device location[307] (via satellite or nearby Wi-Fi networks) and allowing to setup connections with nearby Google Android devices to exchange data and files.[308]

(239) Importantly, for third-party apps to operate on a level-playing field with Alphabet's apps, access to many of these features in the same way as Alphabet is indispensable in practice. Alphabet and third-party developers build apps for Google Android devices using the APIs provided by Google Play Services. App developers rely on access to Google Play Services APIs to deliver functional apps on Google Android mobile devices, which are designed with the expectation that Google Play Services is available, and its functionalities are accessible to their apps. Indeed, Alphabet estimated that, as of April 2022, 70-80% of apps available on the Google Play Store use at least one Google Play Services API.[309]

(240) Therefore, it is the system software Google Play Services that turns AOSP into a fully functional modern mobile device operating system that can reliably run the billions of Google Android apps that exist today, as developers and users expect. Therefore, Google Play Services, through its system functionalities and developer APIs, enables software applications to run on Google Android. For these reasons, in practice OEMs have de facto no choice but to license and pre-install Google Play Services on their Google Android smart mobile devices, if they want to be attractive for developers and ultimately users. Indeed, save for very few exceptions[310] any

---

304 [Alphabet internal document], Doc ID […].

305 Moving features into Google Play Service confers on Alphabet more control as it can itself update Google Play Services across various Google Android versions and Google Android devices instantaneously, without third party agreement or intervention. Indeed, Alphabet uses its middleware to update the Google Android operating system on third-party OEM devices without requiring action from third-party OEMs.

306 See Google for Developers (2026), 'Google Play services', Google for Developers website, https://developers.google.com/android, Doc ID [...], accessed 13 March 2026 ("Bring the most secure and reliable Android experiences to users worldwide using the latest Google features and technologies."). In fact, deprecation of features is always announced, and a deprecation period of 12 months is typically observed during which the feature will continue to be available. For most deprecations, Alphabet recommends an alternative approach to provide the same functionality. See https://developer.android.com/about/versions/12/deprecations, accessed on 5 March 2026, Doc ID [...].

307 See *Simple, battery-efficient APIs for location and context,* https://developers.google.com/location-context, Doc ID [...].

308 See Nearby Connections, *https://developers.google.com/nearby/connections/overview,* Doc ID [...].

309 CMA Market Study on Mobile Ecosystems, Final Report, 10 June 2022, https://assets.publishing.service.gov.uk/media/63f61bc0d3bf7f62e8c34a02/Mobile_Ecosystems_Final_Report_amended_2.pdf, Doc ID [...].

310 For instance, [OEM] was cut off from Google Android due to U.S. sanctions and therefore could no longer source GMS from Alphabet. [OEM] offers devices without GMS [OEM's] [confidential

mass-market OEM licenses GMS Core Services (which includes Google Play Services).

(241) *Fifth,* almost all Android smartphones with appreciable sales in the European Union come with GMS Core Services (of which Google Play Services is a component) pre-installed.[311] As a result, Google Play Services exhibits strong network effects that, for all intents and purposes, are in line with those of AOSP (and Google Android as a whole): developers rely on Google Play Services APIs to develop most apps; end users expect to be able to use those apps (and the functionalities that they support thanks to Google Play Services); and OEMs must therefore pre-install GMS Core Services (and comply with Alphabet's contractual requirements) – otherwise their devices would not support most Android apps, meaning that consumers would not buy them.

(242) In its response to the PFs, Alphabet accepts that its middleware may in principle fall within the designated services Google Android OS insofar as it fulfils the definition under Article 2(10) DMA but argues that it has not identified any component that would fulfil that definition[312] and that "not all" of the middleware features listed above are part of the Android OS.[313] Beyond this generic statement, Alphabet does not specify which middleware features do not form part of Android OS nor on what basis, except for Google Hotword Engine, AICore/Gemini Nano models (which are discussed below in this section) and Android/Pixel System Intelligence. Alphabet simply states that the PFs fail to substantiate that Alphabet's middleware is part of the Google Android CPS and it deliberately avoids to *"analyse each of those for the purposes of this response, as the majority are entirely unrelated to the proceedings at hand."*[314] Instead, Alphabet makes some *"overarching observations"* and *"reserves the right to put forward an assessment of whether [a specific software component] fulfills the definition of an OS in the event that Google receives a specific request for interoperability with that component"*.[315]

(243) In its overarching observations, which focus on Google Play Services, Alphabet claims that some of the findings in the PFs concerning in particular the amount of

---

commercial information on an OEM]. See Agreed minutes of meeting with [OEM] on 9 March 2026, Doc ID […], paragraphs 1-3, 8-9,11, 12. See also Competition and Markets Authority Mobile Ecosystems 2022 market study, Annex E, p. 15-16, which details that Alphabet stated that: *Where a developer uses Google proprietary APIs for its app(s), the proper functioning of the app can only be guaranteed if the device also runs Google Play Services (though some Google proprietary APIs may function without Google Play Services)' – in this regard Google estimated that, as of April 2022, [70-80]% of apps available on the Play Store use at least one Google Play Services API."* And further detailed that: "*To the extent that the gap between the features and functionalities offered by the APIs in Android Open Source Project and the APIs in GMS increases over time, the cost of replicating these APIs for alternative providers would also increase and it would be more difficult for app developers to port their apps to Android Forks or other versions of Android not using GMS."* Doc ID [...].

311 Alphabet itself acknowledges that [information related to Alphabet's commercial agreement] see Alphabet's reply to RFI PO 1 of 6 February 2026, Doc ID [...], question 2. [Number of OEMs that have entered into specific commercial agreements]. See Alphabet's reply to RFI AI 2 of 23 September 2025, Doc ID [...], question 17, paragraph 45. Alphabet's reply to RFI AI 2 of 23 September 2025, Annex Q17, Doc ID [...]. [OEM] and [OEM]indicated that all their Android mobile devices are subject to the EMADA and have Google Play Services (and therefore also GMS Core Services) pre-installed. See [OEM's]reply to RFI 1 of 5 February 2026, Doc ID [...], question 3.a.; [OEM's] reply to RFI 1 of 28 February 2026, Doc ID [...], question 3.a.-; [OEM's] reply to RFI 1 of 5 March 2026, Doc ID [...], question 3.a.

312 Alphabet's response to the PFs, Doc ID [...], para 167.

313 Alphabet's response to the PFs, Doc ID [...], para 166.

314 Alphabet's response to the PFs, Doc ID [...], para 166.

315 Alphabet's response to the PFs, Doc ID [...], para 166 and footnote 127.

permissions that are granted to it, the fact that it is pre-installed on all Google Android devices, the fact that it exposes publicly documented APIs, and the fact that it controls features which are expected on a modern mobile device – are all irrelevant considerations because these properties are shared by other apps that are not part of the operating system.[316] Alphabet also claimed that it is irrelevant whether a feature controlled by the middleware is important to other apps, because only *"'basic functions' which are required for the operation of the device, or control management of critical device resources"* are covered by the definition of operating system.[317]

(244) The Commission has analysed Alphabet's arguments and considers that these arguments do not hold, for the following reasons.

(a) *First,* Alphabet seems to consider each of the main characteristics of Google Play Services *in isolation* and argue that it does not imply that Google Play Services provides "basic functions". However, Alphabet's arguments do not address the nature of Google Play Services as a whole. For example, while certain non-Alphabet apps may be preinstalled *or* may be granted extensive permissions *or* may expose publicly documented APIs, it is rare that they meet all such conditions at the same time on all Google Android devices. In fact, Alphabet did not provide a single example of such an app.[318]

(b) *Second,* this is at odds with Alphabet's own submission that Google Play Services are relied on by 70-80% of apps. Indeed, this fact is difficult to reconcile with the claim that it does not provide basic functions and does not enable software applications to run on it.[319]

(c) *Third,* Alphabet's claims are directly contradicted by the many internal documents cited above in this section. The specific claim that user expectations are irrelevant is directly contradicted by an Alphabet's internal document stating that system services are [describes Alphabet's position on system services]*;* that is inter alia the case if the service is headless (no app icon), is preinstalled, is signed by Google, and perform basic device functionality (for which is it sufficient to be assigned as a default app by the platform – see also recital (377).[320] Making features accessible by the operating system entails a particular commitment and predictability which is critical for developers as it ensures that their apps function in the same manner on all devices across the operating system.

(245) In any event, for the purpose of this Decision, the relevant parts of Alphabet's middleware that are part of the Google Android CPS and that are subject to measures in this Decision are only AICore/Gemini Nano and the Google Hotword Engine, and they are further discussed in Sections 6.2.2.1 and 6.2.2.2 of this Decision.

*5.1.3. Relevant aspects of Alphabet's contractual relationship with third-party OEMs*

(246) An important aspect for the case at hand, and in particular to understand the degree of control Alphabet exerts over the Android ecosystem, is the web of contractual relationships between Alphabet and Google Android OEMs. To recall, OEMs must enter into the European Mobile Application Distribution Agreement (EMADA) to be

316 Alphabet, response to the PFs, Doc ID [...], paragraphs 167.
317 Alphabet, response to the PFs, Doc ID [...], paragraphs 168.
318 The three apps mentioned by Alphabet that expose APIs ([third party apps]) are not pre-installed on most Google Android devices.
319 Article 2(10) DMA.
320 [Alphabet internal document], Doc ID […].

allowed to use Google Android middleware and Google Applications (including Google Search).

(247) Alphabet requires OEMs to sign the EMADA (and enter into Alphabet's Android Compatibility Commitment (ACC)) to be allowed to use:[321]

(a) Google Android middleware (GMS Core Services), including Google Play Services, and

(b) Any of the Google Applications. These include the Google Play Store, Google, Chrome, Gemini, Gmail, Maps, YouTube, Meet, and Photos.

(248) Under the EMADA, Alphabet must approve all Android software builds from any third-party OEM before being launched as Google Android.[322] Alphabet only approves software builds if Alphabet considers that the device complies with the obligations set out in the EMADA. These obligations include pre-installing the Google Android middleware (i.e. GMS Core Services)[323] as well as certain Google Applications.[324] [Description of obligations regarding security stemming from contractual arrangements between Alphabet and OEMs].[325]

(249) The Google applications that the OEM must preinstall include Gmail, Maps, YouTube, Play Store, Meet, and Photos.[326] [Description of commercial agreements with OEM regarding apps pre-installation].[327] [Description of commercial agreements with OEM regarding apps pre-installation].[328] To license the Google app (which includes the Google Search service) and Chrome, OEMs must sign separate licence agreements.[329]

(250) In addition, the EMADA requires OEMs to comply with the "GMS Requirements".[330] The GMS Requirements consist [description of commercial agreements] of detailed requirements on third-party OEM mobile devices, the operating system, and pre-installed apps.[331] The OEM must ultimately guarantee

---

321 [Alphabet internal document], provided as part of Alphabet's reply to RFI AI 3 of 17 October 2025, Doc ID [...], Doc ID [...], Doc ID [...], Doc ID [...], Doc ID [...], Doc ID [...], Doc ID [...], Doc ID [...].

322 See [Commercial agreement] (see Alphabet's reply to RFI AI 2 of 23 September 2025, Doc ID [...], Annex Q13.1, Doc ID […]). See for instance Alphabet internal document, Email stating that [direct quote describing an OEM's approval process], Doc ID […].

323 Alphabet's reply to RFI PO 3 of 17 October 2025, Doc ID [...], Annex Q.2.1, [Alphabet internal document], Doc ID […].

324 [Content of Alphabet's commercial agreements] see Alphabet's reply to RFI AI 2 of 23 September 2025, Doc ID [...], Annex Q13.1, Doc ID […]). [Content of Alphabet's commercial agreements]. (see Alphabet's reply to RFI AI 2 of 23 September 2025, Doc ID [...], Annex Q13.1, Doc ID […]).

325 [Content of Alphabet's commercial agreements]. (see Alphabet's reply to RFI AI 2 of 23 September 2025, Doc ID [...], Annex Q13.1, Doc ID […]).

326 Alphabet's reply to RFI AI 3 of 17 October 2025, Doc ID [...], Annex Q2.1. Alphabet's reply to RFI PO 3, Doc ID [...], Annex Q2.2, Doc ID [...].

327 Alphabet's reply to RFI PO 3 of 17 October 2025, Doc ID [...], Annex Q2.1. Alphabet's reply to RFI PO 3, Doc ID [...], Annex Q2.2, Doc ID [...].

328 [Description of commercial agreements with OEMs] provided as Annex Q1.1.1 to Alphabet's reply dated 27 October 2025 to RFI AI 3, Doc ID [...] provides: [Description of commercial agreements with OEMs] See also Alphabet's reply to RFI AI 2 of 23 September 2025, Doc ID [...], [Direct quote form commercial agreements].

329 [Alphabet internal document], see Alphabet's reply to RFI AI 3 of 17 October 2025, Doc ID [...], Annex Q.2.1 and Q.2.2, Doc ID [...] and Doc ID Doc ID [...].

330 See Alphabet's reply to RFI AI 2 of 23 September 2025, Doc ID [...], [Direct quote form commercial agreements, Doc ID [...], ; Alphabet's reply to RFI AI 2 of 23 September 2025, Doc ID [...], question 13, Annex Q.13.2: "Annex Q13.2 - DMA.100210 - EC AI RFI #2 of 23 September 2025 (8 October 2025) (Confidential - Contains Business Secrets).pdf", [Description of commercial agreements].

331 [Description of commercial agreements with OEMs]

compliance with all policies established by Alphabet. Therefore, Alphabet exercises a great degree of control over how third-party OEMs implement Google Android on their mobile devices.

(251) [Description of Alphabet's commercial agreements with OEMs], Alphabet requires OEMs to grant some of the Google Applications extensive "system privileged permissions" (see Section 5.2.3), which enable access to important features, including those addressed in this Decision. OEMs have no choice but to grant these system permissions, as not granting them would mean [description of Alphabet's commercial agreements with OEMs] and thus Alphabet not approving the OEM's device as Google Android certified device.[332] Alphabet validates the compliances through several test suites, including the Google Mobile Services Test Suite (GTS).[333] In the context of these proceedings, [OEM] (an OEM) explained that it *"does not touch the GMS package, they just integrate it. Any change to the GMS package could also lead to test failures. Google would not certify their Android version if one of the tests would fail."*[334]

(252) Finally, there are substantial flows of payments between Alphabet and third-party OEMs pursuant to the various agreements. Pursuant to EMADA, [description of Alphabet's commercial agreements with OEMs].[335] OEMs receive payments from Alphabet if they pre-install the Google app (which includes Google Search and Gemini) and the browser app Chrome. In case of Samsung, Alphabet paid Samsung between [description of Alphabet's commercial agreements with OEMs] in calendar year 2025 for the EEA.[336]

## 5.2. Access controls in Android

(253) In addition to the net of various agreements, the complexity of the Android ecosystem is further reinforced by an intricate system of access controls through which Alphabet controls the various degrees of access to features of Google Android. Indeed, this system ultimately determines the scope of accessible features, the stakeholders who enable the access (Alphabet, OEMs or users) as well as the scope of beneficiaries that can obtain access. Depending on various parameters, this can range from (i) features where access is very limited, is controlled by Alphabet and its contractual relations with OEMs, and therefore in practice is only accessible by Alphabet, to (ii) features where the end user has the agency to grant it to any app that the end user installed and chose to grant those permissions to.

(254) Through this access control system, Alphabet allows OEMs to either fully block apps installed by the end user (i.e. user-downloaded apps) from obtaining access to the features covered by this Decision or limit such access to a single app (e.g. default app).

---

332 Agreed minutes of meeting with [OEM] on 9 March 2026, Doc ID [...], paragraph 14. [OEM] explained that "[OEM] *would have to integrate the GMS package in any event, without implementing any changes, including not introducing any changes to the permissions, which are not subject to any negotiation. Any negative result from the security or privacy vetting of Google's apps could therefore only result in deciding not to pre-install Google's apps at all, which means that* [OEM] *cannot distribute devices with GMS even though it is critical to some developers and users*".

333 [Alphabet internal document]

334 Agreed minutes of meeting with [OEM] on9 March 2026, Doc ID [...], paragraph 12.

335 [Alphabet internal document], Doc ID […].

336 [Alphabet internal document], Doc ID […].

*5.2.1. The operating system decides which apps have access to features and Alphabet decides how feature access is implemented in the operating system*

(255) Google Android is fully owned by Alphabet and Alphabet decides how access controls are implemented in their OS. When implementing a new feature, Alphabet takes a decision on the degree of access that guards the feature. Alphabet may choose any of the existing access controls (see Sections 5.2.2 to 5.2.5) or introduce additional access controls, such as in relation to App Functions (see Section 10.2.3.2) or Computer Control (see Section 10.3.3). For instance, in the context of the roll-out of Alphabet's Circle to Search solution on Android devices, Alphabet considered several possibilities including creating an "*open API*" and creating an "*OEM-controlled API*". This shows that Alphabet is always in full control over access to the Google Android features.[337]

(256) Importantly, under the CDD, OEMs are contractually obliged to enforce the access control system set up by Alphabet on their Google Android mobile devices[338] : "*[n]o permissions and no roles may be omitted, altered, or ignored"*.[339]

(257) The Commission notes that Alphabet grants access to the vast majority of features controlled by Google Android to all apps, regardless of any OEM decisions. In fact, OEMs are contractually required by the CDD to keep Android SDK APIs intact and open to all apps.[340] Only some features are implemented in a way that access is tied to preinstallation by OEMs (see Section 5.2.5). At any point in time, Alphabet is free to change this decision by simply reclassifying the feature as a public API in the AOSP source code. This only requires Alphabet to change a few lines of the AOSP source code.[341]

(258) Moreover, Alphabet requires OEMs to grant Alphabet's own services access to some Android features via the contractual framework described in Section 5.1.3 above. [Description of Alphabet's commercial agreements regarding hotwords].[342]

(259) The degree of control Alphabet enjoys is further evidenced by its ability to waive any of the GMS requirements it imposes on OEMs to derogate from obligations under

---

337 [Alphabet internal document], Doc ID […].

338 As explained in Section 5.1.3, the CDD (Android 16 Compatibility Definition, https://source.android.com/docs/compatibility/16/android-16-cdd, Doc ID [...]) imposes technical obligations on OEMs, effectively dictating how the AOSP should be implemented on their devices. In particular, the CDD prevents OEMs from being able to modify the Android permission model: "*The CDD makes it mandatory for OEMs and OS developers that subscribe to the CDD to enforce, without alteration, each permission as described in the SDK documentation*." (See Alphabet's reply to RFI 2 of 23 September 2025, question 23). Section 3.1 of the CDD entitled "*Managed API Compatibility*" specifically mandates OEM to implement the AOSP in accordance with the Android permission levels chosen by Alphabet for each API: *"[C-0-5] MUST NOT allow third-party apps to use non-SDK interfaces*".

339 Android 16 Compatibility Definition, https://source.android.com/docs/compatibility/16/android-16-cdd, Doc ID [...], Section 9.1.

340 See Alphabet's reply to the RFI of 18 June 2025, Doc ID [...], paragraph 17 (page 5). See also Section 3.1 of the CDD (https://source.android.com/docs/compatibility/16/android-16-cdd, Doc ID [...]), especially requirement [C-0-3], establishing that OEMs *"MUST NOT omit any managed APIs, alter API interfaces or signatures, deviate from the documented behavior, or include no-ops, except where specifically allowed by this Compatibility Definition"*.

341 For example, the AlwaysOnHotwordDetector API is marked as a system API using the @systemAPI annotation in the source code. Furthermore, the API checks if the app has certain permissions, e.g. @RequiresPermission(allOf = {RECORD_AUDIO, CAPTURE_AUDIO_HOTWORD}). If Alphabet would remove the annotation and the permission checks, it would allow any third-party app to use the API.

342 See Alphabet's reply to RFI AI 2 of 23 September 2025, Doc ID [...], [Description of Alphabet's commercial agreements regarding hotwords].

signed agreements to Alphabet's own benefit.[343] [describes commercial agreements][344] [describes commercial agreements with OEMs].[345]

(260) Alphabet may discuss with OEMs certain elements of its access control system or the requirements it seeks to impose on them (e.g. the GMS Requirements), for reasons such as early information of anticipated changes so that OEMs can prepare themselves or regarding the implementation effort that such requirements may entail for OEMs. However, Alphabet is the final and sole arbiter of the access controls and requirements it imposes on OEMs.

(261) Given the lack of alternatives and the strong network effects enjoyed by Alphabet's middleware, OEMs have little choice but to pre-install Alphabet's middleware and accept Alphabet's contractual conditions (see Section 5.1.2).

*5.2.2. General access control model on Google Android*

(262) Alphabet operates a multi-layer access control model whereby it decides for each Google Android feature if and under which conditions an app can access an API, feature, or resource. This multi-layer model allows access levels ranging from full access to APIs for all apps on the device to APIs which are reserved for the operating system's internal processes only.

(263) The different Android *access levels* for AOSP APIs are:

(a) **Public APIs** (or **Android SDK APIs**). These APIs can be used by any app that is installed on the device. These APIs are part of the public Android Software Development Kit (SDK).[346] In general, a third-party developer of a user-downloaded app, e.g. distributed through the Google Play Store, can only access public APIs. As mentioned in the previous section, Alphabet mandates that OEMs must preserve and make accessible these APIs to user-downloaded apps; and

(b) **System APIs.** System APIs represent a set of APIs that is reserved for the OS, OEMs and partners of OEMs.[347] System APIs are not available to third-party developers of user-downloaded apps. To use a system API, an app must be preinstalled by the OEM;[348] such apps are also called "system apps".[349]

---

343 Annex B to Alphabet's Report on Document Production dated 24 February 2026, Doc ID [...].

344 Alphabet's internal documents [description of Alphabet internal discussions regarding product design choices] See email with subject [Alphabet internal document], Doc ID […], page 1.

345 [Description of Alphabet's commercial agreements], Alphabet internal document, [Alphabet internal document] Doc ID […].

346 See https://source.android.com/docs/core/architecture, Doc ID [...], accessed on 23 March 2026.

347 Alphabet's reply to the RFI of 17 July 2025, Doc ID [...], question 42-43. See https://source.android.com/docs/core/architecture, Doc ID [...], accessed on 23 March 2026. In between system APIs and public APIs, also hidden APIs do exist. These APIs could be used by any app that is installed on the device, but they are intentionally hidden from developers creating the apps and it is not advised to use the APIs. These APIs are unstable and using them could lead to crashes of the app. Developers would have to work around restrictions to use hidden APIs.

348 The Android OS is split into different partitions. A system app must be installed in the on the system, vendor, or product partition The system partition contains the main parts of the operating system. The vendor partition contains hardware-specific parts for the current device, e.g. access to the camera. The product partition contains OEM customisation, e.g. styling of the device. All three partitions can be used to preinstall system apps.

349 Alphabet's reply to the RFI of 17 July 2025, Doc ID [...], question 42-43. See https://developer.android.com/guide/app-compatibility/restrictions-non-sdk-interfaces, Doc ID [...], accessed on 18 March 2026.

(264) Any API from either access level can be further restricted by requiring that the app must be granted certain Google Android *permissions*. In that case, the OS allows an app to use the API only if the permission has been granted to the app. Furthermore, any API from any access level can be further restricted by limiting access to apps that are assigned an Android role, are set as the default application, or are configured as a "system service". These assignments grant an app additional, sometimes exclusive, permissions. The next three sections provide more details on these mechanisms to restrict feature access.

(265) Some Google Android APIs are not part of AOSP but rather of Alphabet's middleware (most notably Google Play Services) – see Section 5.1.2 above. For these APIs, Alphabet decides who gets access and no other party, including OEMs, has a role in the decision.

*5.2.3. Conditioning access to features to apps holding certain permissions*

(266) The Google Android source code defines a list of permissions that an app must hold to be allowed to use certain APIs or resources,[350] e.g. accessing the microphone to record audio (via the *RECORD_AUDIO* permission) [351]

(267) Google Android permissions are tiered by *protection levels*. Depending on the protection level, different prerequisites are necessary to grant a permission to an app. The most important protection levels for Google Android permissions are described below.[352]

(a) **Normal.** The normal protection level is for *install-time permissions*. These permissions are implicitly granted when installing the app. For example, the permission to access the internet has the normal protection level;

(b) **Dangerous.** The dangerous protection level is for *runtime permissions.* An app is required to request user consent while running showing a separate prompt for each permission. For example, the possibility to access the microphone audio is a dangerous permission;

(c) **App-op.** The app-op protection level is for *user-switchable permissions* that are not *runtime permissions.* An app cannot prompt the user to grant this permission. The user must navigate into the device settings and intentionally grant the permission to an app.[353] For example, the API to access all files of the user stored on the device is guarded by an app-op permission;

(d) **Privileged.** The privileged protection level is for the special category of preinstalled apps: privileged system apps. Permissions with this protection level must be granted by the OEM through preinstallation, and a configuration of the privileged permission allowlist. APIs that are restricted by virtue of a

350 In the Android source code, one file called *AndroidManifest.xml* defines most permissions that exist in the operating system.

351 See https://cs.android.com/android/platform/superproject/+/5750195121e36bb55b446b77c2d212a5770b2e43:frameworks/base/core/res/AndroidManifest.xml;bpv=1, Doc ID [...], accessed on 24 March 2026.

352 See https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/permission/Permissions.md, Doc ID [...], accessed on 7 March 2026.

353 These permissions are used for "capabilities that present even greater risk" and require the user to perform extra steps to grant them. See Annex 1 of Alphabet's response to the PFs, Doc ID [...], para 26.

privileged permission are termed *privileged APIs.*[354] For example, the permission to execute App Functions is privileged;

(e) **Role.** The role protection level limits the permission to apps that are assigned to a specific Google Android role, such as the default digital assistant or the default SMS app. Setting an app as this role would automatically grant it all permissions that are assigned to the role (see Section 5.2.4); and

(f) **Signature.** The signature protection level is limited to software that are built by the OEM. This is one of the highest protection level, limited to software made by the OEM with the intention to extend the system.

(268) Some permissions can be granted by the end user. This includes all permissions with the protection levels normal, dangerous, and app-op. In addition, some roles are also user-switchable, such as the default SMS app or the default digital assistant. While there are different paths on how users can grant these permissions, e.g. granted on installation vs. granted manually in the device settings apps, the user has the agency to grant those to any user-downloaded app.[355]

(269) By contrast, some permissions must be granted by the OEM prior to selling the device or via an OS update and cannot be granted by the user. This includes all permissions with the protection level signature and privileged. These permissions must be granted by the OEM prior to selling the device or via an OS update. A user therefore has no possibility to grant this permission to a third-party user-downloaded app that they have chosen and installed.

(270) In addition, the Commission notes that if an app is no longer used by the user, Android will remove the user-granted permissions of this app. This ensures that an app does not access sensitive data, even when the user has stopped using it, but forgot to delete it.[356] This does not apply to permissions pre-granted by an OEM to an app, such as privileged permissions, which remain granted. Such an app could therefore indefinitely access sensitive data, even if an end user does not actively use the app.

(271) The following two sections provide more details on how certain apps hold a special status that allows them to have privileged or even exclusive access to certain permissions – and, therefore, to the OS features that those permissions grant access to.

*5.2.4. Conditioning access to features on roles, default applications, and services*

(272) Google Android limits access to certain features to apps which are (i) assigned a role, (ii) set as a default application (which stems from a type of role), or (iii) configured as system services. As a consequence, some features can only be accessed by one single app at a time.

---

354 System apps must be preinstalled by the OEM in the /system, /product, or /vendor partition of the operating system. A privileged system app must be preinstalled in the priv-app folder of one of the mentioned partitions. Then the OEM must list the granted permissions in an allowlist. See https://source.android.com/docs/core/permissions/perms-allowlist, Doc ID [...], accessed on 23 March 2026.

355 The expert report of Prof. [professor], annexed to Alphabet's response to the PFs, confirmed the above definition of the Android permission model. He described the permission model as a way to guarantee the security principle of "least privilege", guaranteeing that an app only has the permission to access features and data that it needs for perform the app's functionality. The fact that many permissions are not granted upon installation is a way to guarantee the "secure defaults" security principle. See Annex 1 of Alphabet's response to the PFs, Doc ID [...], para 9, 10, 27, 28.

356 See Annex 1 of Alphabet's response to the PFs, Doc ID [...], para 27.

(273) **Roles.** Android defines "*roles*" that are associated with a set of permissions. These permissions are all granted together when that role is granted to an application, subject to certain requirements. For example, the SMS role grants an app multiple permissions including reading and writing a user's SMS database.[357] Alphabet defines what type of functionality an app must offer to hold the role.

(274) For certain features, Alphabet requires that an app holds a specific "role" to gain access.[358] In addition, Alphabet defines roles with attributes that can impose limitations on who can access the role, and therefore, possibly limitations on who can access the features gated by that role. Such attributes are not mutually exclusive. Attributes relevant to feature access are:

(a) **Exclusive**. The "exclusive" (or "exclusivity") attribute limits a role to being held by one app at a time.[359] This limits certain permissions and features to that one app. Deciding which roles are exclusive is at Alphabet's discretion;

(b) **Visible**. The "visible" roles are under control by the user. For a role that is "requestable", the app can prompt the user to grant that role to the app.[360] Otherwise, the user can configure which app(s) hold the role in system settings.[361] If the role is not marked as "visible", the user cannot change which app holds the role and therefore cannot grant access to features exclusive to that role to another app; and

(c) **System Only**. The "systemOnly" attribute limits the role to being assigned to system apps, i.e. preinstalled apps.[362] For example, the *SYSTEM_UI_INTELLIGENCE* role is marked as "systemOnly".[363] This role is required to access certain ambient data features (cf. Section 9.4); therefore, only privileged system apps (cf. Section 5.2.5) can access those features.

(275) **Default application.** A certain combination of role attributes leads to the concept of a 'default application'. Concretely, a role that is both "exclusive" and "visible" results in the (single) app that holds that role to be considered the 'default' app.[364] If

---

357 See https://android.googlesource.com/platform/packages/modules/Permission/+/refs/heads/main/PermissionController/src/com/android/permissioncontroller/role/Role.md, Doc ID [...], accessed on 9 April 2026.

358 See https://android.googlesource.com/platform/packages/modules/Permission/+/refs/heads/main/PermissionController/src/com/android/permissioncontroller/role/RolePermissionProtection.md, Doc ID [...], accessed on 9 April 2026. In particular, when the protection Level is set to "internal role", which "enforces that only role may grant this permission to apps".

359 See https://android.googlesource.com/platform/packages/modules/Permission/+/refs/heads/main/PermissionController/src/com/android/permissioncontroller/role/Role.md, Doc ID [...], accessed on 9 April 2026.

360 See https://source.android.com/docs/core/permissions/android-roles, Doc ID [...], accessed on 9 April 2026, See https://developer.android.com/reference/android/app/role/RoleManager, Doc ID [...], accessed on 9 April 2026.

361 See https://android.googlesource.com/platform/packages/modules/Permission/+/refs/heads/main/PermissionController/src/com/android/permissioncontroller/role/Role.md, Doc ID [...], accessed on 9 April 2026.

362 See https://android.googlesource.com/platform/packages/modules/Permission/+/refs/heads/main/PermissionController/src/com/android/permissioncontroller/role/Role.md, Doc ID [...], accessed on 9 April 2026.

363 See https://cs.android.com/android/platform/superproject/+/android-latest-release:packages/modules/Permission/PermissionController/res/xml/roles.xml;l=886;bpv=0;bpt=0, Doc ID [...], accessed on 9 April 2026, lines 885-893.

364 See https://android.googlesource.com/platform/packages/modules/Permission/+/refs/heads/main/Permission

the role is not "requestable", users must go to settings to configure the 'default' app of a certain category. Starting from the settings screen, this requires six steps.[365] OEMs ship Google Android powered smart mobile devices with a configuration that sets which app(s) hold(s) each role.[366] This includes the assignment of apps as default.

(276) For instance, the Google Android *assistant role* is marked as "exclusive", meaning that only one app can hold that role at a time, as well as "visible",[367] making it configurable for the user to set the "default assistant". The assistant role is also not "requestable", meaning that the end user must go to system settings to select their default assistant, instead of being able to do so through a prompt within the assistant itself. Certain features are only available to the app that holds the default assistant role, such as reading centralised app data from AppSearch (cf. Section 9.2),[368] and access to those features is therefore limited to that one app.

(277) **System services.** AOSP defines certain "system services"[369] or "delegate APIs".[370] These system services mediate access to certain hardware and software features. The AOSP system services can be seen as placeholders ("holes"), which are left for separate components that provide concrete implementations ("plugs") of the logic for interacting with those hardware and software features.

(278) OEMs ship Google Android with a configuration that tells the OS for each system service ("hole") to which system app ("plug") it "binds".[371] This is called a "configuration overlay", as the OEM "overlays" the existing configuration, meaning that it fills in or replaces certain assignments, with one that has system apps assigned. Whenever the OS uses the functionality offered by that service, it uses this configuration to find and launch the relevant system app that handles that functionality and integrates with the relevant features. For example, LPH/LPNH contextual invocation (Section 8.2) is managed via a system service: the OEM configured the system service to point to the Google app, which then means that upon invocation, the OS will launch the Google app (which then offers Google's Circle to Search).

---

Controller/src/com/android/permissioncontroller/role/Role.md, Doc ID [...], accessed on 9 April 2026. Alphabet internal document, [Alphabet internal document], Doc ID […].

365 Selecting "*Apps*", "*Default apps*", "*Digital assistant app*", "*Default digital assistant app*", choosing the desired assistant app, and selecting "*OK*" to confirm. [Alphabet internal document], Doc ID […].

366 This is achieved through a configuration overlay or resource overlay. Alphabet's reply to AI RFI 6 of 23 February 2026, Doc ID [...], paragraph 177a. Such a configuration will set certain values for configuration resources that are referred to in the "defaultHolders" attribute of a role. See https://android.googlesource.com/platform/packages/modules/Permission/+/refs/heads/main/PermissionController/src/com/android/permissioncontroller/role/Role.md, Doc ID [...], accessed on 9 April 2026.

367 See https://cs.android.com/android/platform/superproject/+/android-latest-release:packages/modules/Permission/PermissionController/res/xml/roles.xml, Doc ID [...], accessed on 9 April 2026, line 104.

368 This is done by gating the feature with the *READ_ASSISTANT_APP_SEARCH_DATA* permission. See https://cs.android.com/android/platform/superproject/+/android-latest-release:packages/modules/Permission/PermissionController/res/xml/roles.xml , Doc ID [...], accessed on 9 April 2026, line 148.

369 The term "service" is to be read different from the concept of services in the DMA. See https://android.googlesource.com/platform/frameworks/base/+/master/services/core/java/com/android/server/SystemService.java, Doc ID [...], accessed on 9 April 2026.

370 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID [...], paragraph 182.

371 This is achieved through a configuration overlay or resource overlay. See https://source.android.com/docs/core/architecture/configuration, Doc ID [...]. Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID [...], paragraph 177a.

(279) This architecture of services imposes an important limitation: only one app can be bound to a system service at a time, restricting the possibility of accessing the hardware and software features mediated by that service to that one single app. This crucially prevents concurrent access to those features.

(280) **Access limitation.** Through roles, services, and default status, Alphabet currently limits the access to certain features of Google Android for third-party services. In practice, this prevents users from benefitting from the concurrent services, as key features are earmarked to only one service (e.g. an AI assistant) at a time. If an end user wishes to use different assistants for different purposes that rely on a feature subject to the default assistant role, they would have to change that role at each query, through a multi-step journey via settings. It would not be possible for the end user to use multiple assistants that have the same level of access (i.e. multi-homing): for example, the user might want to use one assistant for personal use (e.g. connected to a taxi app for booking a taxi with friends) and another assistant for professional use (e.g. connected to the user's work email). This is clearly a sub-optimal user experience, and the current friction associated with the change of assistant leads to a risk that users would typically use only the default assistant.

(281) A system based on a default role (or a system service) therefore limits feature access to a single app rather than allowing access for all business users, thereby leading to a strong advantage of the default app. This *winner-takes-all* approach to system permissions can easily lead to entrenchment and adversely impact competition, because it inherently favours out-of-the-box defaults and creates artificial penalties for users who decide to multi-home.[372] Specifically, end users may suffer from inertia to change the default app or they may perceive the default app as superior or more trustworthy simply due to being the default out of the box when they purchase the device. As Gemini is most of the times the default assistant out of the box (see Section 5.4.2), Gemini stands to gain the most out of this system based on a default role. Moreover, Gemini enjoys exclusive integration with many popular Alphabet apps and services (such as Gmail, Google Docs, Google Maps, YouTube, etc.), meaning that users who would like to multi-home might not do so if that requires removing Gemini's default status and thereby reducing Gemini's capabilities.

*5.2.5. Conditioning access to features on app preinstallation by the OEM*

(282) In addition to conditioning access to some features to default role, some Google Android features are only available to apps pre-installed by the OEMs on the mobile device.

(283) An OEM can choose to preinstall apps on their devices using different methods:

(a) **Normal app.** An OEM can preinstall an app as a normal app,[373] equivalent to user-downloaded apps, but it cannot obtain any permissions with the protection level "privileged";

(b) **System app.** If the OEM decides to preinstall an app as a system app, the app can access system APIs. A system app cannot be uninstalled by the user; and

---

372 Recitals 13, 27, and 59 DMA. See [academic] submission to the Public Consultation, Doc ID [...], Section I: "*The search cases found, on the basis of evidence Alphabet itself produced at trial, that this kind of default-driven exclusion is sufficient to maintain a monopoly*."

373 The app may be shipped with the device, or automatically downloaded and installed when the device is turned on the first time (Play Auto Install).

(c) **Privileged system app.** The OEM can grant a preinstalled system app permissions with the protection level "privileged". Such an app equally cannot be uninstalled by the user.

(284) The OEMs are free to install any app by any developer as a privileged system app, though Alphabet dissuades them from doing so for third-party apps (i.e. not Alphabet's own apps or OEM apps): "*[…] it is not suggested, but still common practice to install 3rd party apps as privileged.*"[374] For permissions that can only be granted by OEMs, such as privileged permissions, the OEM can freely grant those permissions to preinstalled system apps.[375]

(285) An OEM can also "pre-grant" a permission that would normally be user-controlled[376] to a preinstalled app. Effectively, this changes the permission from user "opt-in" to "opt-out", thereby denying the user the ability to explicitly consent to granting the permission. The user may still revoke the permissions unless the OEM further marks them as "fixed", in which case they are irrevocably granted to the app.[377]

(286) Contrary to *privileged permissions*, Alphabet is vigilant about OEM's practices in pre-granting *user-controlled* permissions, as the lack of explicit user consent bears the risk of being a "*violation of user trust.*"[378] Alphabet recommends it only for apps that are "*part of the platform*", offer "*the most basic phone functionality*" (such as the phone or SMS app), or support "*highly critical use cases, not to improve the user experience.*"[379]

(287) Alphabet requires to approve all cases where an OEM pre-grants a user-controlled permission to a preinstalled app.[380] Effectively, this undermines Alphabet's argument that OEMs must be responsible (or are a reliable arbiter) for device integrity, security, and privacy (Section 6.3.1). [Describes policies regarding Alphabet's commercial agreement with OEMs].[381] [Describes policies regarding Alphabet's commercial agreement with OEMs]. OEMs have no recourse, as otherwise they would not be certified as by Alphabet as a Google Android device and could thus not fully benefit from key products and services in the Google Android ecosystem, such

---

374 See https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/permission/Permissions.md, Doc ID [...], accessed on 23 March 2026.

375 [Alphabet internal document], Doc ID [...], Section 13.2.2. To grant an app a permission with the protection level privileged, in addition to pre-installation as a system app, the OEM must add the app to a *privileged permission allowlist*, which is at its discretion. Android Open See https://source.android.com/docs/core/permissions/perms-allowlist, Doc ID [...], accessed on 10 April 2026.

376 Being a normal, dangerous, or app-op permission.

377 See https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/permission/Permissions.md, Doc ID [...], accessed on 23 March 2026.

378 See https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/permission/Permissions.md, Doc ID [...], accessed on 23 March 2026.

379 See https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/permission/Permissions.md, Doc ID [...], accessed on 23 March 2026.

380 [Alphabet internal document], Doc ID […], Agreed minutes of meeting with [OEM] on 9 March 2026, Doc ID [...], paragraph 16.

381 [Alphabet internal document], Doc ID [...]. Agreed minutes of meeting with [OEM] on 09 March 2026, Doc ID [...], para 16-17.

as the Google Play Store, Google Play Services and other Google Mobile Services (GMS).[382]

(288) In summary, Alphabet decides unilaterally that certain features are implemented via system APIs inaccessible to third-party user-downloaded apps (Section 5.2.2), and/or that they are subject to privileged permissions, which the user is unable to grant to the third-party user-downloaded apps of their choice (Section 5.2.3). Alphabet also decides unilaterally to tie some features to system roles or system services that cannot be held by third-party user-downloaded apps, or to make those features only accessible by one app at a time via a default role or system services (Section 5.2.4). Alphabet delegates the choice of which apps can access such features to OEMs, through preinstallation and (pre-)granting of permissions (Section 5.2.5), or which unique app is selected as the default or for the system service. As explained in the rest of this Decision (see especially Section 5.4), Alphabet uses its privileged position as licensor of Google Android to gain access to these important Google Android features, while third parties rarely have access.

### 5.3. Implementation of Gemini and Google Assistant in Google Android

(289) Alphabet's assistants, and in particular its AI-powered assistant Gemini, are at the core of these proceedings, which seek to ensure that AI-powered third-party services and assistants have a degree of access to Google Android that is equal to the degree of access of Alphabet's assistants.

(290) At the time of this Decision, Alphabet's main AI assistant on Google Android is Gemini. Gemini, as well as its predecessor Google Assistant, employ a peculiar architecture where their core functionality is in fact embedded in the Google app.[383] Alphabet explains this concept as a "multi-tenant" app.[384] End users can still download an app for Gemini and Google Assistant from the Play Store, but this app does not offer any additional functionality. Instead, the downloaded app is merely a [description of Alphabet's internal processes and systems].[385]

(291) This implementation detail becomes visible to a user in the Google Android settings, when the user wants to change their default assistant. Here, the settings screen only shows the Google app. The Gemini or Google Assistant *shortcut apps* are not listed here, irrespective of whether if the user has installed them manually or not.

(292) The widespread pre-installation of the Google app therefore grants Alphabet's AI assistant an immediate access to a whole ecosystem of devices and enormous advantage in terms of both distribution and feature access.

(293) Gemini has more capabilities than Google Assistant and, importantly, uses large language models (LLMs) to comprehend user requests better and to plan multi-step actions performed in response. Gemini is still accessible through a voice interface, which can be activated by uttering the "Hey Google" hotword. In addition, Gemini

---

382 Android devices of OEMs must pass several test suites Google Mobile Services Test Suite (GTS), Compatibility Test Suite (CTS), Vendor Test Suite (VTS) to fulfil the GMS requirements and become certified. Only a certified device may licence Google apps. Alphabet's reply to RFI PO 17, Annex Q18.2, [Alphabet internal document], Doc ID [...], Section 3.6. Alphabet's reply to RFI AI 2 of 23 September 2025, Doc ID [...], DMA.100210, EMADA, Annex Q13.1, Doc ID [...], Section 2.

383 The Google app can be identified through the package name `com.google.android.googlequicksearchbox`. The app has a Google Play Store entry under this URL: https://play.google.com/store/apps/details?id=com.google.android.googlequicksearchbox, Doc ID [...], accessed on 8 July 2026.

384 Alphabet's response to RFI AI of 23 September 2025, Doc ID [...], question 33, paragraphs 80.

385 Alphabet's response to RFI AI 2 of 23 September 2025, Doc ID [...], question 33, paragraphs 80.

offers a chat-based interface that allows to write in a continued conversation and to pose follow-up questions.[386]

(294) The implementation of Gemini and Google Assistant as part of the Google app has the following implications:

(a) Gemini and Google Assistant are fully accessible by the user through the Google app, including invocation through hotwords and long pressing the power/side button. It is not necessary for the user to install the *shortcut app* on their device to use Gemini and Google Assistant;

(b) All privileged permissions that are available to the Google app are also available to Google Assistant, Gemini and vice-versa. For example, if the Google app has the permission to access the Android settings, Gemini enjoys the same degree of access;

(c) All OEM pre-granted permissions to the Google app are also available to Google Assistant, Gemini, and vice-versa. For example, if an OEM grants the Google app access to the user's contacts, Gemini can automatically access the user's contacts;

(d) All user granted permissions to the Google app are also available to Google Assistant and Gemini, and vice-versa. For example, if the user grants camera access to the Google app, to search for a picture they took, Gemini also gains access to the camera without additional user consent; and

(e) If an OEM has preinstalled the Google app, which is the case on all major OEM devices,[387] the functionality of Gemini and Google Assistant is automatically also installed and available out of the box.

(295) This has a particular consequence on the end user's ability to properly understand and control which features and data Gemini is accessing. The typical path would be for the end user to review the list of permissions in the "app info" section of settings for the Gemini app (Figure 1, right), in line with Android's model for platform security and privacy.[388] However, the information shown gives the end user the wrong impression that Gemini has *no* permissions, which is technically true as the Gemini app is only a "shortcut" app (as explained above), but practically it is incorrect. Instead, the relevant permissions of the Gemini *service* on Android must be consulted in the app info of the Google app (Figure 1, left), which hosts the actual Gemini functionality.

(296) The end user has no way to distinguish whether the permissions shown there are used by Gemini, Google Assistant, or a functionality in the Google app, and can therefore only assume that all the permissions of the Google app, to which the Gemini service has access, could be used by Gemini. Importantly, the end user therefore has no means to remove permissions specifically from the Gemini AI service: doing so could disable or break important functionality of the broader Google app, including Google search service. This further reinforces the opaque redistribution of the

386 See https://support.google.com/android/answer/14554984?hl=en_DE&co=GENIE.Platform%3DAndroid, Doc ID [...], accessed 26 March 2026.

387 Alphabet's reply to AI RFI 3 of 17 October 2025, [Alphabet commercial agreement], Annex Q1.6.1, Doc ID [...].

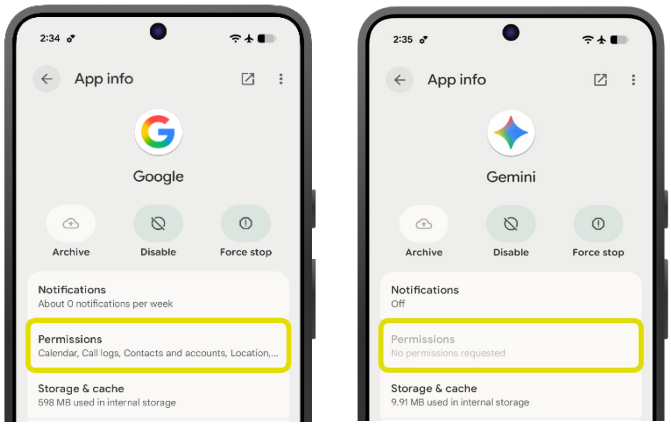
388 The Android Platform Security Model (2023), https://arxiv.org/pdf/1904.05572v3, Doc ID [...], accessed on 9 April 2026, Section 4.3.2 ("Android permissions").

Google app's broad permissions to Gemini, leading to broader feature access for Gemini and a loss of user agency.

(297) In the expert report annexed to Alphabet's response to the PFs, the expert explained that the security principle of "*least privilege states that privileges (i.e., the ability to access certain device functionality) granted to software should never be higher than necessary to perform the software's functionality.*" The implementation of Gemini inside the Google search app breaks this principle, as each of the services would have access to the permissions granted to the other service. Meaning that the Gemini has access to all permissions that are granted for any services in the Google app. Therefore, the Google app has more permissions than needed to provide the functionality of each of the services separately.[389]

(298) Another security principle in the same report is "secure defaults", which states *"that the default configuration of a system should reflect a restrictive and conservative enforcement of security policy.*" It is further evaluated as "*by default, a newly installed app does not receive dangerous runtime permissions automatically; instead, the user must grant those permissions when the app requests them in context.*" [390] The pre-granting of wide-ranging permissions to the Google app and in relation to the Gemini assistant also breaks the "secure defaults" principle. As the Google app receives access to sensitive permissions, including microphone, camera, and contacts, without the user granting those individually.

**Figure 1. Left Screenshot shows Google's app permissions.**
**Right screenshot shows permissions of the Gemini "shortcut" app.**




*Source: Screenshot taken on a Google Pixel 9 Pro*

## 5.4. OEMs have granted access to Gemini and its own AI assistants, but only rarely to third parties

(299) In the context of these proceedings, Alphabet argued that Google Android's access control system does not create unequal access between Alphabet's and third parties' apps. According to Alphabet, third parties are free to reach pre-installation agreements with OEMs to access any features subject to system privileges.[391] In this context, it is important to distinguish between pre-installation agreements, which potentially include other placement commitments,[392] and feature-access agreements which include (and require) pre-installation, as well as access to system privileges.

389 The expert report was written by [Academic]. See Annex 1 of Alphabet's response to the PFs, Doc ID [...], Section I.C, para 8.

390 See Annex 1 of Alphabet's response to the PFs, Doc ID [...], paragraphs 9, 28.

391 Alphabet submission of 17 March 2026, Doc ID [...], paragraph 18.

392 Such as setting an app as a default service, e.g., the default browser or assistant, or placing the app icon on the prominent Home screen.

Only the latter provide access within the meaning of Article 6(7) DMA to most of the features subject to this Decision.

(300) In practice, the conditioning of feature access on OEM approvals restricts, if not completely impedes, business users from offering feature-rich services to end users. Besides the efforts and delays associated with the negotiations necessary to obtain feature access, which must be concluded with each and every OEM, the evidence presented in this section shows that, even in the rare circumstances in which third parties have successfully concluded pre-installation agreements with OEMs, only in one case did a third party manage to obtain feature access.

(301) The Commission first notes that, despite its claims that third parties can simply conclude pre-installation agreements, Alphabet, in its role as OEM for its Pixel devices, has not granted any access to any competing AI service. To the Commission's knowledge, only one OEM, namely Samsung, has granted access only to one third-party provider of AI services, Perplexity, on some of its devices. This arrangement took place under very specific circumstances: [details on partnership] and Perplexity was able to provide Samsung's competing assistant Bixby with valuable inputs, including access to Perplexity's programmatic search APIs, [details on partnership].[393] [394]

(302) By virtue of being the Google Android provider, Alphabet has long-standing, deep, and mutually dependent relationships with the relevant OEMs, such as [list of OEMs],[395] which sometimes involves direct financial investments,[396] and facilitates the conclusion of pre-installation and feature-access arrangements. Alphabet has a direct line with all Android OEMs, including frequent meetings where it informs them of planned changes to the CDD and [Alphabet's confidential contractual terms].[397] By extension, Alphabet is also aware of [description of Alphabet's commercial relationship with OEMs] and is closely involved in the implementation of Google Android on OEMs' devices, including through the approval of such devices as Google Android certified devices, and the preinstallation of Google apps on these devices. Alphabet regularly negotiates [description of Alphabet's commercial relationship with OEMs] of agreements with OEMs, including amendments to existing agreements. It is a matter of fact, that this longstanding and mutually beneficial relationship cannot be matched by any other provider of AI services.

(303) Obliging third parties to develop these bilateral relationships is a time-consuming and complex requirement given that Google Android is implemented by many OEMs, with success ultimately far from guaranteed, as illustrated by third parties' actual experiences of engaging with OEMs, described further in Section 5.4.1. Alphabet is aware of the barrier created by the privilege system and the advantage it confers upon Alphabet. [description of Alphabet's internal systems and product design choices regarding system privilege][398]

(304) This concern is consistent with the feedback provided by an AI developer in its response to the Commission's Public Consultation: "*A competitor that must*

---

393 See https://www.samsungnext.com/blog/why-we-invested-in-perplexity, Doc ID [...], accessed on 10 April 2026.

394 Agreed minutes of meeting with [AI service provider] of [date], Doc ID [...], paragraphs 10-12 and [AI service provider] reply to RFI of 20 February2026, Doc ID [...], questions 1 and 2.

395 [Alphabet internal document], Doc ID […].

396 [Alphabet internal document], Doc ID […].

397 See, for instance, [Alphabet internal document], Doc ID […].

398 [Alphabet internal document], Doc ID […].

*negotiate each integration point separately, work around restricted APIs, or accept higher friction in user-facing interface is placed at a structural disadvantage and may ultimately be blocked from expanding in the sector*",[399] and is clearly evidenced by the large gap between the very limited instances where OEMs have concluded such agreements with third parties, versus the ecosystem-wide pre-installation with full feature access Google's Gemini.

*5.4.1. OEMs rarely grant access to third-party AI service providers*

(305) Alphabet, for its own Pixel devices, has not preinstalled any apps of third-party AI service providers nor [description of Alphabet's design choices regarding Pixel devices].[400]

(306) As regards third-party OEMs, since October 2023, several third-party AI providers, including [AI service provider],[401] [AI service provider],[402] and [AI service provider][403] have reached out to OEMs seeking pre-installation and/or access to some of the features concerned by this Decision.

(307) To date, only Perplexity has managed to reach a feature-access agreement with an OEM, namely Samsung. Specifically, Perplexity obtained access to hotwords [information on Perplexity and Samsung's partnership].[404] [further information on Samsung and Perplexity's collaboration which involves Samsung relying on Perplexity's functionalities for its Bixby assistant].[405] As referenced above, [information about the partnership], which may have been a further incentive for Samsung to support Perplexity and may not apply to other third parties.

(308) Perplexity also reached a pre-installation agreement with Motorola, but [information about the agreement] many features subject to these proceedings are inaccessible to Perplexity [information about the agreement].[406]

(309) Similarly, [AI service provider] engaged with many OEMs to secure pre-installation agreements but was often unable to conclude these agreements, including system level integration, attributing this outcome to OEM's assessment of contractual, technical, or commercial limitations with Google.[407] Even in circumstances in which it successfully reached pre-installation agreements with [OEM], [OEM] and [OEM] for its AI assistant [AI service provider] to be available out-of-the-box on their devices, these agreements required financially compensating OEMs[408] and [AI service provider] was unable to secure access to privileged APIs. Its AI assistant, [AI service provider], therefore does not enjoy access to the same system privileges as Gemini on these devices.

(310) [AI service provider], is not currently preinstalled on any Android mobile devices. As a result, [AI service provider] has not met the first condition necessary to access to the crucial features subject to these proceedings. [AI service provider] explained

399 Response to the Commission's public consultation, Doc ID [...], paragraph 8.
400 Alphabet's reply to RFI PO 19 of 17 April 2026, Doc ID [...], [Description of Alphabet's commercial negotiation with third parties].
401 [AI service provider]reply to RFI 1 of 20 February 2026, Doc ID [...], questions 1 and 3.
402 [AI service provider] reply to RFI 1 of 20 February 2026, Doc ID [...], questions 3
403 [AI service provider]reply to RFI 1 of 20 February 2026, Doc ID [...], question 3.
404 [AI service provider's] reply to RFI 1 of 20 February 2026, Doc ID […], questions 1 and 3.
405 Agreed minutes of meeting with [AI service provider] of [date], Doc ID […], paragraphs 10-12 and [AI service provider's] response to RFI dated 20 February 2026, Doc ID […], question 2.
406 Agreed minutes of meeting with [AI service provider] of [date], Doc ID [...], paragraph 2.
407 [AI service provider]reply to RFI 1 of 20 February 2026, Doc ID [...], questions 1 and 3.
408 Agreed minutes of meeting with [AI service provider] of 8 December 2025, Doc ID [...], paragraph 19.

that it had held discussions with various OEMs regarding the pre-installation of [AI service provider], but no agreements could be reached. [AI service provider] submitted that various OEMs indicated that contractual constraints imposed by Alphabet prevent them from entering into potential partnerships. [AI service provider] also stated that even if it achieved pre-installation with an OEM, the OEM may not grant "system app" permissions so that [AI service provider] would still not obtain access to permission-guarded features.[409]

(311) Similarly, [AI service provider], another established third-party AI provider of a successful AI assistant [AI service] did not have its assistant pre-installed on any Android mobile devices as of March 2026. As a result, [AI service provider] explained that it lacks access to these features[410] and that the policy of reserving permissions to preinstalled apps puts downloaded apps at a competitive disadvantage.[411]

(312) Another established third-party AI provider did not reach any pre-installation agreements, let alone a feature-access agreement, with OEMs, despite having engaged in discussions with various OEMs about integrating its AI assistant on their devices between late 2024 and mid-2025.[412] Its AI assistant is therefore neither pre-installed nor granted the required permissions for feature access.

(313) The above-mentioned companies are amongst the main challengers in the AI space. Smaller AI service providers would evidently face even larger difficulties to obtain feature access from OEMs, since they lack (i) the resources to negotiate pre-installation agreements with major OEMs, particularly any involving access to system privileges, and (ii) the financial ability to pay for such arrangements. Furthermore, unlike Perplexity, they do not typically have other services or inducements to offer to incentivise OEMs to preinstall their apps and grant system privileges.

(314) For example, [AI service provider] stated that there are too many OEMs and negotiating feature access with each is lengthy and complex. [AI service provider] does not have the resources to commit to this. Moreover, each OEM has its own financial incentives and potentially competing services. [AI service provider] considers it very unlikely to obtain feature access from an OEM which is financially incentivised to boost the success of Alphabet's AI assistant or promote its own AI assistant.[413]

(315) Similarly, [AI service provider] described a system whereby the OEM decides which apps or services obtain access to important functionality, such as device access points or hotwords, as problematic. Negotiating access with each individual OEMs is unrealistic for a company such as [AI service provider] because it is too resource-intensive, costly and, in [AI service provider's]'s view, it should be the user who determines access points configuration. [AI service provider] does not have any relationships with OEMs and stated that OEMs may refuse access for a variety of reasons.[414]

(316) The above outlined experiences of AI service providers are also reflected in OEMs' submissions made in the context of these proceedings.

---

409 [AI service provider] reply to RFI 1 of 20 February 2026, Doc ID [...], paragraph 3.2.

410 [AI service provider] reply to RFI 1 of 20 February 2026, Doc ID [...], questions 1 and 3.

411 Agreed minutes of meeting with [AI service provider] of 20 March 2026, Doc ID [...], paragraph. 10.

412 Third party's reply to RFI 1 of 17 February 2026, Doc ID [...], paragraphs 1-3.

413 Agreed minutes of meeting with [AI service provider] of 2 March 2026, Doc ID [...], paragraphs 16-17.

414 Agreed minutes of meeting with [AI service provider] of 2 April 2026, Doc ID [...], paragraph 3.

(317) For example, [OEM] explained that, while [information about relationship between OEM and AI players] with some AI players, it did not consider granting them system privileges [415] and that its *"primary driver was a commercial discussion, not a technical discussion."*[416]

(318) [OEM] stated that, on its devices, it preinstalls only one AI assistant app, namely Alphabet's Gemini.[417] [OEM] explained that, while it also pre-installs [AI service provider] (powered by [AI service provider] and other AI apps) on certain mobile devices, these devices are produced for [third party] (upon request by [third party])[418] and [AI service provider] (including [AI service provider]) does not have access to privileged APIs.[419] Therefore, no other AI assistant has access to the permission-guarded features subject to these proceedings on devices produced by [OEM]. [OEM] explained that it is in "commercial discussions" with several AI app developers regarding the pre-installation of their AI services. The determining factors as to whether [OEM] will preinstall their apps are *"[c]ommercial terms, user experience, and AI abilities*"[420] (i.e., not integrity considerations).

(319) [OEM] explained that it engaged with several AI app developers with whom it did not reach agreements and faced obstacles due to timing and technical relationship building.[421] [OEM]'s draft term sheets required third parties to make payments to [OEM], including revenue share requirements (e.g., a percentage of the third-party AI service provider's revenues realised on the [OEM] device). Furthermore, third-party AI service providers would have to meet other requirements, such as sharing their product roadmap on a quarterly basis, providing a free offering to certain customers for a certain period of time, and committing to a parity clause. Another clause provides that the parties shall mutually agree upon optimizations or the development of AI related services intended for [OEM]'s implementation of Android, [OEM interface], and [OEM] first party apps.[422]

(320) In summary, the submissions from third-party AI service providers and OEMs show that pre-installation, which acts as a pre-condition to the granting of system privileges, can only be achieved in exchange for financial and/or non-financial consideration. Furthermore, even when OEMs agree to preinstall third-party AI assistants, they may not grant access to privileged features. Moreover, these submissions show that several OEMs did not grant feature access for reasons unrelated to the integrity of the device or the operating system, but rather for other reasons such as "commercial" considerations, i.e. relating to the commercial terms of the agreements.

(321) Furthermore, it is not excluded that the difficulties of AI service providers to reach pre-installation or feature-access agreements with OEMs may also stem from the fact that some OEMs have their own AI assistant preinstalled on their devices, such as Samsung's Bixby. Samsung pre-installs its own AI assistant Bixby on recent Samsung mobile devices which integrate with other apps installed on users' devices.[423] This dynamic could further deter OEMs from entering into pre-

415 [OEM] reply to RFI 1 of 05 February 2026, Doc ID [...], questions 6 and 7
416 [OEM] reply to RFI 1 of 05 February 2026, Doc ID [...], question 7.
417 [OEM] reply to RFI 1 of 05 February 2026, Doc ID [...], question 22.
418 [OEM] reply to RFI 1 of 05 February 2026, Doc ID [...], question 5.
419 [Third party's] reply to RFI 1 of 5 February 2026, Doc ID [...], response to question 13.
420 [OEM] reply to RFI 1 of 05 February 2026, Doc ID [...], question 7.
421 [OEM]reply to RFI 2, Doc ID [...], question 3.
422 [OEM]"[Description of the commercial agreement]" submitted on 24 March 2026, Doc ID [...].
423 See https://www.samsung.com/us/explore/bixby/apps/, Doc ID [...], accessed on 23 April 2026.

installation and feature-access agreements with potential rivals, or at least amplify the financial compensation demanded to acquiesce to such a request. As explained above, Samsung granted feature access to Perplexity in exchange for valuable Perplexity functionality which is used by Samsung's Bixby.

(322) Therefore, Alphabet's claim that third-party AI service providers can reach pre-installation or feature-access agreements with the OEMs fails to account for the various hurdles faced by third-party AI service providers. Indeed, the evidence on the file in the present proceedings shows that, despite the best efforts of many third-party AI service providers, there are a range of factors that influence and negatively impact the ability of third-party AI service providers to secure pre-installation agreements on OEMs' devices, let alone obtain access to features guarded by system privileges.

*5.4.2. Gemini is pre-installed by OEMs with relevant feature access*

(323) The near complete failure of OEMs to grant feature access to third-party AI service providers – both large and small – contrasts strongly with the level of access enjoyed by Alphabet's AI services. The Google app, which contains the Google Search service and Gemini, is preinstalled on all major OEM devices: representing [90-100%] of all Google Android software builds of smartphones that are active in the EU.[424]

(324) Alphabet's claim that others can simply conclude pre-installation or feature-access agreements with OEMs should also be put in perspective with the level of investment associated with these agreements. Indeed, Alphabet pays OEMs [describes Alphabet's commercial agreements with OEMs] through commercial agreements in the EEA that include obligations on OEMs to pre-install certain Google Applications (see Section 5.1.3), illustrating the scale of investment required to secure these agreements.

(325) In addition, OEMs also set Gemini as the default assistant on almost all Google Android mobile devices. For instance, under the Gemini Commercial Agreement, Alphabet pays [OEM] a fixed sum [contractually agreed amount][description of Alphabet's commercial agreement with OEMs].[425]

(326) As a result, it is clear that Google Android's access control system creates substantial barriers for third parties wishing to secure system privileges and, in turn, access to several features that fall within the scope of these proceedings. These barriers are not merely hypothetical but borne out by the actual experiences of third parties that engaged in commercial negotiations with OEMs, as well as the scale of investment undertaken by Alphabet to secure, close to, sole access to these features. This system therefore hinders both contestability – in particular, contestability of Gemini – and fairness, as this system results in discrimination against some third-party services and favours others by allowing OEMs to selectively restrict or grant interoperability.

---

424 Alphabet's reply to RFI PO 6 of 23 February 2026, Annex Q2, Doc ID [...]. [Description of Alphabet's internal data]

425 Alphabet's reply to RFI AI 4 of 28 November 2025, Doc ID [...], question 4.

## 6. ALPHABET'S OBLIGATIONS UNDER ARTICLE 6(7) DMA

### 6.1. Alphabet must take all necessary measures to ensure effective interoperability with all Google Android powered smart mobile devices

*6.1.1. Taking all necessary measures in practice*

(327) As explained in Section 4.6 above, Alphabet, as the designated gatekeeper for the core platform service Google Android, remains responsible for ensuring that effective interoperability with Google Android and access to Google Android's hardware and software features are available to app developers throughout the entire Android ecosystem as required by Article 6(7) DMA.

(328) For mobile devices supplied by third parties powered by Google Android, this means that Alphabet must take all actions of legal, contractual and technical nature, necessary to ensure that the implementation of Google Android on such devices complies with Article 6(7) DMA. For instance, Alphabet has the right to deny approval of a third-party Google Android device – meaning that the third-party OEM is contractually prohibited from marketing such device as an Android device – if this device does not implement detailed measures to ensuring compliance with certain regulatory obligations under the DMA. The Commission notes that, to date, Alphabet has not put in place similar provisions or taken measures to ensure compliance specifically with its interoperability obligations across the Google Android ecosystem.

(329) Alphabet already tightly regulates the manner in which third-party OEMs implement Google Android on their handsets by virtue of its control over the operating system and its distribution.

(330) As described in Section 5.1.1 of this Decision, Alphabet imposes detailed requirements on OEMs in the CDD and other licensing arrangements which OEMs must accept in order to be allowed to use Google Android and be awarded the status of Android compatible device. These requirements concern *inter alia* the device hardware, software features, permission model, and the implementation of security or privacy safeguards.[426]

(331) In terms of enforcement and monitoring, third-party OEMs must submit the software build of each Google Android device to Alphabet for its approval. If the software build (which includes the operating system) does not conform to Alphabet's requirements, Alphabet has the right to deny approval of the device. Alphabet therefore has contractual means to verify and enforce compliance with its rules.

(332) Alphabet already today uses its control over the distribution of Google Android to ensure compliance with the DMA. It does so by imposing on OEMs via the GMS Requirements that their Google Android mobile devices adopt certain measures taken by Alphabet to comply with the DMA.[427] More specifically, if an OEM preloads the Google app, the OEM "*MUST implement the choice screen*", as mandated by Article 6(3) DMA.[428] [Description of Alphabet's commercial agreement].[429] To comply with the DMA, Alphabet has not only provided the necessary contractual terms in various agreements but also employed technical

---

426 For instance, [OEM]reply to RFI of 5 February 2026, Doc ID [...], question 4, and [OEM]reply to RFI 1 of 5 February 2026, Doc ID [...], question 4.

427 [Description of Alphabet's commercial agreement], see Alphabet 's reply to RFI AI 2 of 23 September 2025, Doc ID [...].

428 Alphabet 's reply to RFI AI 2 of 23 September 2025, Doc ID [...].

429 [Description of Alphabet's commercial agreement]. See [Alphabet internal document], doc ID […].

solutions which OEMs pre-install on their smart mobile devices to ensure that Google Android, as deployed on OEMs' devices, is implemented in compliance with the DMA.[430]

(333) Alphabet submitted that it does not currently have contractual means to require third-party OEMs to roll out update on existing devices (which Alphabet approved in the past – so-called legacy devices).[431] To ensure that the operating system on these devices complies with Article 6(7) DMA or that the devices comply with the DMA more broadly, Alphabet may have to request in contractual negotiations with OEMs that the latter must roll out such updates.[432] Without such contractual right, Alphabet may not be able to bring legacy devices of third-party OEMs, which by definition account for a large part of total Google Android mobile devices, into compliance with that Regulation. To ensure compliance with Article 6(7) DMA on legacy devices, Alphabet may have to invoke and enforce existing contractual rights.[433]

*6.1.2. No disproportionate impact on third-party OEMs*

(334) Alphabet submits that the Commission's approach regarding third-party OEM devices unduly compromises OEMs' business autonomy. In particular, Alphabet contends that this approach would require Alphabet to tightly control the behaviour of OEMs, curtailing their ability to innovate and control device design, as well as unnecessarily imposing costs on OEMs and capping their ability and incentive to innovate.[434] Several OEMs raised similar concerns.[435] These are however not warranted for the following reasons.

(335) *First,* any autonomy which OEMs enjoy today finds its limit in applicable EU legislation, in this case Article 6(7) DMA. As explained in Section 4.6, that provision also applies to third-party OEM devices to the extent third-party OEMs implement on their device a product developed by a third party, which is subject to a specific regulatory framework.

(336) *Second,* it is unclear, based on Alphabet's generic claims, what degree of "autonomy" should in Alphabet's view be permissible within the framework of Article 6(7) DMA. Alphabet does not single out any particular feature or measure which would affect OEMs interests to a degree which would require more permissive interpretation of Article 6(7) DMA.

(337) The Commission considers that the measures specified in this Decision do not disproportionately affect third-party OEMs for the following reasons.

(338) *First,* the Commission considers that OEMs will retain significant autonomy over device customisation, including as regards device security and functionality, allowing them to innovate and compete as set out below. Alphabet's compliance with Article 6(7) DMA does not curtail OEMs' ability to differentiate, innovate and compete on the basis of their Google Android mobile devices, such as through (i) pre-installing certain apps, setting default services, offering innovative complementary software and user interface (UI) layouts, as well as on the basis of

430 [Description of Alphabet's commercial agreement] [Alphabet internal document], Doc ID […].

431 Alphabet's reply to RFI PO 21, Doc ID [...], question 1.

432 [Description of Alphabet's commercial agreement], see Alphabet's reply to RFI AI 2 of 23 September 2025, Doc ID [...], [terms of Alphabet's commercial agreement].

433 Alphabet's reply to RFI AI 2 of 23 September 2025, Doc ID […], EMADA, Annex Q13.1, Doc ID […], provides that [terms of commercial agreements].

434 Alphabet's response to the PFs, Doc ID [...], paragraphs 128-133.

435 See for instance [OEM] submission to the Public Consultation, Doc ID [...], [OEM] submission to the Public Consultation Doc ID [...] or [OEM] submission to the Public Consultation, Doc ID [...].

hardware features (e.g. camera, battery, memory space, etc) or (ii) providing attractive experiences for users and maintaining branding and product feel.[436]

(339) *Second,* OEMs will also continue to be able to effectively monetise their investments to end users (e.g. via device sales) and app developers (e.g. for pre-installation and default-setting, including with respect to apps which can generate continuous revenues for OEMs such as app stores).[437]

(340) *Third,* as explained above in this Section, the Commission's approach regarding third-party OEM devices is consistent with how Alphabet distributes Google Android today. While the implementation of the measures specified in this Decision may have as consequence some changes to the requirements which Alphabet imposes on third-party OEMs, Alphabet already now regulates OEMs' use of Google Android through agreements which impose a multitude of requirements which are regularly updated.

(341) *Fourth,* while the measures specified in this Decision may impose on OEMs certain additional implementation costs, they regularly have to invest in order to implement Google Android on their handsets.[438] In particular, the measures do not require OEMs to install additional hardware.[439] Alphabet, through modifications of the source code of Google Android and the contractual framework including the GMS Requirements, regularly imposes costs on OEMs [description of commercial agreement].[440] Importantly, OEMs have no contractual rights protecting them from changes to Google Android.

(342) *Fifth,* the measures specified in this Decision will not compromise OEMs' reputation and competitiveness by weakening their authority over device and partner management:[441] As further explained in paragraph (409) below, Alphabet already exerts control on integrity, security, and privacy matters, over OEMs via the EMADA,[442] GMS Requirements[443] and the CDD.[444] As explained below, this is perhaps best exemplified by the Mobile Bundled Apps (MBA) policy (which is part of the GMS Requirements), where Alphabet unilaterally imposes integrity, security, and privacy requirements on OEMs for any app the OEM seeks to pre-install.[445] OEMs submitted that they are subject to these requirements.[446]

---

436 Alphabet submission of 17 March 2026, Doc ID [...], paragraphs 20-21, 25-28.

437 Alphabet submission of 17 March 2026, Doc ID [...], paragraph 22.

438 In particular, while OEMs highlighted in their submissions to the Public Consultation that the measures would require integration efforts on their part, none of them identified engineering efforts or investments of a different nature than what they already need to engage in to implement and distribute each year a new version of Google Android on their handsets (see for instance [OEM's] submission to the Public Consultation Doc ID [...]).

439 Contrary to what some of the OEMs seemed to believe based on their responses to the Public Consultation (see for instance [OEM's] submission Doc ID [...]).

440 [Alphabet internal document], Doc ID […], [Direct quote describing Alphabet discussions regarding OEM compliance with commercial agreements].

441 Alphabet's response to the PFs, Doc ID [...], paragraph 132.

442 [Description of Alphabet's commercial agreement regarding security requirements], see Alphabet's reply to RFI AI 2 of 23 September 2025, Doc ID [...], [Terms of commercial agreement regarding security requirements].

443 [Description of Alphabet's commercial agreement regarding security requirements].

444 [Description of Alphabet's commercial agreement regarding security requirements] Doc ID [...].

445 [Description of terms of Alphabet's commercial agreements with OEMs regarding security requirements].

446 For instance, [OEM] reply to RFI 1 of 5 March 2026, Doc ID [...], question 4; [OEM] reply to RFI 1 of 5 March 2026, Doc ID [...].

(343) Finally, the Commission notes that Alphabet does not claim that the measures would interfere with OEMs' rights, nor has any OEM made such claim in the public consultation.

## 6.2. The scope of the designated Google Android operating system

(344) The Designation Decision designates Google Android on mobile devices smartphones and tablets, as an operating system within the meaning of Article 2(10) DMA.

(345) The Designation Decision designates Alphabet's middleware as part of Google Android insofar as it contributes to controlling the basic functions of Android tablets and smartphones and to enabling software applications to run on them, thereby ensuring an effective functioning of the operating system.[447]

(346) As explained above (Section 5.1.1), AOSP, including its source code, is the foundational part of Google Android and therefore part of the designated CPS.

(347) In its response to the PFs, Alphabet submitted that AICore/ Gemini Nano ODMs, Android/Pixel System Intelligence and the Google Hotword Engine are not part of Google Android because they do not control basic functions of the hardware or software and do not enable software applications to run on them.[448]

(348) According to Alphabet, the definition of 'operating system' in Article 2(10) DMA should be interpreted to only capture *"features which (i) are 'required for the operation' of the device; or (ii) control management of critical device resources e.g., 'processing, memory and storage'"*, and requires that the features are part of "system software" and enable software applications to run on it ("the mediating role of the OS").[449]

### *6.2.1. Interpretation of Article 2(10)*

(349) The Commission considers that Alphabet's interpretation of Article 2(10) DMA is misplaced. The interpretation of Article 2(10) should be guided by the operating system's intermediary role as a platform for third-party software and hardware.

(350) *First,* the DMA is concerned with the operating system as "the heart of devices and ecosystems",[450] because it is a gateway for business users to reach end users whenever those end users use a smart device that can run apps and interact with other hardware. Operating as a platform is the primary purpose of the operating systems which power the devices that most people use every day – most notably smartphones and computers.

(351) The operating system makes the device hardware usable by managing resources and providing a platform on which software applications can run. It does so by making functionalities centrally accessible to all software applications and users. As software applications may rely on these functionalities to offer services to end users, the operating system is a gateway. Accordingly, Article 6(7) DMA seeks to establish a level playing field for business users (who rely on designated operated system as a gateway to reach end users) as regards the features of the "same operating system" as is available or used by the gatekeeper.[451]

---

447 Designation Decision, Section 5.7.

448 Alphabet's response to the PFs, paragraphs 160-161, see also paragraphs 157-159.

449 Alphabet's response to the PFs, paragraph 158 (fn. omitted).

450 Impact Assessment 1/2, p. 115.

451 The travaux préparatoires reflect this concern about the operating system as a gateway with respect to, in principle, all its functionalities. The Impact Assessment conveys the concern about a "provider of

(352) The use of the term "system software" in Article 2(10) DMA therefore encompasses the operating system as a unified entity. The system software may consist of thousands of functionalities itself and may control numerous functions outside the system software, such as hardware and software that is located outside of the system software. The definition in Article 2(10) DMA does not mean that only basic functions are part of the operating system nor that only functionalities that software applications on the device cannot run without[452] are covered: these requirements are defining characteristics of an operating system based on what it *does*, but they do not delineate its boundaries.[453]

(353) Moreover, Article 6(7) DMA applies to features which are *accessed or controlled via the operating system*. It therefore focusses on the intermediary function of the operating system, not on where the implementation of the features is located. Recital 56 clarifies that "any technical functionality" of a device is in scope of Article 6(7) DMA. Article 2(10) of that same Regulation should be interpreted in this light as Alphabet's interpretation excludes features which the legislator specifically sought to include, such as near-field-communication (NFC) technology and secure elements,[454] which may not be "basic" in the sense that they are required for the operation of the device or a critical device resource. It would also result in the paradoxical situation that "basic" features must be made interoperable (which typically are already interoperable), while advanced features, including new hardware, are not in scope even though they are used by the gatekeeper to provide important services and can be particularly important for contestability.[455] Article 6(7) DMA should therefore not be interpreted in isolation but in light of the substantive obligations.

(354) In particular, the operating system cannot be limited to its narrow core that manages critical resources or simply keeps the device operational, such as the kernel, file system functionality, device drivers, the networking stack and security and user management. Such foundational functionality of an operating system is several layers away from the features which are exposed by the operating system to software applications, and which are most relevant for the operating system's role as a gateway. Today's operating systems are feature-rich to satisfy user demand and to enable innovative apps to run on them.[456] The use of the term "system software"

---

operating systems giving differentiated conditions of access to its operating systems or device features" which "prevents third parties from competing in a level playing field" and "limiting the access to or the interoperability of its operating system and respective functionalities (e.g. NFC) with the services offered by business users, reserving those functionalities to their own services." (IA 1/2, p. 42-43.) Further, it refers to the unfair practice of preventing interoperability "with the same operating system, hardware or software features that are available to or used by [..] the gatekeeper" to prevent gatekeepers from discriminating "in terms of access to their operating system, hardware or software features". (IA 2/2, p. 56).

452 Alphabet's response to the PFs, Doc ID [...], paragraph 164, last bullet.

453 In addition, the definition of active business users of an operating system in Annex 1 of the DMA confirms that an operating system is not limited to some basic functionality. Relevant business users are developers which offer a software application "using the programming language or any software development tools of, or running in any way on, the operating system". Relevant is not the use of certain "basic" or "critical" features, but the use of the operating system as a whole. See Annex 1 of the DMA, section E.

454 Recital 56 DMA.

455 For example, Apple launched the near-field communication chip in September 2014 followed only one month later by the launch of NFC-based payments via Apple Pay in the US. Apple opened the NFC chip for payment uses in late 2024 after a Commission commitments decision.

456 See e.g. https://www.ebsco.com/research-starters/computer-science/mobile-operating-systems, Doc ID [...], accessed on 10 July 2026 and https://www.walturn.com/insights/traditional-os-to-ai-os-the-evolution-of-operating-systems, Doc ID [...], accessed on 10 July 2026.

connotes a wide understanding in the sense of *"a collection of software that manages computer hardware resources and provides common services for computer programs"*[457] and *"a program that runs on a computer and provides a software platform on which other programs can run."*[458] The gateway perspective is well reflected in the following description: *"the operating system is the software that makes your computer come alive. It manages your apps, hardware, files, and security, shaping the experience every time you sit down at your device. Every click, tap, and scroll travels through the OS, turning your input into something that feels fast, smooth, and natural."*[459]

(355) Similarly, limiting the scope to features without which a software application could not run[460] could exclude important operating system functionality critical for modern software applications such as for instance notifications, audio/video, input (keyboard, mouse, etc.), text services (spell check, autocorrect), background tasks, networking (internet, Wi-Fi), location/sensors (GPS), clipboard/share sheet (copy/paste and sharing between apps), and hardware access to execute machine learning models.

(356) *Second*, a requirement for the Commission to assess for each function of the hardware, software and the operating system whether they are "basic features" and whether software applications can run without them is incompatible with the legislative goal to adopt clear and self-executing rules which can be complied without *"an extensive investigation of often very complex facts on a case by case basis"*.[461] In particular, Alphabet's interpretation would undo the legislative intent to adopt a clear-cut rule.

(357) *Third,* Alphabet's proposed interpretation of Article 2(10) is inconsistent with its own submissions. In these proceedings Alphabet accepts the gateway-centric approach with respect to features which are part of AOSP. Nowhere does Alphabet conduct or propose an assessment of whether the features controlled by AOSP are "basic" or indispensable for software applications. In fact, Alphabet arbitrarily decides whether to implement certain features in AOSP rather than in the middleware (see Section 5.1.2). Recital 14 DMA makes it clear that the definition of a core platform service in the DMA should be technology neutral[462] and should not depend on the technological choices made by the gatekeeper, such as deciding in which software component to implement a certain feature.[463]

(358) Nor did Alphabet propose such an assessment in the designation phase, in the context of which it submitted that the Google Android operating system should be designated as a whole, i.e., including all features it offers,[464] stating that the AOSP code [direct quote describing Alphabet's internal views].[465] [Description of Alphabet's internal

---

457 Elaine Barker, Miles Smid, and Dennis Branstad, *NIST Special Publication 800-152: A Profile for U.S. Federal Cryptographic Key Management Systems*, October 2015, https://doi.org/10.6028/NIST.SP.800-152, Doc ID [...].

458 Karen Kent, Suzanne Chevalier, Tim Grance, and Hung Dang, NIST Special Publication 800-86: Guide to Integrating Forensic Techniques into Incident Response: Recommendations of the National Institute of Standards and Technology, August 2006, https://doi.org/10.6028/NIST.SP.800-86, Doc ID [...].

459 See https://www.microsoft.com/en-us/windows/learning-center/what-is-an-operating-system, Doc ID [...], accessed on 8 July 2026.

460 Alphabet's response to the PFs, paragraph 164, last bullet.

461 See e.g., recital 5 DMA.

462 Recital 14 DMA states that *"the definition of core platform services should be technology neutral and should be understood to encompass those provided on or through various means or devices"*.

463 See judgment of 8 July 2026, Apple v Commission, T-1079/23, ECLI:EU:T:2026:451, paragraph 80.

464 [Direct quote of EC engagement]

465 [Direct quote of EC engagement]

views].[466] Accordingly, the Commission designated Google Android as a unified entity, not only specific functionalities of that operating system under the DMA (and designated Microsoft Windows and Apple iOS and iPadOS based on the same approach). Alphabet has not challenged the designation decision.

(359) Finally, Alphabet's internal documents show that Alphabet considers the operating system and system software to be a broader concept than its proposed interpretation of Article 2(10) DMA. They show that Alphabet views system services [Direct quote of Alphabet internal document]*"*.[467]

(360) As a result, Alphabet's attempt to narrow the definition of the operating system for the purpose of the DMA cannot stand.

*6.2.2. Designated middleware*

(361) The definition of Article 2(10) DMA is relevant to assess Alphabet's claim that certain parts of its middleware, namely ODMs and the Google Hotword Engine, are not part of Google Android. A gatekeeper cannot avoid the application of Article 6(7) DMA by artificially splitting the operating system into several parts and by misapplying a legal test which relates to the totality of the operating system, not each component. The framework of the DMA distinguishes between the operating system and software applications running on top of it.[468] Middleware, which is normally system software,[469] is not mentioned in that Regulation.

(362) The Designation Decision reflects the fact that middleware is system software which typically performs functions of the operating system. It therefore designates Alphabet's middleware insofar as it contributes to controlling basic functions of Google Android tablets and smartphones and contributes to enabling software applications to run on them, thereby ensuring an effective functioning of the operating system.[470]

(363) Such a contribution is likely to exist in particular where middleware makes functionality accessible to software applications in a central and reliable way on all relevant devices. In such a case, middleware functions in the same way, and serves the same purpose, as the core of the operating system by enabling software applications to run and to make use of the functionality it exposes to them. In particular, middleware which offers functionality which are improved versions of AOSP features de facto constitutes an extension of AOSP features, as in this case the middleware operates as an extension of the operating system.[471]

(364) One fundamental (and indeed basic) function of the operating system is to provide the environment for apps to function and for users to interact with such apps. The operating system does so by exposing and documenting thousands of public APIs in a central and easily accessible manner which all apps can use. Without such APIs, no app could operate on a device, as these APIs mediate access to the device software and hardware, including the processor, memory and storage. Operating systems commit to maintain these APIs in updated versions of the operating system and only deprecate them in a predictable way. This commitment and predictability are critical

---

466 Final Form GD, 3 July 2023, paragraph 134.

467 [Alphabet internal document], Doc ID […].

468 Article 2(10) and Article 2(15) DMA.

469 See Section 5.1.2.

470 Designation Decision, Section 5.7.

471 See Commission Decision of 17 December 2020 in Case M.9660 – Google/Fitbit where market participants were concerned that "that Google might 'duplicate' APIs in GMS and innovate those APIs instead of AOSP APIs" (see paragraph 991).

for developers: if APIs disappear or change, their apps will not work at all ("break") or not as intended in terms of their functionality. The availability, stability and consistency of APIs have been referred to as "API contract". Alphabet publishes detailed "Android API guidelines" as *"a guide for developers to understand the general principles that the API Council enforces in API reviews."*[472] The purpose of these APIs is to enable software applications to run on the operating system.

(365) These principles equally apply to APIs exposed by Alphabet's middleware which is system software and exposed to all apps and which developers are encouraged to rely on (see Section 5.1.2 with respect to Google Play Services).[473] Similarly, Android APIs that are used by Alphabet's apps (but not made accessible to third parties) and enable apps to run on the operating system can be part of the operating system as withholding interoperability is liable to prevent a level-playing field and undermine contestability and fairness. It is not the role of system software to offer functionality only to the apps of a single developer.

(366) The Commission assesses Alphabet's claim that two features are not part of the designated Google Android operating system against this background in the subsequent sections.

6.2.2.1. AICore and the Gemini Nano ODMs are part of the designated Android middleware

(367) Alphabet submitted that AICore and Gemini Nano ODMs are not part of Google Android because they do not coYupntrol basic functions of the hardware or software and do not enable software applications to run on them.[474]

(368) AICore and the Gemini Nano ODMs are relevant for the system-level ODMs feature (Section 11.2). As explained in detail in Section 11.2, Alphabet decided to grant third parties access to this feature and claims that there is no interoperability gap.

(369) Alphabet claims that it does not need to grant third parties access to this feature because it is not part of the designated operating system.

(370) Alphabet's claim in their response to the PFs is inconsistent with Alphabet's public representations. Alphabet considers that AICore/Gemini Nano ODM are an integral part of the Google Android operating system today and going forward. In the developer information Alphabet states that the Gemini Nano ODMs are "*the foundation model on Android"*,[475] that *"Gemini Nano runs in the Android's AICore system service"*,[476] and *"AICore enables the Android OS to provide and manage AI foundation models"*[477] or that Gemini Nano ODMs are *"directly integrated to the*

---

472 See https://source.android.com/docs/setup/contribute/api-guidelines, Doc ID [...], accessed on 8 July 2026.

473 The merger decisions cited by Alphabet support this reading. In Case COMP/M.6381, Google/Motorola Mobility, paras 19, 21 (cited in Alphabet's response to the PFs, Doc ID [...], fn. 125), Google itself did not distinguish between the Android OS and the Android platform. The Commission found that the term platform software refers to any software product that exposes an API for use by applications and that such platforms typically provide many APIs which "enable new combinations of functions to offer richer usability and innovations" at the app layer. Case COMP/M.9205, IBM/RedHat, para 187 (cited in Alphabet's response to the PFs, Doc ID [...], fn. 125) does not refer to "critical device resources" – instead, it provides that it is the "operating system which provides [application developers] with the necessary computer resources to run", which are precisely the APIs exposed by the operating system and middleware.

474 Alphabet's response to the PFs, Doc ID [...], paragraph 161.

475 See https://developer.android.com/blog/posts/gemma-4-the-new-standard-for-local-agentic-intelligence-on-android, Doc ID [...], accessed on 8 July 2026.

476 See https://developer.android.com/ai/gemini-nano, Doc ID [...], accessed on 8 July 2026.

477 See https://developer.android.com/ai/gemini-nano, Doc ID [...], accessed on 8 July 2026.

*Android OS via AICore."*[478] The developer documentation is provided as part of the Android documentation.[479] The following diagram from the Android developer website identifies AICore and the Gemini Nano ODMs as part of "Android OS".

**Alphabet's depiction of AICore and Gemini Nano as part of the Google Android operating system**




*Source: Google developer website*[480]

(371) In addition, in the context of these proceedings Alphabet submitted that AICore can be described as middleware[481] and described it as a *"system-level module"*.[482] As such, it shares the characteristics of other Alphabet middleware and therefore, as system software, functions similarly to AOSP.

(372) In terms of functionality, AICore/Gemini Nano ODMs provide the basic operating system function of making system functionality accessible to all apps via public APIs as Google Android does. Alphabet published well-documented APIs to use functionality of the Gemini Nano ODMs, the encouraging developers to use these APIs. By positioning AICore/Gemini Nano ODMs as part of Google Android, Alphabet indicates that these APIs will be as accessible, stable and consistent as Google Android APIs.

(373) AICore/Gemini Nano ODMs contribute to controlling the basic functions of the software by powering other system functionality. In this respect they function as a part of Google Android. AICore/Gemini Nano ODMs are deeply interwoven with Google Android system functionality.

(a) Alphabet submitted that AICore supports *"critical system functionality"*. "Critical" system functionality is a basic function of a device' software. Alphabet provides the example that AICore is used to support scam call detection. More specifically, on calls, the user "*could lose the ability to detect scam calls due to latency in AICore"*.[483] The scam detection functionality is powered by Gemini Nano ODMs;[484]

---

478 See https://developers.google.com/solutions/pages/android-with-ai, Doc ID [...], accessed on 8 July 2026.

479 "You can learn more about Gemini Nano's technical architecture in the Android documentation." See https://developers.google.com/solutions/pages/android-with-ai, Doc ID [...], accessed on 8 July 2026.

480 See https://developer.android.com/ai/gemini-nano, Doc ID [...], accessed on 8 July 2026.

481 Alphabet's reply to RFI PO 11 of 20 March 2026, Doc ID [...], question 16.e.

482 See https://developer.android.com/ai/gemini-nano, Doc ID [...], accessed on 8 July 2026.

483 Alphabet's response to the PFs, Doc ID [...], paragraph 164 (last bullet), 324.

484 See https://support.google.com/phoneapp/answer/15654065, Doc ID [...], accessed on 8 July 2026.

(b) The Commission understands that AICore/Gemini Nano ODMs power the functionality *AI notification summaries*, which summarizes notifications.[485] Notifications are a feature of AOSP and the *AI notification summaries* functionality merely extends the existing AOSP feature; and

(c) Android System Intelligence also uses AICore to offer *"on-device intelligence services"*. Alphabet itself positions Android System Intelligence as a system component,[486] and it is used to provide certain system-level utility functions.[487] AICore enjoys access to system-privileged permissions which "*are inherently designed for the features that call on AICore, particularly on-device intelligence functionalities*".[488]

(374) In addition, AICore/Gemini Nano ODMs contribute to providing the basic operating system function of text creation and editing tools. Such tools have been part of all user-facing operating systems. For instance, already the early operating system Microsoft Windows 1, launched in 1985, included a text editor and clipboard support (copy and paste text) as basic functions. Later on, features such as speech recognition and proofreading (which the Gemini Nano ODMs offer) were added in the early 2000s and 2010s. Speech recognition functionality became a major Windows feature in Windows Vista in 2007.[489] Similarly, on macOS, dictation features became powerful in OS X Mountain Lion in 2012.[490] iOS introduced keyboard dictation together with Siri in 2011.[491] Alphabet included speech recognition in Android 2.1 in 2010.[492] Proofreading functionality was introduced in macOS 10.0 in 2001 via a system-wide spell-checking framework available across applications,[493] in Windows in 2012 with Windows 8,[494] in iOS in 2009,[495] and in Android in 2011.[496] Word prediction and text suggestion features were added in 2012 in Windows,[497] in 2013 on Android,[498] and in iOS in 2014.[499]

---

485 See https://www.androidauthority.com/android-notification-summaries-leak-3574675, Doc ID [...], accessed on 10 July 2026.

486 See https://support.google.com/pixelphone/answer/12112173, Doc ID [...], accessed on 8 July 2026.

487 Recital (853).

488 Alphabet's response to the PFs, Doc ID [...], paragraph 331.

489 See https://learn.microsoft.com/en-us/archive/msdn-magazine/2006/january/exploring-speech-recognition-and-synthesis-apis-in-windows-vista, Doc ID [...], accessed on 8 July 2026

490 See https://www.apple.com/newsroom/2012/07/25Mountain-Lion-Available-Today-From-the-Mac-App-Store, Doc ID [...], accessed on 8 July 2026

491 See https://www.apple.com/newsroom/2011/10/04Apple-Launches-iPhone-4S-iOS-5-iCloud, Doc ID [...], accessed on 8 July 2026

492 See https://android-developers.googleblog.com/2010/03/speech-input-api-for-android.html, Doc ID [...], accessed on 8 July 2026

493 See https://developer.apple.com/documentation/foundation/nsspellserver, Doc ID [...], accessed on 8 July 2026

494 See https://learn.microsoft.com/en-us/windows/win32/intl/about-the-spell-checker-api, Doc ID [...], accessed on 8 July 2026

495 See iOS 3.1 user guide ("Depending on the application you're using, the intelligent keyboard may automatically suggest corrections as you type to help prevent mistyped words."), accessible at https://cdsassets.apple.com/live/6GJYWVAV/user/ma616_iphone_ios3_1_user_guide.pdf, Doc ID [...].

496 https://developer.android.com/about/versions/android-4.0-highlights, Doc ID [...], accessed on 10 July 2026.

497 For touch keyboards (see https://www.computerworld.com/article/1379684/windows-8-s-touchscreen-keyboard-is-terrible-here-s-a-better-built-in-option.html, Doc ID [...]). For physical keyboards in 2018 (see https://www.howtogeek.com/429702/how-to-enable-text-prediction-for-a-hardware-keyboard-on-windows-10/, Doc ID [...]).

498 Through Google's keyboard improvements in Android 4.4 (see https://www.gottabemobile.com/android-4-4-kitkat-google-keyboard-available-now/, Doc ID [...]).

(375) These developments marked the transition from simple autocorrect systems to context-aware predictive text powered by statistical language models and, later, machine-learning techniques. The functionality offered by AICore/Gemini Nano presents a further advancement of this operating system functionality.

(376) Alphabet argued that ODMs are not *"'basic functions' but sophisticated enhancements"*.[500] What qualifies as "basic" evolves over time and can change rapidly in the face of major technological developments. Modern mobile operating systems are complex and powerful software that seek to enable developers to build feature-rich apps.[501] Interfaces to use sophisticated machine learning models have long been part of iOS and Google Android.[502] Since the advent of powerful LLMs in 2020 and the launch of ChatGPT in 2022, major operating systems have successively been adding LLM-based AI features which are together considered as "basic". Alphabet stated that *"Android transitions from an operating system into an intelligence system"*.[503] As Alphabet admits, the new technology often only "enhances" existing functions, some of which are basic (such as notifications, text creation and editing, including speech recognition, proofreading and word prediction and text suggestion, see above).

(377) Alphabet's contention is inconsistent with its own understanding of "basic device features" of Google Android devices, which includes any functionality which fulfils a platform-defined role, which is assigned as a default app, and is part of AOSP. It also includes features which [examples of features that are considered basic device features].[504] AICore/Gemini Nano models operate like a default app because they are pre-installed, provide a useful feature used by a significant number of users and preserve device resources (apps do not need to implement their own models which blocks memory and storage resources).

(378) In addition, Alphabet submits that ODMs *"perform no role in mediating access to device hardware and software"*.[505] The Commission notes that the accessibility of AICore/Gemini Nano ODMs as part of the operating system allows developers to easily use these ODMs and access the underlying hardware, thus indirectly mediating access to significant hardware resources. This mediation is particularly pronounced for AICore/Gemini Nano ODMs which require a large part of the device RAM (see Section 11.3), leaving limited space for other ODMs. Alphabet promotes this ease of hardware access to developers, which "don't need to worry about downloading or updating large models over the network, nor impact on your app's disk and runtime memory budget" nor about "the underlying hardware interfaces".[506]

(379) As a result, the Commission considers for the purpose of these proceedings that AICore and the Gemini Nano ODMs are part of the Google Android operating system as designated under the Designation Decision because they contribute to

---

499 Via the QuickType predictive keyboard in iOS 8 (see https://www.imore.com/quicktype-keyboards-ios-8-explained, Doc ID [...]).

500 Alphabet's response to the PFs, Doc ID [...], paragraph 161.

501 See Section 5.1.2 and recitals (225)-(228).

502 See https://developer.apple.com/news/?id=06052017c, Doc ID [...], https://developer.android.com/ndk/guides/neuralnetworks, Doc ID [...], accessed on 10 July 2026.

503 See https://blog.google/products-and-platforms/platforms/android/gemini-intelligence, Doc ID [...]; https://developer.apple.com/documentation/FoundationModels, Doc ID [...]; https://blogs.windows.com/windowsexperience/2025/05/06/introducing-a-new-generation-of-windows-experiences/, Doc ID [...].

504 [Alphabet internal document], Doc ID […].

505 Alphabet's response to the PFs, Doc ID [...], paragraph 164.

506 See https://developer.android.com/ai/gemini-nano, Doc ID [...], accessed on 8 July 2026.

controlling basic functions of Google Android devices as well as to enabling apps to run on them, thereby ensuring an effective functioning of Google Android.

6.2.2.2. The Google Hotword Engine is part of the designated middleware

(380) The Google Hotword Engine is a software component that is part of the Google Android operating system and is required for always-on hotword detection (AOHD) in Google Android. AOHD enables apps on Android to be invoked through uttering of a hotword, e.g. the user saying "Hey Google". Alphabet's services, such as Gemini and Google Assistant, use the Google Hotword Engine to implement AOHD in their Android apps.

(381) In its response to the PFs, Alphabet argued that the Google Hotword Engine is not part of the Google Android operating system because it does not control basic functions of the hardware or software and does not enable software applications to run on it.[507]

(382) One basic hardware component present on almost all Google Android mobile devices in the last 10 years is the DSP chip. The DSP is a power-efficient chip, which handles several tasks related to digital signals. The DSP processes the audio from the microphone to cancel background noises. The DSP includes hotword detection function, i.e., the ability to wake an app when the user speaks the hotword phrase, e.g. "Hey Google".

(383) Alphabet developed the Google Hotword Engine to activate the hotword detection function on the DSP. The Google Hotword Engine is integrated into Google Android, i.e., it is implemented at the system level, not the app level.

(384) It is part of Google Android's system libraries and preinstalled on all Google Android devices.[508] Alphabet submitted that the Google Hotword Engine is a "Core service" which [Description of Alphabet's commercial agreement with OEMs][509] The Google Hotword Engine must be implemented into the OS at build time, i.e., it is part of the operating system on device which will be shipped to the user.[510] The only way to update the DSP library is via an OS update. [Description of Alphabet's internal views on OS integrity].[511]

(385) In addition, AOSP exposes APIs that can interact with components of the DSP to enable hotword detection. These APIs, which are part of Google Android, are designed to only work with the Google Hotword Engine. Therefore, without the Google Hotword Engine, the Google Android operating system would not work as

---

507 Alphabet's response to the PFs, Doc ID [...], paragraph 161.

508 Google Android comes with a range of system libraries that are used to contribute to control basic functions of the OS. Many of these system libraries are available, prebuilt like the Google Hotword Engine, in the AOSP project. See here: https://cs.android.com/android/platform/superproject/+/android-latest-release:prebuilts/module_sdk/;drc=5750195121e36bb55b446b77c2d212a5770b2e43, accessed 16 June 2026, Doc ID [...], accessed on 10 July 2026

509 See Alphabet's response to RFI AI 4, Q3a.18, Doc ID [...], [Description of Alphabet's commercial agreement with OEMs]

510 See Alphabet's reply to RFI AI 4, Annex Q3A.20, Doc ID [...], Section *Integrating the DSP library*. See also Alphabet's reply to RFI PO 6, Doc ID [...], question 32. See Alphabet's response to RFI AI 4, Doc ID [...], [description of Alphabet's commercial agreement with OEMs].

511 [Description of Alphabet's internal views on integrity] See Alphabet's submission [description of Alphabet's internal views on integrity].The sound model does not execute code directly and could not cause such failures, as long as the underlying engine is implemented correctly. See also Agreed minutes of meeting with [Academics] o f 29 May 2026, Doc ID [...].

intended.[512] By contrast, Google Android does not offer standardised APIs (not even for preinstalled apps) that would allow developers to implement always-on hotword detection for their apps without using the Google Hotword Engine. While a third-party OEM may implement a separate AOHD feature decoupled from Google Android by building a custom interface to the DSP based on low-level integration, such level of access is not accessible to app developers.[513]

(386) The Google Hotword Engine therefore activates device hardware to enable an operating system feature. It thereby contributes to controlling the basic functions of the software (the operating system feature) and the hardware (the DSP).

(387) Hotword detection is itself a basic function of the DSP. It is supported by most DSPs today.[514] Qualcomm, a major supplier of chipsets [description of commercial activity], has been offering integrated chips with DSP functionality from as early as 2013 thereby enabling *"always-on, low-power listening feature to be "woken up" by a custom voice command"*.[515] Similarly, 13 years ago, Motorola, at that time a subsidiary of Alphabet, released the Moto X, the first smartphone hotword detection based on the DSP.[516] The Moto X, was deeply integrated with hotword detection. Users could speak the hotword phrase "OK Google Now" to wake "Google Now", the predecessor of Google Assistant.[517] Google Now already allowed the user to perform online searches and request information via voice input.[518]

(388) By activating hotword detection on the device's DSP, the Google Hotword Engine contributes to controlling the basic functions of the hardware (the DSP).

(389) In the response to the PFs, Alphabet claimed that the Google Hotword Engine only serves the narrow purpose *"to facilitate an invocation method for Google's assistive apps".* As explained, the Google Hotword Engine is not an app-level feature.[519] Its wider role is underlined by the fact that, to the Commission's understanding, the Google Hotword Engine also enables other system functionalities, such as the detection if music is playing in background, enabling Alphabet's Now Playing

---

512 The Android source code is designed to be hardwired to the Google Hotword Engine. the code shows even if different modules are loaded (i.e. different hotword engines), it will always select the first module (see line 110, ".findFirst()"). Even if an OEM or DSP vendor would offer a replacement engine, the Android source code does not allow to use this engine. See Android source code https://android.googlesource.com/platform/frameworks/base/+/45034f0663f960d9ee5fb0a101a4732b71f6e2f4/media/java/android/media/soundtrigger/SoundTriggerManager.java#105, Doc ID [...].

513 See agreed minutes of meeting with [AI service provider] of 12 March 2026, Doc ID [...], para 3-6. See agreed minutes of meeting with [Third party] of 26 March 2026, Doc ID [...], para 6, 11-14, 16.

514 See the list of supported DSPs submitted by Alphabet as a response to AI RFI 4 of 28 November 2025, Annex Q9, Doc ID [...]. Other non-Android smartphones, such as the iPhone provide their own DSPs with similar functionality.

515 See https://www.qualcomm.com/news/onq/2013/02/snapdragon-wakes-mobile-world-snapdragon-voice-activation, Doc ID [...].

516 Apple released the iPhone 6S with always-on hotword detection two years later in 2015. See https://www.apple.com/newsroom/2015/09/09Apple-Introduces-iPhone-6s-iPhone-6s-Plus/, Doc ID [...], accessed on 7 June 2026.

517 Gibbs, S. (2014), 'Motorola Moto X review: less "hello Moto", more hello Google', The Guardian, 17 January, https://www.theguardian.com/technology/2014/jan/17/motorola-moto-x-phone-review-google, Doc ID [...].

518 Ravenscraft, E. (2012), 'Google Search Updated, Brings New Google Now Cards And Voice Actions - Yes, You Can Set Calendar Events', Android Police website, 29 October, accessed 15 June 2026, https://www.androidpolice.com/2012/10/29/google-search-updated-brings-new-google-now-cards-and-voice-actions-yes-you-can-set-calendar-events/, Doc ID [...].

519 See Alphabet's response to the PFs, Doc ID [...], paragraph 164.

system feature on Pixel devices.[520] [Description of Alphabet's commercial agreements with OEMs regrading hotwords].[521] Its technical implementation does not impose such a limitation, and could provide broader application beyond support for hotwords of Google's assistive apps.[522]

(390) The Google Hotword Engine contributes to enabling software applications to run on Google Android. Indeed, the Google Hotword Engine activates the hotword detection on the DSP. Once hotword detection is activated, it enables software applications to run on the operating system.

(391) Moreover, functions which the operating system makes accessible to software applications enable these applications to run on the operating system: their purpose is precisely to enable these applications to run on the operating system using these functions (typically exposed through APIs).

(392) The same applies to the system functions exposed through such APIs. Without such functions the API would be an empty shell, contrary to its purpose. Without the Google Hotword Engine the AOHD feature in Google Android would not work. It would remove a feature that has existed for 13 years on Google Android. The apps currently using this feature, such as Google Gemini, could no langer access it. While the Android APIs that should provide access to this feature, and that are used by Google Gemini, would continue to exist, their functionality would no longer be accessible without a complete hardware activation (which includes the Google Hotword Engine).

(393) As a result, the Commission considers for the purpose of these proceedings that the Google Hotword Engine is part of the Google Android operating system as designated under the Designation because it contributes to controlling basic functions of Google Android devices and to enabling software applications to run on them thereby ensuring an effective functioning of Google Android.

### 6.3. Integrity justification

(394) As set out in Section 4.8 of this Decision, Article 6(7) DMA specifically provides for the possibility for the gatekeeper to take strictly necessary and proportionate measures to ensure that interoperability does not compromise the integrity of the operating system and hardware or software features provided by the gatekeeper, provided that such measures are strictly necessary, proportionate, and duly justified by the gatekeeper. The burden is on the gatekeeper to justify that any integrity measure in the implementation of an interoperability solution is strictly necessary and proportionate.

(395) In the context of these proceedings, Alphabet argued that the current Android ecosystem is already designed to mitigate the integrity issues for all features subject to these proceedings. Specifically, Alphabet argued that its permission model – and in particular delegating to OEMs the ability to grant access to privileged permissions

---

520 Alphabet's response to RFI PO 6, Doc ID [...], question 33, paragraph 160, where Alphabet explained that the Now Playing sound model is using a DSP library. As the Google Hotword Engine is the only DSP library supported on Pixel phones, the Commission understands that Now Playing must therefore be using the Google Hotword Engine as a DSP library.

521 [Description of Alphabet's commercial agreements with OEMs regrading hotwords], Alphabet's reply to RFI AI 2 of 23 September 2025. [information related to Alphabet's commercial agreements].

522 See agreed minutes of meeting with [third party] of 26 March 2026, Doc ID [...], paragraph 6, 11-14, 16.

– already amounts to strictly necessary and proportionate integrity measures within the meaning of Article 6(7) DMA.[523]

(396) As a fallback, during the course of these specification proceedings, Alphabet gradually articulated an alternative integrity measure, whereby access to certain features would instead require AI assistants to obtain a certification from Alphabet if they meet certain eligibility criteria defined by Alphabet – see Section 12.4.

(397) In this respect, the Commission notes that the generic argument that the permission model designed by Alphabet is the appropriate integrity measure in the context of Alphabet's implementation of its interoperability obligations – meaning the least restrictive measure, i.e. that Google Android is currently as open as it can be – cannot be accepted. As established in Section 5.2.1 above, when developing a feature Alphabet regularly assesses and decides the degree of openness of the feature, and may move features away from requiring system privileges. This means that the requirement of system privileges for that feature was not the least restrictive measure to begin with. Integrity considerations should therefore come into play in the context of the implementation of interoperability solutions, rather than as a defence against developing an interoperability solution at all.

(398) Alphabet insisted on this generic argument to avoid allowing interoperability with the features covered by these proceedings. Therefore, the Commission assesses it in the next section.

*6.3.1. Delegating interoperability decisions to OEMs is not a necessary and proportionate integrity measure within the meaning of Article 6(7) DMA*

(399) Alphabet argues that granting access to the features covered by this Decision gives rise to integrity risks that are proportionately addressed by subjecting the features to system-level privileges under Android's permission model.[524] Specifically, Alphabet argued that the system put in place by Alphabet, whereby OEMs must decide whether to grant feature access to third parties for Google Android implemented on their devices, as described in Sections 5.2.3 and 5.2.5, constitutes the appropriate integrity measure, relying on the "objective role of OEMs" to grant access to these features,[525] and is a necessary and proportionate integrity measure per Article 6(7) DMA.[526]

(400) Alphabet submits that the OEM delegation integrity measure is necessary and proportionate in view of the openness of the Android ecosystem.[527] Alphabet states that the Android permission model guards against these risks while maintaining the openness of the ecosystem, as it is "*an essential mechanism to balance between functionality and security*", in which "*for some features, Google determines that OEM pre-grant is a proportionate condition for access*".[528]

---

523 Alphabet's response to the PFs, Doc ID [...], paragraph 134.

524 Alphabet's response to the PFs, Doc ID [...], Section V. Alphabet explained that concerns are particularly high for powerful features (Computer Control, App Functions, Ambient data) and for always-on hotword detection – see Technical meeting Alphabet/Commission of 6 March 2026, Doc ID [...], slide deck pages 29-30 and 31-36.

525 Alphabet's submission of 3 March 2026, Doc ID [...].

526 Alphabet's response to the PFs, Doc ID [...], Section V.

527 Alphabet's response to the PFs, Doc ID [...], paragraphs 138-140.

528 Alphabet's response to the PFs, Doc ID [...], paragraph 140. Expert report of Prof. [academic], Annex 1 of Alphabet's response to the PFs, Doc ID [...], paragraph 30.

(401) The Commission does not contest Alphabet's general position[529] that a permission model with tiers of access and user authorisation, like the Android permission model, is beneficial to integrity, user agency, security, and privacy. The Commission also recognises that certain features covered by these proceedings enable strong capabilities that may have implications for integrity, security, and privacy. These implications are exacerbated by the use of AI systems, which introduce both novel vectors for potential abuse and a higher risk of unintended consequences, even without malicious intent.[530] These risks equally apply when the feature is used by Alphabet's services.

(402) Nevertheless, subjecting the specific features that are available to Alphabet's services to the "privileged" protection level and therefore OEM preinstallation cannot be accepted as a necessary and proportionate integrity measure within the meaning of Article 6(7) DMA, as set out below. Alphabet's decision to delegate responsibilities about whether interoperability should be provided, to whom and under which conditions artificially creates scarcity in a context – access to features – where there is none.[531] It unduly limits access to these features to a limited set of beneficiaries selected by the OEM – often for commercial reasons, not integrity reasons – even if there are other beneficiaries that achieve the same or higher safety standards. In fact, there are several factors that indicate that OEM delegation and preinstallation are not suitable safeguards to protect integrity.

(403) *First*, subjecting access to key features to preinstallation agreements with OEMs, *de facto* puts OEMs in a position to refuse feature access even if the conditions for such preinstallation do not meet the requirements for taking integrity measures pursuant to Article 6(7), 3rd sentence. The "objectivity" of OEMs with respect to integrity decisions concerning the designated operating system is questionable and is in no way assessed, controlled or ensured by Alphabet. In particular,

(a) OEMs may have an incentive to monetise feature access and to preference their own services, including their own AI assistant apps, which directly goes against the fairness objective of the DMA (see Section 5.4.1);

(b) Pre-installation agreements allow OEMs to charge for something that Article 6(7) DMA guarantees to its beneficiaries free of charge, thereby casting further doubt on whether OEMs can be the "objective" arbiter of integrity;

(c) Alphabet does not explain why OEMs are best placed to make objective integrity decisions nor why they would be better placed or more objective than e.g. an independent third-party assessor. Alphabet merely argues that OEMs share Alphabet's incentives not to allow devices to become corrupted or users to be harmed because they would carry "significant reputational and financial risk" if their devices became vectors for user harm.[532] In fact, this argument is contradicted by Alphabet's own statements that only Alphabet is able to decide which services meet appropriate eligibility standards to access some restricted features – see Section 12.4.

---

529 Alphabet's response to the PFs, Doc ID [...], paragraph 138. Expert report of Prof. [academic] Annex 1 of Alphabet's response to the PFs, Doc ID [...], para 31.

530 For example, the AI system may misinterpret a user's request and inadvertently start deleting a user's files, when this was not the user's intent, without any malicious intent by the AI system either.

531 In the limited cases where it is technically infeasible to allow multiple services to concurrently access a feature, this can be addressed by allowing the user to choose which services should have that access.

532 Alphabet's response to the PFs, Doc ID [...], paragraph 142.

(404) *Second*, recital 50 DMA, last sentence, provides that, with respect to both the protection of integrity and security, the gatekeeper should be prevented from implementing such (integrity and security) measures as a default setting or as preinstallation. The legislator was aware of the disproportionate risks of allowing the gatekeeper to use default settings or preinstallation to justify potential non-compliance based on integrity and/or security. This is consistent with the fact that both default settings and preinstallation are "crude" measures which, by their very nature, are not tailored to the actual risks. By analogy, this also applies to a delegation of compliance decisions to OEMs by a gatekeeper.

(405) *Third*, with respect to the proportionality of Alphabet's preinstallation system, the Commission notes that this system in practice favours Alphabet, which is particularly well placed to obtain feature access from OEMs as the provider of Google Android, while third parties (including smaller app developers) do not have the knowledge, relationships and resources to negotiate feature access with multiple OEMs (see Section 5.4). Building relationships and reaching agreements with many OEMs that are aligned with the OEM device cycles places a substantial burden on third parties and may be extremely difficult for smaller AI service providers.

(406) Nowhere is Alphabet's privileged position clearer than on its Google Pixel devices. Alphabet argues that it can freely decide that only Alphabet's apps have access to features in scope of Article 6(7) on Google Pixel devices.[533] In other words, Alphabet argues that the integrity justification allows Alphabet (as the provider of Google Android) to delegate to Alphabet (as the provider of Google Pixel devices) the decision of whether to provide interoperability pursuant to Article 6(7) to any third party. Alphabet, or any other gatekeeper providing a designated operating system, could simply avoid complying with Article 6(7) by supplying the operating system to its own OEM, i.e. itself. Similarly, Alphabet's argument implies that Alphabet could theoretically remove access to any feature that is currently interoperable and subject it to OEM approval, which would result in millions of apps to stop working as intended, and still consider that it complies with Article 6(7). This further illustrates the extent to which delegation to OEMs of the interoperability decision is contrary to the objectives of Article 6(7).

(407) *Fourth*, the preinstallation system also means that OEMs cannot open a feature to all third parties and end users cannot select which apps should have access to any particular feature. This is because, under this model, the decision over feature access is inherently taken before the device reaches end users. This system limits end user choice and the ability of business users to compete on the merits and bring innovative services to end users, including after the device was shipped. OEMs also seek to minimise the number of preinstalled apps, particularly those that are preinstalled in the system partition and cannot be uninstalled, given the inherent storage limitations associated with preinstalling apps on mobile devices, while end users may want to preserve a large amount of storage space and may dislike when OEMs preinstalled many apps that they have not decided to download – which for these reasons are often referred to as 'bloatware'.[534]

---

533 See Alphabet updated reply of 23 November 2025 to RFI AI of 17 July 2025, Doc ID [...], paragraph 12.

534 See https://www.androidpolice.com/i-finally-deleted-all-my-androids-bloatware-and-it-feels-like-a-new-phone/, Doc ID [...], accessed on 8 July 2026. [OEM] provided that it does not pre-install any apps except Google Applications which it is contractually obligated to pre-install. [OEM] wants as little bloatware (i.e., pre-installed apps) as possible. Agreed minutes of meeting with [OEM] on 9 March

(408) *Fifth*, the preinstallation system requires approval by each OEM separately. Even if a third party manages to negotiate access through preinstallation, the OEMs would likely not preinstall the app on all their devices, which wholly contradicts the need for integrity measures to be objective and consistently applied.[535] This system is ineffective at granting interoperability to third parties, as it would burden them with unnecessarily repeating the same vetting and negotiation for preinstallation of their app across every single OEM. Moreover, an OEM could decide to not preinstall that app on future devices, again for reasons that are potentially entirely unrelated to integrity. This inevitably leads to differences and unreliability of the Android API surface, which is the main reason behind Android's success: if OEMs were free to open or close access for every API, developing an Android app would be immensely more difficult and expensive. This is precisely the reason why Alphabet prohibits OEMs from restricting access to Android SDK APIs (i.e. for features that Alphabet has decided to make interoperable), but then allows them to do so for some features where such differences in the accessible APIs plays to Alphabet's advantage (see Section 5.2.1), such as the APIs that grant access to the features covered by this Decision. In contrast, Alphabet exempts itself from this arduous requirement of repeated approval from different OEMs. [Description of commercial agreement] .

(409) *Sixth*, contrary to the claim that the OEMs should be the objective arbiter of integrity questions, Alphabet itself seeks control on integrity, security, and privacy matters, over OEMs via the EMADA,[536] GMS Requirements,[537] and the CDD.[538] This is perhaps best exemplified by the Mobile Bundled Apps (MBA) policy (which is part of the GMS Requirements), where Alphabet unilaterally imposes integrity, security, and privacy requirements on OEMs for any app the OEM seeks to pre-install.[539] OEMs submitted that they are subject to these requirements.[540]

(410) Alphabet also goes beyond purely contractual requirements on OEMs. In 2018, Alphabet introduced a Build Test Suite, which evaluates every OEM Android build submitted. The suite checks for security issues in the Android software but also checks if any potentially harmful apps are preinstalled. In 2018, Alphabet found 242 OEM Android builds that contained potentially harmful apps.[541] The numbers of builds with potentially harmful apps decreased since then, but remains [Alphabet

---

2026, Doc ID [...], para 18. As an Alphabet internal document shows, users and the market [direct quote describing Alphabet's internal processes].

535 Amazon's Alexa assistant in the Alexa built-in program was only preinstalled on a small number of devices of the OEM's devices. See Fatmi, N. S. and Maring, J. (2021), 'Best Alexa built-in phone 2022', Android Central website, 13 December, accessed 3 July 2026, https://www.androidcentral.com/best-alexa-built-phone, Doc ID [...].

536 [Description of terms of commercial agreements with OEMs regarding security requirements].

537 [Description of terms of commercial agreements with OEMs regarding security requirements].

538 Section 9 focused on the Security Model and Section 9.8 in the CDD is focused on privacy, see Android 16 Compatibility Definition, https://source.android.com/docs/compatibility/16/android-16-cdd, Doc ID [...].

539 [Description of terms of commercial agreements with OEMs regarding security requirements].

540 For instance, [OEM] reply to RFI 1 of 5 March 2026, Doc ID [...], question 4; [OEM] reply to RFI 1 of 5 March 2026, Doc ID [...].

541 See "Android Security & Privacy 2018 Year In Review", March 2019, https://source.android.com/static/docs/security/overview/reports/Google_Android_Security_2018_Report_Final.pdf, Doc ID [...], page 11.

internal data].[542] In those cases, Alphabet reaches out to the OEM "*to remediate and remove the PHA from the build before it can be offered to users.*"[543]

(411) This is likely in recognition of the tension with OEMs' commercial motive: Alphabet wants to avoid that OEMs get enticed by financial incentives to the detriment of integrity, security, and privacy. In addition, preinstalled apps are not immune to vulnerabilities and are in fact an attractive target for malicious actors: in the words of a Google security researcher, such actors "*only have to convince 1 company to include [their] app rather than thousands of users*".[544] For example, in 2019, malicious preinstalled apps were deployed on millions of devices.[545] In the case of preinstalled apps, the user does not have any agency to prevent such a deployment of a malicious app, other than choosing another device, while for downloaded apps, the user remains fully in control and can take an immediate remedial action when the trust is broken.

(412) *Seventh*, Alphabet's claim that OEM control over certain features is the necessary shield against integrity risks is not supported by Alphabet's internal documents.[546] For example, internal documents show that [Alphabet internal document describing approach to permission-guarded features and justifying denial of access] .[547] Another internal document shows that the use of System APIs increases [direct quote describing Alphabet's commercial and strategic discussions], indicating that the measure limits access beyond integrity concerns.[548]

(413) Furthermore, Alphabet's decision to leave it to OEMs to decide over feature access is not necessarily motivated by integrity considerations. Alphabet changed the LPH/LPNH contextual invocation feature (which it currently uses with Google Search's Circle to Search solution) [description of Alphabet's internal processes and systems].[549]Another internal document shows that even for sensitive AI-related features, such as [direct quote describing Alphabet's commercial and strategic discussions]. Leaving it to OEMs was one option beside a "Public API with security check" or a "Public API".

(414) Similarly, Alphabet's own assessment of allocation of responsibilities associated with AI-related features points to Alphabet's responsibility for security rather than delegating it to OEMs as a matter of principle. For instance, Alphabet states that [direct quote describing Alphabet's ecosystem product design choices].[550] [Direct quote describing Alphabet's ecosystem product design choices. [551]

---

542 [Alphabet internal data], Alphabet's reply to RFI PO 22 of 17 June 2026, Annex Q3, Doc ID [...].

543 See "Android Security & Privacy 2018 Year In Review", March 2019, https://source.android.com/static/docs/security/overview/reports/Google_Android_Security_2018_Report_Final.pdf, Doc ID [...], page 11.

544 See "Securing the System: A Deep Dive into Reversing Android Pre-Installed Apps", presentation by Google security researcher Maddie Stone, Black Hat USA 2019, https://i.blackhat.com/USA-19/Thursday/us-19-Stone-Securing-The-System-A-Deep-Dive-Into-Reversing-Android-Preinstalled-Apps.pdf, Doc ID [...], page 14.

545 See "Securing the System: A Deep Dive into Reversing Android Pre-Installed Apps", presentation by Google security researcher Maddie Stone, Black Hat USA 2019, https://i.blackhat.com/USA-19/Thursday/us-19-Stone-Securing-The-System-A-Deep-Dive-Into-Reversing-Android-Preinstalled-Apps.pdf, Doc ID [...], pages 53 and 71.

546 Alphabet's response to the PFs, Doc ID [...], paragraph 141-143.

547 Alphabet internal document, Presentation [Alphabet internal document], Doc ID [...], page 5.

548 Alphabet internal document, Document [Alphabet internal document], Doc ID [...], page 5.

549 [Alphabet internal document], Doc ID […].

550 [Alphabet internal document], Doc ID […].

551 [Alphabet internal document], Doc ID […].

(415) The argument that preinstallation is a suitable integrity measure is also contradicted by an internal document illustrating that the pre-installation requirement contradicts Android's philosophy in terms of fairness and security. [Direct quote describing Alphabet's commercial and strategic discussions].[552]

(416) [Description of Alphabet's commercial and strategic discussions regarding circle to search].[553]

(417) *Eighth*, Alphabet already today requires OEMs to implement integrity, security and privacy measures that are less restrictive than the OEM delegation integrity measure.[554] For example, Alphabet requires OEMs to allow developers to ask users to grant access to sensitive features (e.g. microphone/photos/location). These measures are less restrictive because they allow third-party developers to access certain features, conditional on user consent, and apply in a non-discriminatory manner to Alphabet's, third-party and OEM apps.

(418) On the basis of the above and the evidence available, the Commission considers that allowing OEMs to control feature access via preinstallation does not meet the requirements of the integrity justification in Article 6(7) DMA.

*6.3.2. The risk of the features in the present proceedings*

(419) Alphabet submits that the present proceedings concern the most dangerous and high-risk features of the operating system with the intended beneficiaries being powerful AI services with unique attendant risks[555] and that this would justify withholding access from all user-downloaded apps (see previous section).

(420) *First,* the Commission notes that Alphabet repeatedly conflates integrity with security and privacy, disregarding the fact that Article 6(7) only refers to integrity measures.[556] In fact, Alphabet argues that *"end user security and privacy are relevant to integrity"*,[557] but does not explain *how* they are relevant, meaning that Alphabet's argument simply seeks to import security and privacy into Article 6(7). In fact, the Commission agrees that protecting integrity can benefit security and privacy, but this does not mean that the notions are the same.[558] The PFs explained that integrity should be interpreted as the state of being unimpaired and protected against manipulation that is not authorised by the end user.[559] Alphabet's response to the PFs completely disregards these arguments and arbitrarily stretches the notion of integrity to include any measure that Alphabet believes it should be entitled to impose and enforce.

---

552 [Alphabet internal document], Doc ID […].

553 [Description of Alphabet's commercial and strategic discussions regarding circle to search], see here: "[Alphabet internal document].

554 To that extent, the measures annexed to this Decision do not materially lessen OEMs' control over device security compared to the status quo, as raised by some OEMs based on their submission to the Public Consultation (see for instance [OEM] submission Doc ID [...] or [OEM] submission Doc ID [...]).

555 Alphabet's response to the PFs, Doc ID [...], paragraph 150, second and third bullet. Various submissions from the Public Consultation also invited the Commission to take account of the specific risks posed by AI services (see for instance [OEM] submission to the Public Consultation Doc ID [...], [OEM] submission to the Public Consultation Doc ID [...], [OEM] submission to the Public Consultation Doc ID [...], [OEM] submission to the Public Consultation Doc ID [...], [OEM] submission to the Public Consultation Doc ID [...]).

556 Alphabet's response to the PFs, Doc ID [...], includes 143 occurrences of the word "security" and 65 occurrences of the word "privacy".

557 Alphabet's response to the PFs, Doc ID [...], paragraph 148.

558 PFs, Doc ID [...], paragraph 109.

559 PFs, Doc ID [...], paragraphs 105 and 109.

(421) *Second,* Alphabet repeatedly references the Accessibility service API as a prime example of a powerful API with capabilities to observe user interactions and control the device, which is prone to misuses by malware.[560] However, such an API still serves a legitimate and important purpose, not in the least to enable services for people with accessibility needs. Alphabet's expert in the context of these proceedings recognises that the choice to introduce such an API therefore presents inherent trade-offs between integrity, security, and privacy on the one hand, and functionality on the other hand.[561] He explains that while a computer without core functionalities such as Internet access or app installation would be "extremely secure", it would also be useless for most people.[562] To enable functionalities that are of great utility to users with disabilities, the apps that provide such accessibility services need access to APIs capable of supporting the user's interactions, even if those may present risks. In the same vein, AI services such as assistants must get access to APIs so that they can be invoked, access user context, act on apps and the OS, and get access to device resources: otherwise, they might be in practice useless for users.

(422) The Accessibility service API is open to all apps, i.e. not subject to OEM approval. This indicates that Alphabet considers that the benefits of this API outweigh the negative drawbacks, including the risk arising from malicious apps and "actual user harm".[563] Instead, to use the Accessibility service API, the user must manually grant the permission in the Android settings app.[564] Apps distributed through the Google Play Store are subject to additional review as well as policies and use case restrictions set by Alphabet.[565]

(423) Alphabet admits that further restrictions on access to the Accessibility service API would have a [strategic considerations around risk posed].[566] Such a risk is precisely why OEM delegation is not an appropriate integrity measure under Article 6(7). Within the trade-off between functionality and integrity, security, and privacy, Alphabet has made the conscious choice to develop these features and make them available to its services, thereby creating the risks and attack surfaces that these new capabilities may bring about. Article 6(7) then applies to these features, requiring equally effective access for third parties.

(424) *Third,* in any event, none of the features and measures in this Decision would – in particular on their own – achieve the same level of access to user data or ability to act on the user's behalf, as the existing Accessibility service API does today. If anything, they would allow for much more granular measures and user consent, in line with the security principle of least privilege.[567]

---

560 Alphabet's response to the PFs, Doc ID [...], para 139. The Commission interprets the final sentence of that paragraph, "*Even well-intentioned apps can expose users to security risks and/or to vulnerabilities in non-malicious apps.*", as pointing out that malware can access and abuse the Accessibility API through vulnerabilities in benign apps; this would still constitute an (indirect) misuse by malware. The expert report attached to Alphabet's response also points to the dangers of this API, Expert report of Prof. [Academic], Annex 1 of Alphabet's response to the PFs, Doc ID [...], para 17, para 21-23.

561 Expert report of [Academic], Annex 1 of Alphabet's response to the PFs, Doc ID [...], paragraphs 19, 20, 23.

562 Expert report of [Academic], Annex 1 of Alphabet's response to the PFs, Doc ID [...], paragraph 19.

563 Expert report of [Academic], Annex 1 of Alphabet's response to the PFs, Doc ID [...], para 17, para 21-23.

564 Expert report of [Academic], Annex 1 of Alphabet's response to the PFs, Doc ID [...], para 18.

565 Expert report of [Academic], Annex 1 of Alphabet's response to the PFs, Doc ID [...], paragraphs 21, 23.

566 Technical meeting Alphabet/Commission of 6 March 2026, Doc ID [...], page 26.

567 The principle of least privilege asserts: "Every program and every privileged user of the system should operate using the least amount of privilege necessary to complete the job." – see Jerome H. Saltzer

(425) That said, the Commission recognises that some features in scope of this Decision are particularly sensitive and meet some exceptional conditions that might undermine developers' buy-in unless access is restricted in some way. Therefore, after the adoption of the PFs and the Public Consultation, the Commission decided to allow Alphabet to restrict features meeting such exceptional conditions and subject access to them to an eligibility programme – see Section 12.4.[568]

(426) Finally, the measures in this Decision do not prevent Alphabet from implementing additional integrity measures and other technical safeguards, including safeguards that support privacy and security, provided that these are based on transparent, objective, precise, and non-discriminatory conditions that also apply to Alphabet's services and provided that they do not prescribe how third parties should protect privacy and security in their own services (see also Section 4.8.2). Such measures should be proposed and duly justified by Alphabet to the Commission, and should be subject to feedback from interested third parties when appropriate.

## 7. INTRODUCTION TO THE FEATURES SUBJECT TO THE PROCEEDINGS

(427) Through these proceedings, the Commission is specifying the measures that Alphabet needs to implement to effectively comply with Article 6(7) DMA in relation to certain Google Android features relevant for AI services.

(428) For the purpose of this Decision, these features are organised into four groups, which represent the high-level building blocks necessary for effectively operating an AI service:

(a) **Invocation**, or the ability for an end user to initiate their interaction with an AI service through a variety of access points;

(b) **Context**, or the ability for an AI service to gather inputs and data to then understand and anticipate end users' needs;

(c) **Actions on apps and the OS**, or the ability for an AI service to execute tasks on a user's behalf; and

(d) **Access to resources**, or the ability for an AI service to use sufficient hardware and software resources to execute its tasks.

(429) These building blocks, and the features encompassed within them, cover the spectrum of components, triggers, inputs, and outputs that AI services rely on. These features enable various end user journeys that AI services may want to employ, ranging from *reactive* services that respond to explicit user commands, *proactive*

---

(1974), *"Protection and the control of information sharing in multics"*, https://dl.acm.org/doi/10.1145/361011.361067, Doc ID [...]. This principle is applied in Google Android by "Architectural decomposition", which means *"breaking privileged processes into less privileged components and applying attack surface reduction"* for some attack vectors that are inherent in Google Android – see See The Android Platform Security Model (2023), https://arxiv.org/pdf/1904.05572v3, Doc ID [...], accessed on 9 April 2026, pages 6 and 18. The measures specified in this Decision uphold the principle of least privilege because they allow Alphabet to require user authorisation in order to grant access to the features covered by this Decision.

568 The concerned features were flagged by some respondents to the Public Consultation as particularly sensitive, such as structured on-device integration (see for instance [OEM] submission Doc ID [...]), screen automation (see for instance [OEM] submission Doc ID [...], [OEM] submission Doc ID [...], [OEM] submission Doc ID [...] or [Think tank] Doc ID [...]), system integration (see for instance [OEM] submission Doc ID [...] or [OEM] submission Doc ID [...]), context-aware intelligence (see for instance the [Think tank] Doc ID [...]), or centralised access to on-device app data (see for instance [Think tank] Doc ID [...]).

services that anticipate user needs and suggest actions, to *autonomous* services that can act independently on the end user's behalf.[569] [Describes Alphabet's commercial and strategic position regarding AI services].[570] These groups of features align with core requirements that business users have identified for AI services to enjoy effective interoperability on Google Android.[571]

(430) All features within the scope of these proceedings are accessed or controlled via Alphabet's operating system Google Android and are used by, and/or available to, services provided by Alphabet.

## 8. MEASURES ON FEATURES FOR INVOCATION

### 8.1. Introduction

(431) There are different ways through which a user can launch or "invoke" an app or a service on mobile devices, commonly referred to as access points or affordances. While the traditional method consists of tapping an app icon on the device's screen, users are increasingly relying on other access points to interact with apps and services, in particular those that are accessible from anywhere in the system, such as hardware buttons, on-screen gestures, voice commands or system-level shortcuts.[572] With the growing integration with AI capabilities and agents into mobile devices, there is indeed a clear shift toward greater automation, where users increasingly rely on these alternative access points, to quickly interact with services, rather than explicitly opening apps. Similarly, many users have a habit to invoke apps with a voice command (also referred to as hotwords), especially for simpler tasks and questions or in certain hands-free situations. This evolution reflects a broader trend towards more seamless and context-aware interactions, which reduce the need for direct app-based navigation.

(432) In this new context, it is essential to ensure, on the one hand, user agency (i.e. the ability of the user to exercise choice), and on the other hand, an equal access to these access points and commands by all apps and services, be their first-party or third-party. This is because the way services are connected to specific access points can significantly shape and even entrench behavioural biases: when a particular service is tied to a prominent or convenient invocation method, it benefits from increased visibility and ease of use, which in turn translates into a higher usage and user adoption. In addition, when it comes to AI services, invocation represents the beginning of the user journey of interacting with a service or an app, and therefore it is key that all developers are on an equal footing from the first interaction with the user. Interoperability with these access points on mobile devices is also important to foster innovation by lowering barriers to entry for new services and enabling developers to experiment with novel interaction models. Ease of access to access points facilitates switching between AI services and multi-homing of several AI-powered services potentially aiming at different use cases.

---

569 Alphabet internal document, Document [Alphabet internal document] of 8 January 2026, Doc ID […], page 7.

570 Alphabet internal document, [Alphabet internal document], Doc ID […], page 56.

571 See for example, [AI service provider], "DMA.100220 – Preliminary and non-exhaustive list of topics", 18 February 2026, section "[AI service provider's] Introductory Comments", Doc ID […]. [AI service provider], "Ensure Effective Interoperability in Google's Android Ecosystem under the EU Digital Markets Act", 22 March 2026, Doc ID […].

572 For access points which can be invoked in different ways, such as long press, single tap and double tap, each such way of invocation constitutes a different access point.

(433) All these access points are rooted in the operating system of a device. To ensure a level playing field, Alphabet should not have preferential access to any of the access points – be it hardware buttons, on-screen gestures, or the ability to be woken by voice commands – by mere virtue of controlling Google Android. Subject to the measures in Section 8.2 below, this is without prejudice to a single access point (except the always-on hotword access point) being reserved to the default assistant in the out-of-the-box experience.

(434) Based on the assessment developed hereafter, the Commission considers that there is currently an interoperability gap when it comes to features for invocation for third-party services on Google Android mobile devices.

(435) The Commission therefore specifies how Alphabet must fill this gap, to comply with Article 6(7) DMA. The measures address core access points, being Long-press Home/Long-press Navigation Handle contextual invocation (Section 8.2) and always-on hotword detection (Section 8.3), which contribute to effectively enabling interoperability within the meaning of Article 6(7) DMA for third-party AI services.

## 8.2. Measures for Long-press Home/Long-press Navigation Handle contextual invocation feature

### *8.2.1. Introduction*

(436) On most modern Android devices, Alphabet has reserved a centrally located access point to invoke its AI-powered Google Search solution *Circle to Search*, which enables the user to select all or a portion of the screen for e.g. searching or translating with Google Search. This access point is, depending on the device setup, either the long-pressing the home button (LPH) or long-pressing navigation handle (LPNH), or any other access point agreed upon between Alphabet and the OEM.

(437) The LPH/LPNH access point is present on all Android devices[573] and is one of the most easily accessible ways for users to invoke an app or a service. By default, any app invoked through that access point additionally receives contextual data from the operating system, namely the content of the device screen or the app in the foreground, which the invoked service can use in many ways, such as to answer a query about what is visible on screen.

(438) Most (if not all) OEMs selling Android devices with Google Search's Circle to Search solution enabled on them have hardwired it to this prominent access point. This access point is particularly popular as it used to be the second-most used way to trigger the assistant,[574] and [describes Alphabet's strategic position regarding access points].[575]

(439) Given the importance of the access points in launching services on Google Android, equal access to the LPH/LPNH access point and other system-wide access points (as well as the related contextual data) is important for third-party AI services to offer easily accessible and attractive services to end users.

---

573 While the access point is present on all devices, Google's Circle to Search solution is enabled on that access point on the majority of modern Android devices.

574 [Describes Alphabet's internal metrics regarding use of features]. Alphabet internal document, Presentation [Alphabet internal document], Doc ID […].

575 Alphabet internal document, Email, with subject [Alphabet internal document], of 14 August 2024, Doc ID […].

*8.2.2. Description of the feature*

(440) The LPH/LPNH contextual invocation feature consists in the ability to use the LPH/LPNH access points (and other access points) as well as to receive contextual data, on par with Alphabet's services.

(441) The functionalities of the LPH/LPNH contextual invocation feature, include, amongst others, the following:

(a) Ability to be invoked system-wide through the LPH/LPNH access point and any other access points available to Alphabet's services, such as Google Search (via the Circle to Search solution);

(b) Upon invocation, ability to receive contextual data, including but not limited to a current screenshot, information about the open apps, information provided by apps in the foreground, such as a URL, picture, or additional data, as available to Alphabet's services, such as Google Search (via the Circle to Search solution); and

(c) Upon invocation, the ability to overlay content.

(442) Based on the default implementation of the Google Android launcher,[576] when the user triggers the LPH/LPNH access point on an Android mobile device, a service is invoked and can appear as an overlay on top of what the user sees on screen. In addition, the service receives contextual information including, but not limited to, a screenshot and information about the app(s) in the foreground. The service can use the contextual information provided to offer a service such as answering a query based on what the user sees on the device screen.

8.2.2.1. Access points

(443) On Android mobile devices, access points allow the user to invoke a service. The figure below shows system-level access points that Alphabet considered to use for its AI services.

**Figure 2 - Access points to invoke AI services on Android devices**

[Alphabet internal document]

*Source: Alphabet's internal document, [Alphabet internal document], Doc ID […], page 7.*

(444) The LPH/LPNH is one such access point. It allows users to quickly and seamlessly reach services without navigating to the specific related apps.

(445) Depending on the device setup either the LPH or the LPNH is invocable by the user:

(a) The LPH access point is invocable on devices where the user has chosen the traditional Android three-button navigation layout, which includes a home button; and

(b) The LPNH access point is invocable on devices where the user has chosen gesture-based navigation, where a navigation handle appears as a small horizontal bar, and acts as a dynamic OS-level access point that routes user gestures to specific services.

(446) When a user then presses and holds the home button or navigation handle, respectively, this extended touch is interpreted as a command to open a service.

---

576 The default system launcher for Android (called "quickstep") causes the LPH and LPNH access point to invoke the app linked to the ContextualSearch API. This API is used by Google to implement Circle to Search, when using the LPH/LPNH access point. Technical meeting Alphabet/Commission of 12 February 2026, Doc ID […], slide deck page 32.

(447) The LPH/LPNH access point is particularly relevant for users because it is located at the bottom of the device's screen just below the user's thumb regardless of what the screen displays (an app, the home screen etc.). Based on Alphabet's internal documents, it represented [10-20]% of the triggers of Google Assistant on Android devices when Google Assistant was located on that access point. It also represented [30-40%] of "*single day – single query*" invocations.[577]

(448) The Commission considers that the LPH/LPNH access point as well as other access points are operating system features. This is because they are implemented, controlled and managed by Google Android for the following reasons:

(a) The behaviour of these access points is defined at the operating system level and is part of AOSP.[578] It is controlled by the system launcher or system UI, which are part of AOSP;[579]

(b) Invocation of a service through these access points in fact rely on system APIs. Therefore, these access points act as gateway to services, with the operating system handling the input and coordinating service invocation. As explained by Alphabet regarding access points in general: "*AOSP provides the "plumbing" for invocation of services on the device. It detects input events, interprets events (e.g. long or short press), and routes input events to defined outcomes.*".[580]

(c) These access points are system-wide features. They provide a consistent behaviour regardless of which app is active on the device, because they are part of the operating system rather than app-level code; and

(d) The CDD provides requirements and recommendations on the behaviour of the LPH and other access points.[581] [describes Alphabet's commercial agreements with OEMs],[582] [describes Alphabet's commercial agreements with OEMs].

8.2.2.2. Contextual data surfaced through the access points are part of the Google Android feature

(449) Depending on the access point, invocation has the ability to provide the app with contextual data. This is the case for the LPH/LPNH access point (as well as for instance through the access point dedicated to invoking the default assistant). Based on the default implementation of the Google Android system launcher in the AOSP

---

577 Alphabet internal document, Presentation [Alphabet internal document], Doc ID […], slide deck 10. The Commission understands that "single day-single query" invocations correspond to invocations from "casual mobile assistant users" which mainly issue only one single query, as per slide 31 of the document.

578 See https://cs.android.com/android/platform/superproject/+/android-latest-release:frameworks/base/packages/SystemUI/src/com/android/systemui/navigationbar/views/NavigationBar.java?q=navigationbar&ss=android%2Fplatform%2Fsuperproject, Doc ID […], accessed on 8 July 2026.

579 Alphabet's reply to RFI PO 16 of 01 April 2026, Doc ID […], question 3.

580 Alphabet's reply to RFI PO 16 of 01 April 2026, Doc ID […], question 4(c).

581 See Alphabet's reply to RFI PO 16 of 01 April 2026, Doc ID […], question 4(d), where Alphabet explains that in particular, Section 7.2.3 of the CDD mandates that the core navigation functions (Home, Back and Recent) be accessible with a single action, which should be made clear to the user. [Describes terms of Alphabet's confidential agreements with OEMs], Alphabet also imposed conditions on how those buttons should behave in GMS requirements (Alphabet's reply to RFI PO 17 of 02 April 2026, Doc ID […], question 18).

582 [Description of Alphabet's confidential commercial agreements with OEMs], Alphabet's reply to RFI PO 17 of 02 April 2026, Doc ID […], Annex Q18.2, Doc ID […].

source code, invocation through the LPH/LPNH access point provides the invoked app with contextual data, including:[583]

(a) A screenshot of the current on-screen content that was shown on the device when the user invoked through LPH/LPNH;

(b) Additional contextual information provided by the foreground app, i.e. the app that the user was using just before invoking through LPH/LPNH; and

(c) Additional information on the UI of the foreground app, potentially including content that is not visible on screen, but could be reached through scrolling.

(450) Access to the contextual information provided upon invocation is an operating system feature which is implemented, controlled and managed by Google Android, for the following reasons:

(a) It is provided through a system API that is implemented in AOSP (as further detailed in Section 8.2.3).

(b) Collection of the contextual information (including information about the open app(s), screen context information, etc.) is conducted at the operating system level and only in a second stage transmitted to the dedicated service at app level; and

(c) Transmission of the collected information to the invoked app is done by the operating system.

(451) The ability to overlay content on top of the screen is also an operating system feature, which is rendered possible through an API implemented in AOSP.

*8.2.3. Technical implementation of contextual invocation in Google Android*

(452) On Google Android mobile devices, invocation through the LPH/LPNH access point is governed through the system launcher by the Contextual Search API package, which is implemented in the AOSP source code. This API is targeted at enabling the user to (visually) search online (or launch other functionalities, such as translation) for all or a portion of what is currently on screen.

(453) In the AOSP source code, the system launcher governs the device's navigation and action gestures, including the LPH/LPNH access points. By default, the system launcher is implemented to activate the Contextual Search package, i.e. invoke the configured app and provide contextual information to it, when using the LPH/LPNH access points.[584] The Compatibility Test Suite, which validates whether Google Android has been implemented on a device in line with the applicable technical requirements of the CDD, comprises a test of the implementation of the Contextual Search API invoked by the LPH access point.[585]

(454) The Contextual Search API package is implemented in the AOSP source code as a configuration overlay (Section 5.2.4), i.e. the source code is to be filled by the OEM with the app that will be the dedicated Contextual Search package app. This means that the AOSP source code allows the OEM to configure one (and only one) app as the Contextual Search package app, which will in turn allow it to receive the

583 Alphabet's reply to RFI PO 4 of 13 February 2026, Doc ID […], question 10.

584 The launcher can be implemented differently by the OEM.

585 See https://cs.android.com/android/platform/superproject/main/+/main:cts/tests/contextualsearch/src/android/contextualsearch/cts/ContextualSearchManagerTest.kt;l=80;drc=61197364367c9e404c7da6900658f1b16c42d0da, Doc ID […], accessed on 8 July 2026.

contextual data upon invocation.[586] After the OEM has configured one app as the Contextual Search package app, that app can only be replaced by another app via an update and integration work by the OEM.[587]

(455) The Contextual Search API is a system API, which means that, currently, only preinstalled apps (such as the Google app which is preinstalled on most Android mobile devices) can be configured as the Contextual Search package app by the OEM.

*8.2.4. Feature falls within the scope of Article 6(7) DMA*

(456) The Commission considers that the LPH/LPNH contextual invocation feature as described in Section 8.2.2 – together with its functionalities – is controlled via Google Android and falls within the scope of Article 6(7) DMA.

*8.2.5. Current access for Alphabet's own services and hardware*

(457) Alphabet's services can be (and are) invoked through a number of access points on Google Android devices. For example, Alphabet's service Gemini is invoked through the dedicated access point for the default assistant when it is selected by the user as such. Some access points are however not user-customisable or even reserved for Alphabet's services (as further explained below in the case of Google Search via the Circle to Search solution).

(458) On most recent Android mobile devices, the LPH/LPNH access point (or any other access point agreed upon in writing between Alphabet and the OEM) is used by Google Search via the Circle to Search solution, which was launched on 31 January 2024 as a new "*innovative interface to Google Search enhanced with its generative AI technology*".[588] Circle to Search is Google Search solution (which is provided through the Google app), which relies on a privileged access to the LPH/LPNH contextual invocation feature (i.e. access points and access to contextual information).

(459) On devices where Google Search's Circle to Search solution is enabled, the Google app is the dedicated Contextual Search package app. Therefore, the end user can invoke it through the LPH/LPNH access point (or any other access point agreed upon between Alphabet and the OEM). In addition, the Google app receives the contextual information and a screenshot as described in Section 8.2.2, in order to offer Circle to Search based on interaction with the device screen.[589]

(460) Such deep integration with the operating system is relied on by Alphabet to provide a seamless AI-powered experience to users. Alphabet mentions in one internal document the [direct quote describing Alphabet's commercial discussions][590]

(461) To enable Google Search's Circle to Search solution on devices, OEMs enter into a *Circle to Search Enablement Agreement* with Alphabet. Based on that agreement, OEMs can decide to implement Circle to Search on a device-by-device basis. [describes confidential terms of Alphabet's commercial agreements with OEMs].[591] From a technical perspective, this means that on devices where Circle to Search is

---

586 Alphabet's reply to RFI PO 4, Doc ID […], question 10.

587 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID […], question 25.e.

588 Alphabet's [description of Alphabet's commercial agreement terms with OEMs], Doc ID […], "Definitions" section.

589 Alphabet's response to RFI PO 6 of 23 February 2026, Doc ID […], question 25.j.

590 Alphabet internal document, Document [Alphabet internal document], Doc ID […].

591 Alphabet internal document, Document [Alphabet internal document], Doc ID […], Section 2.3.

enabled, OEMs must configure the Google app as the Contextual Search package app out of the box.

(462) As of February 2026, Circle to Search has been enabled on [Alphabet internal data] Android-compatible device models, including the Samsung S25 and S26, and the Pixel 9 and 10, i.e. the majority of Android devices released since January 2024.[592] As a result, the LPH/LPNH access point (as well as any other access point agreed upon between Alphabet and the OEM) and related contextual data is reserved for Google's own Circle to Search on these devices and not accessible by third-party apps.

(463) The Commission therefore considers the LPH/LPNH contextual invocation feature is available to and used by Alphabet within the meaning of Article 6(7) DMA.

*8.2.6. Current access for third-party services and hardware*

(464) Third-party services and hardware do not currently enjoy equal access to the LPH/LPNH contextual invocation feature as is available to Alphabet's own services.

(465) *First,* the feature is reserved to preinstalled apps. Indeed, the Contextual Search API, which Alphabet relies on to hardwire its service to dedicated access points, is a system API, and therefore only apps which have been preinstalled as a system app by the OEM can be configured as the Contextual Search package app. To be configured as the Contextual Search package app, third-party developers would need to negotiate a preinstallation agreement (which, as explained in Section 5.4 above, constitutes a significant barrier) and do some integration work with the OEM. As explained by Alphabet: [direct quote describing the function of OEM system launcher].[593] Importantly, third-party developers would need to repeat this exercise with each OEM in order to be integrated on a range of devices and brands. Conversely, the Google app is already preinstalled on virtually all Android-compatible devices[594] and therefore easily configurable by OEMs as the Contextual Search package app.

(466) *Second,* third-party developers are not in a position to approach OEMs and ask for their apps to be configured as the Contextual Search package app, since there is no documentation explaining how to integrate with the LPH/LPNH contextual invocation feature. Instead, Alphabet provides integration guidelines *to* those OEMs with which it has concluded a Circle to Search Enablement agreement.

(467) *Third*, the Contextual Search package is implemented as a configuration overlay, which means that only one app can be configured to be invoked through the LPH/LPNH access points and to receive the related contextual data. On devices where Circle to Search is enabled, [describes Alphabet's commercial agreements with OEMs]. On all [number of Android models involved] device models where that is the case, other apps therefore cannot rely on that feature.

(468) Once the Contextual Search package has been configured to one app, changing it to another app would require:

(a) that the user disables Google Search's Circle to Search solution;

---

592 Alphabet's reply to RFI PO 1 of 06 February 2026, Doc ID […], Annex Q9, Doc ID […].

593 Alphabet's reply to RFI PO 2 of 09 February 2026, Doc ID […], question 11.

594 According to Alphabet's reply to RFI PO 6 of 23 February 2026, Annex Q2, Doc ID […]. [90-100]% of all Android smartphones have the Google app preinstalled. The numbers account for at least [90-100]% for the "28-day" active Android smartphone device installed base. Multiple Android software builds may exist for the same device, as a device could be updated.

(b) that a third-party developer is aware and asks the OEM to be integrated with the functionality;

(c) that the OEM preinstalls the third-party app via an operating system update after device launch; and

(d) that the OEM configures the third-party app as the Contextual Search package.

(469) As a result, even if there exists a theoretical possibility that a user would turn off the feature on its device, there would be little incentive for a third-party to engage in discussions with various individual OEMs to negotiate pre-installation and integration with the access point on the off chance that some users turn off the solution.

(470) As explained by a third-party AI service developer: "*before the Circle to Search feature was introduced by Alphabet, that access point was customizable, and for instance an AI assistant of the choice of the user could be launched through that access point. Today, a user can disable Google's Circle to Search, but this does not give back to the user the possibility to customize the access point and to decide to launch another service through that access point. As a result, with the roll-out of Google's Circle to Search, third-party developers have lost the possibility to reach users through that access point*."[595]

(471) *Fourth,* third-party user-downloaded apps do not enjoy effective interoperability on par with Alphabet's services when it comes to access to screen content. In particular, possible alternative APIs to the Contextual Search API put forward by Alphabet[596] do not enable third-party providers to provide an equivalent user experience.

(472) Indeed, while the Assist API may give access to the same on-screen and contextual content as the Contextual Search, this API does not provide effective interoperability with the same OS feature. The Assist API is linked to the default assistant role and therefore enables the app selected as the default assistant app to receive similar contextual data upon invocation via the dedicated assistant access point, while the Contextual Search API enables invocation through the LPH/LPNH access point (or the other access point agreed upon between Alphabet and the OEM). When a third-party app is selected by the user as the default assistant, it cannot offer a (contextual) service through the LPH/LPNH access points (or other access points reserved for an Alphabet service), except in the very unlikely circumstances explained in recital (468) above. In addition, in cases where the user selects a default assistant that is different than Gemini, Alphabet can still rely on the Contextual Search API to provide a contextual solution on devices where Circle to Search is enabled, while the third party can only rely on the Assist API, which is linked to the default assistant status.

(473) Similarly, the Media Projection API is a public API that can be used by third-party apps to record the device screen. However, contrary to the Contextual Search API, the Media Projection API does not give access to contextual information associated with the screen content. This API is also subject to user permission every time the app wants to use it.[597]

(474) Finally, the Content Capture APIs provide continuous access representation of an app's current UI on the screen but only to intelligence components (Section 9.3), not

595 Agreed minutes of meeting with [AI service provider] on 8 December 2025, Doc ID […], paragraph 16.
596 Alphabet's reply to RFI PO 3 of 13 February 2026, Doc ID […], question 6.
597 Alphabet's reply to RFI PO 3 of 13 February 2026, Doc ID […], question 8.

to third-party services. It is also a system API, which is only accessible for pre-installed apps.[598]

(475) *Fifth*, more broadly, third parties have no access to the feature to the extent that access points are reserved to Alphabet's services without allowing the end user to customise the access point to another service. This is because, in such circumstances, a third-party service is required to successfully negotiate pre-installation and integration with individual OEMs in order to obtain ecosystem-wide access, which, as further explained in Section 4.6 above, does not provide effective interoperability.

*8.2.7. Alphabet's response to the Preliminary Findings*

(476) In its response to the PFs, Alphabet argued that mandating access to the LPH/LPNH contextual invocation feature exceeds the scope of Article 6(7) DMA.

(477) *In the first place*, Alphabet considered that it already complies with Article 6(7) DMA because both third-party and first-party services can gain access to the feature subject to OEM preinstallation and configuration.[599]

(478) *In the second place*, Alphabet argued that the alleged interoperability gap identified by the Commission is a strictly necessary and proportionate integrity measure which cannot be addressed by less restrictive means.[600] In particular, Alphabet claimed that access to contextual data should only be available to pre-installed apps or default assistants (via the Assist API) because it is appropriate to place limitations on features that allow apps to read the user's screen content (irrespective of whether the user agreed to it).

(479) *In the third place*, Alphabet considered that the draft measures annexed to the PFs are in any event disproportionate.[601] Alphabet contends that the measures go further than necessary in requiring Alphabet to make available APIs used by Circle to Search to "all apps" without limitation on the use cases or categories of beneficiaries. Alphabet also considers that the draft measures annexed to the PFs would give rise to minimal incremental benefits as there is an equivalent route to access the same data, and other prominent access points are available to downloaded apps.[602] The draft measures annexed to the PFs would additionally lessen OEMs' ability to control device configuration and system behaviour by requiring a standardised, user-switchable integration for Circle to Search.[603]

*8.2.8. The Commission's assessment of Alphabet's claims*

8.2.8.1. Concerning the existing level of interoperability and integrity

(480) Alphabet's *first claim* that there is no interoperability gap because third-party and first-party services can gain access subject to OEM preinstallation and configuration, is already addressed in Section 8.2.6, which Alphabet does not factually contest. As further explained in Section 4.6 above, Article 6(7) DMA requires actual "interoperability" and "access", not the mere possibility of obtaining feature access subject to conditions and importantly subject to the unilateral decision of individual OEMs. As such, the fact that OEMs can theoretically select to integrate the feature with a third-party app does not suffice to ensure effective interoperability across Google Android operating system, as evidenced by the fact that on [number of

598 Alphabet's reply to RFI PO 3 of 13 February 2026, Doc ID […], question 8.
599 Alphabet's response to the PFs, Doc ID […], paragraphs 185-187.
600 Alphabet's response to the PFs, Doc ID […], paragraphs 188-191.
601 Alphabet's response to the PFs, Doc ID […], paragraph 192.
602 Alphabet also quotes [OEM's] submission to the Public Consultation in this regard, Doc ID […].
603 [OEM] also raised similar concerns in its response to the Public Consultation, Doc ID […].

Android-compatible device models involved] Android-compatible device models in the EU, it is Alphabet's service which is integrated with the feature to the exclusion of third parties'.

(481) Alphabet's *second claim* that the interoperability gap identified by the Commission is in any event a strictly necessary and proportionate measure to protect integrity is also unfounded.

(482) *First,* to support its claim, Alphabet puts forward arguments which in fact relate to the protection of the user's privacy rather than protection of operating system integrity. In any event, as further detailed in Section 4.8.4 above, a gatekeeper cannot justify an integrity measure (or for that matter, a privacy measure, given that equally effective interoperability must be provided under equal conditions) implemented in relation to third parties' services or hardware solely based on whether the gatekeeper controls or trusts such third parties. In particular, a gatekeeper may not justify measures by the mere fact that third parties are not the gatekeeper, and therefore they cannot be trusted.

(483) In any event, Alphabet has not demonstrated that the measures would represent a material shift in the security model already applied by Alphabet. In fact, Alphabet does not restrict access to APIs giving access to similar contextual data or even more sensitive data in the same way as it currently argues would be justified for the Contextual Search API.

(a) The Assist API, which gives access to similar data to the default assistant upon invocation, can be accessed by any user-downloaded app that self-declares as an assistant and is selected by the user as the default. In fact, as shown by an Alphabet internal document,[604] [Describes Alphabet design choices and technical details regarding Circle to Search]. When Alphabet had to create a role for the default assistant and therefore give away the Assist API to third party services selected by the user for that role, it created the Contextual Search API as a way for Google Search to continue to access the same data. At the time, when considering whether to restrict access, Alphabet itself admitted: [direct quote from Alphabet internal document describing privacy and security considerations regarding the Assist API] (i.e. through the Assist API).[605]

(b) The Media Projection API is available to all user-downloaded apps without a limitation on beneficiaries, while giving access to more sensitive data, i.e. ongoing/continuous access to the user's screen content versus the instantaneous or "*one-off*" data provided by the Contextual Search API. Both APIs would be protected by user consent under the measures annexed to this Decision. While the Media Projection API additionally displays a prominent indicator of ongoing access in the status bar when recording the user's screen, such a protection is connected to the fact that it provides continuous access to screen content without the user actively initiating the recording, and would not make sense in the case of an API providing one-off screenshots upon activation by the user.

(484) *Second,* other less restrictive means to protect users against a misuse of the feature by malicious apps already exist. Indeed, Alphabet already offers developers, who they consider should "*play a critical role in ensuring the safety and security of [their] app*

604 Alphabet internal document, Document [Alphabet internal document], Doc ID […].
605 Alphabet internal document, Document [Alphabet internal document], Doc ID […].

*and its users*",[606] the possibility to protect the most sensitive user data from malware apps or mistriggered screenshots. A support web page for Android incentivizes Android developers to make use of specific "flags" that enable apps to block other apps or the system from taking screenshots of sensitive user content.[607] Banking apps, or password management apps, are regular users of those protections.

(485) App developers are in any event subject to GDPR and other data protection regulations (Section 4.8.2). In addition, the measures in the Annex to this Decision allow Alphabet to take any technical measure to ensure user consent is properly given, as well as to protect integrity.

8.2.8.2. Concerning proportionality

(486) Alphabet claimed that the measures for LPH/LPNH contextual invocation are disproportionate. However, Alphabet does not substantiate why the measures would be more restrictive than necessary to achieve the objectives of Article 6(7) DMA. In particular, Alphabet fails to propose any less restrictive, yet equally effective means of complying with that provision. Nor does Alphabet substantiate why the feature and/or a specific measure imposes a disproportionate burden on Alphabet. Instead, its arguments focus on questioning the scope of Article 6(7) DMA and the actual need for interoperability – a line of reasoning which does not meet the legal standard for a valid proportionality claim, as explained in Section 13 below.

(487) The Commission nonetheless addresses each of Alphabet's arguments in this respect.

(488) *First,* Alphabet considers that the feature should not be available to "all apps" without any limitations on the use-cases or categories of beneficiaries, but rather to search services, and that providing access to all apps would facilitate screen-scraping and violate GDPR data-minimisation principles.

(489) However, as further explained in Section 12.4 below, Article 6(7) DMA does not provide for any limitations as to the beneficiaries, apps, products and use cases for interoperability insofar as a feature is available to, or used by, Alphabet and intended to be used by a third party that is eligible under Article 1(2) DMA. In exceptional circumstances where effective interoperability requires ecosystem buy-in that is difficult to achieve if a feature were to be interoperable for all beneficiaries from the start, there might be reasons to restrict interoperability with a feature, at least in the initial stages, to a limited set of beneficiaries that meet certain conditions. However, as further explained in Section 12.4.5, the LPH/LPNH contextual invocation feature does not meet those conditions.

(490) In addition, the possibility to be invoked through access points is the most basic functionality a service should be able to access on Android devices in order to reach users: it is essential for ensuring visibility, user reach, and effective competition on the platform. The fact that invocation through some access points such as LPH/LPNH gives access to contextual data does not change that fact, nor does it make the functionality a search functionality. Alphabet itself does not only use the feature as a search functionality, but also as a translation and music recognition tool, embedded in the Google Search bar. Even if Alphabet were only using the feature as a search functionality, it should not pre-empt how third-party services might want to

606 Alphabet webpage, https://support.google.com/googleplay/android-developer/answer/14638385?hl=en, Doc ID […], accessed on 24 June 2026.

607 Alphabet webpage, https://support.google.com/googleplay/android-developer/answer/14638385?hl=en, Doc ID […], accessed on 24 June 2026.

make use of it. The contrary would undermine innovation by keeping it contained within the bounds of the gatekeeper's business decisions (Section 4.7).

(491) *Second,* the measures do not unduly lessen OEM's ability to control device configuration and system behaviour. As explained in paragraph (448), access points are operating system features. In fact, Alphabet codifies in the CDD how a number of access points, including the LPH/LPNH access point, should behave. The final measures do not introduce a paradigm shift in that respect. OEMs also remain free to configure access points out-of-the-box however they want, e.g. by selecting which apps are invoked through which access points, and to create new access points, including for marketing purposes. The final measures merely prevent Alphabet from reserving prime real-estate on Android devices for its own services, to the detriment of third-party services.

(492) *Third,* Alphabet's claim that the measures would only offer incremental benefits is ineffective. Article 6(7) DMA warrants that operating system features must be made accessible to third parties on par with Alphabet's services, without imposing any condition as to the importance or relevance of the feature. In any event, as already explained in paragraph (447) above, the LPH/LPNH is a particularly central access point, and ensuring third parties have an equal access to this operating system feature is therefore particularly important. This has also been highlighted by a developer during the Public Consultation, who explained: "*The proposal to open the long-press home button and navigation handle to third-party AI services is important. This system access point is significant for discoverability. Users should be able to designate any AI service for this entry point*".[608] Moreover, in order to ensure effective interoperability, Alphabet should not reserve any access point (be it the LPH/LPNH access point or any other access point) for its own services, as further explained in paragraph (496) below.

(493) The Commission therefore concludes that there is an interoperability gap with respect to the LPH/LPNH contextual invocation feature of Google Android. As a result, third parties and in particular user-downloaded apps, cannot perform important functionalities which places them sub-par in comparison with Google's own AI-powered services. As a result, the Commission considers that it should specify how Alphabet must fill this gap to comply with Article 6(7) DMA.

*8.2.9. Measures that Alphabet should implement*

(494) To meet the requirements of Article 6(7) DMA, Alphabet should implement the measures set out in Section 1.1 of the Annex to this Decision. The interoperability solution for the LPH/LPNH contextual invocation feature, as described in Section 8.2.2 of this Decision, must be equally effective as that available to any of Alphabet's own services.

(495) In order for third-party services to have access to the LPH/LPNH contextual invocation feature on par with Alphabet's services, third parties' apps must be invokable through the LPH/LPNH access point (or any other access point) in the same way as Alphabet's services (such as Google Search via the Circle to Search solution) can. This means that Alphabet should ensure that third-party services no longer enjoy a degraded interoperability with the LPH/LPNH access point and that they are able to be easily invocable through that access point (or any other access point agreed upon between Alphabet and OEMs). For that purpose, Alphabet should make available to third parties all the means for invocation that its services have

608 [Third Party's] submission to the Public Consultation, Doc ID […].

access to, as well as any other functionalities that its services are able to rely on upon invocation. This includes providing access to APIs such as the Contextual Search API, which enable invocation and the receipt of contextual data upon invocation, which third-parties currently only enjoy a degraded access to as further explained in Section 8.2.6 above. This also includes enabling to overlay on top of the device's screen in cases where that functionality is available to Alphabet. The contextual data should include any data that Alphabet has access to via the Contextual Search API or any other implementation of the contextual invocation feature, including from Alphabet's first-party apps.[609]

(496) To guarantee effective interoperability, Alphabet should not be permitted to reproduce the same conduct as described in Sections 8.2.5 and 8.2.6 above in relation to Google Search's Circle to Search solution on any other access point or through any other proprietary service. Alphabet therefore cannot reserve any access point for its own services, be it the LPH/LPNH access points or other access points, via contractual clauses, technical mechanisms or any other means, be it through the exact same conduct as described in Sections 8.2.5 and 8.2.6 above or through mechanisms achieving the same effects. The prohibition applies irrespective of the specific access point or the particular Alphabet service involved: otherwise, Alphabet could easily circumvent it by selecting a different service to be invoked through the access points currently reserved for Circle to Search, or by selecting a different access point through which to preferentially invoke Google Search or any other Alphabet service.

(497) This is without prejudice to one single access point being hardwired to the default assistant role. AI assistants are deeply integrated with the operating system, and the degree of integration is likely to grow given the depth of integration required to perform the agentic tasks. Just as some buttons are typically reserved to core OS interactions (e.g. power button lock/unlock the screen, volume buttons to change volume), reserving one access point for the default AI assistant is therefore consistent with the user's expectation of mobile device functionalities.

(498) It is also important, in order for interoperability to be effective, to ensure proper user agency. As a result, the access points should be user-customisable, i.e. that the user should be able to easily select which service is invoked through which access point. Alphabet should not use behavioural techniques such as complicated multi-step user journeys to disincentivize users to make this choice (Section 12.5). This is without prejudice to the OEM's freedom to design the out-of-the-box experience, i.e. being allowed to select which service is invoked through which access points when the user first activates the device.

(499) Finally, Alphabet claimed[610] that the draft measures also should not mandate that "any app" have access to the feature. Alphabet claims that such risks warrant that the feature should only be available to "eligible apps" through appropriate privacy/security controls. However, as further explained in paragraph (481) to (485) above, Alphabet did not put forward any integrity argument that would justify restricting access to the feature in such a way. In addition, as further explained in Section 12.4 below, Article 6(7) DMA does not provide for any limitations as to the beneficiaries, apps, products and use cases for interoperability insofar as a feature is available to, or used by, Alphabet and intended to be used by a third party that is eligible under Article 1(2) DMA, and the feature does not meet the conditions which, in exceptional circumstances, could warrant to do so.

609 See also Section 4.5.3.

610 Alphabet Second Mark-up of the draft measure, Doc ID […], page 3.

(500) For the purpose of ensuring that effective interoperability continues in the future, Alphabet should also provide effective interoperability with any future updates, including new functionalities, of the LPH/LPNH contextual invocation feature insofar as they are available to Alphabet's own services. To this end, the general measures in Section 12 of this Decision apply, including in particular the measures concerning future updates and new functionalities set out in Section 12.9 in this Decision.

*8.2.10. Timing for implementation of the measures*

(501) To meet the requirements of Article 6(7) DMA, and taking into account Alphabet's proposal and the available evidence,[611] Alphabet should implement the measures in the Annex (Section 1.1) according to the timelines indicated therein.

## 8.3. Measures for always-on hotword detection

*8.3.1. Introduction*

(502) End users expect that they can interact with modern AI services, including AI assistants, through their voice. These voice interactions can happen across various devices, such as smartphones, smart speakers, and wearables (e.g. headphones and smartwatches). Voice interactions should be seamless and allow hands-free communication. Hence, users would expect that they can wake and invoke their assistant by uttering a hotword (also called wake word) and then continue speaking to ask their question in a natural way, very much as in human interaction. Hotword-based invocation allows people to use their assistant whenever they want, even while driving a car, cooking, or while listening to music with their headphones on.

(503) Always-on hotword detection (AOHD) allows an AI service to be capable of continuously detecting a hotword or wake word and for the service to be invoked as soon as the hotword is uttered. To achieve this in an energy-efficient manner, AOHD relies on a digital signal processor (DSP), a chip that can execute specialised tasks such as hotword detection with low power consumption. Google Android has supported AOHD since version 5,[612] which Alphabet released in November 2014, with the first devices made by Google already supporting the feature in 2013.[613] Google's AI assistants Gemini and Google Assistant both support always-on hotword detection: an end user can invoke these assistants by uttering the "Hey Google" or "OK Google" hotword on most Android mobile devices sold in the EEA.[614]

(504) Third-party services do not enjoy the same level of interoperability with this Google Android feature. Crucially, they lack the ability to effectively have their hotword of choice detected by the on-device DSP chip. This means that, for example, user-

611 Technical meeting Alphabet/Commission of 12 June 2026, Doc ID […], slide deck page 13. Alphabet's reply to RFI PO 21 of 8 June 2026, Doc ID […], question 6, paragraph 17.

612 See [Alphabet internal document], Doc ID […], page 1, submitted as Annex Q3a.17 as part of Alphabet's reply to RFI AI 4 of 28 November 2025, Doc ID […], question 3. See also the AOSP source code, which has the copyright of the Always On Hotword Detector set to 2014, https://android.googlesource.com/platform/frameworks/base/+/94b4c163b7dfe5ce3607f7bb8456f9573f7de57d/core/java/android/service/voice/AlwaysOnHotwordDetector.java, Doc ID […].

613 See Gibbs, S. (2014), 'Motorola Moto X review: less "hello Moto", more hello Google', *The Guardian*, 17 January, https://www.theguardian.com/technology/2014/jan/17/motorola-moto-x-phone-review-google, Doc ID […], accessed on 8 July 2026.

614 Alphabet's reply to RFI AI 2 of 23 September 2025, Doc ID [...], question 33 and 34. [Describes Alphabet internal design choices regarding Google Search]. Furthermore, Google stated that the Google app is preinstalled on most Android devices in Europe. Alphabet internal document, [Alphabet internal document], Doc ID […], page 3.

downloaded third-party AI assistants do not have access to always-on hotword detection in the same way as Google AI Assistants have. They cannot offer the same performance to users, in particular in terms of battery consumption, which ultimately prevents a level playing field between Alphabet's services and third-party services.

(505) Developers have been asking for access to AOHD for a long time and, significantly, before the interoperability obligation under Article 6(7) DMA started to apply to Google Android. According to a developer, developers have been asking for interoperability for this feature since 2014.[615] More recently, [third party] has requested an interoperable solution for third-party apps in 2022 and met with Google engineers on this topic (see Section 8.3.6.4).[616]

(506) For Gemini, AOHD was at one point in time [Alphabet internal data on AOHD access point usage]. Depending on the device and OEM, until June 2024, between [20-30]% and [30-40]% of all requests to Gemini on Google Android mobile devices were initiated by hotword invocation.[617] As use cases for Gemini have developed, [Alphabet internal data on AOHD access point usage], but in absolute numbers, it is still a growing invocation method for Google Gemini (see Table F1 below). From the third quarter of 2025 to the first quarter of 2026, Alphabet reported an increase of [40-50]% to [number of invocations – Alphabet internal data on AOHD access point usage] invocations via AOHD.[618]

(507) Alphabet considers voice-based invocation a key feature for its AI assistants. The importance of voice-based invocation is echoed by Alphabet's strategy for deploying Gemini on Apple's iOS: [describes Alphabet commercial strategy and strategic discussions].[619]

*8.3.2. Description of the feature*

(508) AOHD is designed to run in an energy-efficient manner even though the detection is always running, including when the phone is in battery saver mode.[620] For this purpose, AOHD relies on a low-power chip on Google Android mobile devices: the DSP. The DSP is a specialised chipset for efficient processing of digital signals.[621] The DSP can continuously listen to the microphone, run small programs, and execute machine learning models called "sound models" that can detect if the user speaks a

615 See https://stackoverflow.com/questions/26834163/example-of-alwaysonhotworddetector-in-android, Doc ID […], accessed on 8 July 2026.

616 Agreed minutes of meeting with [third party] of [date], Doc ID […], paragraphs 15-20. Alphabet internal document, Document [Alphabet internal document] of 19 July 2022, Doc ID […], page 1ff. Alphabet internal document, Presentation [Alphabet internal document], Doc ID […], page 17.

617 Alphabet internal document, Presentation [Alphabet internal document], Doc ID […], page 5. Data as of 21 June 2024. Alphabet internal document, Document [Alphabet internal document], Doc ID […], page 3.

618 Alphabet's reply to RFI PO 15 of 27 March 2026, Annex Q3, Doc ID […].

619 Alphabet internal document, Document [Alphabet internal document], Doc ID […], page 10-12.

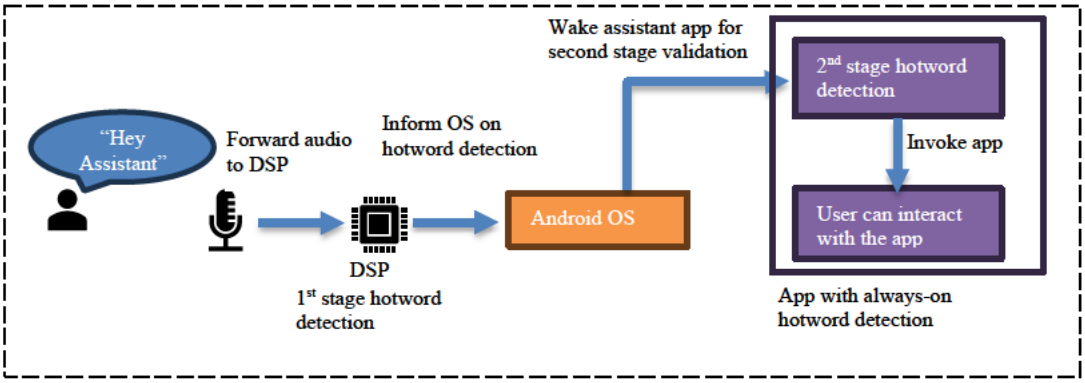
620 See https://developer.android.com/training/monitoring-device-state/doze-standby, Doc ID […], accessed on 8 July 2026.

621 The DSP is generally well suited for continuously running processing of data from device sensors such as the microphone; see also the CHRE discussed in Section (762).

hotword.[622] Most mobile devices nowadays have such a DSP chip, with devices offering low-power AOHD since 2013.[623]

(509) AOHD allows an app, such as an AI assistant, to be invoked and shown to the user when its hotword is uttered. For this purpose, Google Android maintains a mapping of hotwords and their registered sound models.[624]

**Figure 3. Overview of Always-on hotword detection on Android devices.**




*Source: Public documentation and internal documents*[625]

(510) AOHD on Google Android is implemented in two stages (see Figure 3):

(a) The first stage runs using a hotword detection sound model on the DSP. When the (low power) DSP detects the hotword, it wakes the main (high power) application processor and Google Android and forwards the audio that matched the hotword.

(b) Google Android passes the audio to the app linked to this hotword.[626] The app then runs a second stage hotword detection to verify that the correct hotword was detected. Two-stage verification is necessary, as the DSP sound models are built for energy efficiency and not for achieving the highest accuracy.[627]

(511) The functionalities of the AOHD feature available to the Google's AI assistants include, amongst others, the following:

(a) **Sound model enrolment.** The ability to enrol a sound model in the Android operating system to persistently store and retrieve the sound model upon

---

622 Alphabet internal document, Document [Alphabet internal document], May 2015, Doc ID […], page 2ff. Alphabet's reply to RFI AI 4, [case number], Annex Q3A.20, Integration Guide: Android System Hotword Enrollment App, Doc ID […].

623 See Gibbs, S. (2014), 'Motorola Moto X review: less "hello Moto", more hello Google', *The Guardian*, 17 January, https://www.theguardian.com/technology/2014/jan/17/motorola-moto-x-phone-review-google, Doc ID […].

624 Alphabet's reply to RFI AI 4 of 28 November 2025, Annex Q3A.20, Doc ID […].

625 Alphabet's reply to RFI AI 4 of 28 November 2025, Annex Q3a.20, Doc ID […]. See "Integration flows", https://source.android.com/docs/automotive/voice/voice_interaction_guide/integration_flows, Doc ID […]. AOSP source code documentation https://source.android.com/docs/core/audio/sound-trigger, Doc ID […].

626 A service on Android allows an app to execute code in background without showing the main app to the user. The Android documentation describes this concept here: See https://developer.android.com/develop/background-work/services, Doc ID […], accessed on 4 February 2026. For the open-source implementation of the hotword detection service see here: core/java/android/service/voice/HotwordDetectionService.java - platform/frameworks/base - Git at Google, Doc ID […], accessed on 20 April 2026.

627 Agreed minutes of meeting with [third party] of 18 March 2026, Doc ID […], paragraph11. Alphabet's reply to RFI PO 3 of 13 February 2026, Doc ID […], paragraph 49-50.

request. An enrolled sound model can be loaded on the DSP when the app starts AOHD;

(b) **First-stage detection on the DSP.** The ability to load and execute its enrolled sound model on the DSP, and to ensure that the DSP performs the first stage hotword detection on the app's behalf;

(c) **Second-stage validation.** The ability to run a second stage validation after the DSP sound model has recognised the hotword in the first-stage detection, to confirm that the hotword was detected correctly. The second stage may run in a secure and isolated process, disallowing to extract audio data before the hotword was validated;

(d) **Invocation of the app.** The ability of the app to be invoked and to handle the request of the user; and

(e) **Continued access to record audio.** The ability, after invocation, to receive the audio of the hotword detection and to continue to record the audio until the user decides to finish its request or conversation.

(512) By default, the AOHD implementation in Android is based on a sound model that was created by the app developer and that the app then provides and loads onto the DSP. This sound model is trained to detect one or multiple hotword phrases, e.g. "Hey Google". In addition to this implementation, [describes Alphabet's product design strategy and development for hotwords].[628] [describes Alphabet's product design strategy and development for hotwords]. The user would train the sound model with their own voice, and the final sound model would only react on the user's chosen hotword phrase and the user's voice. [Alphabet internal document discussing AOHD feature].[629]

(513) The Google AI Assistants also support AOHD on connected devices, such as headphones. [Alphabet internal document describing technical implementation of the AOHD feature].[630]

(514) Alphabet submitted that AOHD on connected devices is solely implemented via public Android SDK APIs that are interoperable with third-party apps. There are at least two companies, [AI service provider] and [AI service provider], that have both collaborated with manufacturers of connected devices to implement AOHD for their respective assistants.[631]

*8.3.3. Technical implementation of always-on hotword detection in Google Android*

8.3.3.1. Technical implementation of always-on hotword detection in the operating system

(515) Under Alphabet's current setup of Google Android, the use of the AOHD feature of the OS is subject to privileged permissions. As explained in Section 5.2 above, to use the feature, the OEM must grant an app access to privileged Android permissions, which requires preinstallation by an OEM. System-privileged Android permissions cannot be obtained by a user-downloaded app. A user who wants to switch their AI assistant to a user-downloaded assistant after they have bought their Google Android mobile device will not be able to find an assistant that offers the same functionalities as a preinstalled one.

---

628 Alphabet internal document, Document [Alphabet internal document], Doc ID […], page 20.

629 Alphabet internal document, Presentation [Alphabet internal document], Doc ID […], page 27.

630 Technical meeting Alphabet/Commission of 12 February 2026, Doc ID […], pages 19-24. Alphabet's internal document, Presentation [Alphabet internal document], Doc ID […], pages 11-15, 22-25.

631 Technical meeting Alphabet/Commission of 12 February 2026, Doc ID […], pages 19-24.

(516) To perform AOHD, an app requires the following privileged permissions:[632]

(a) *MANAGE_VOICE_KEYPHRASES* – Allows an app to enrol a hotword sound model in Google Android if the app is set as the default digital assistant;

(b) *KEYPHRASE_ENROLLMENT_APPLICATION* – Allows an app to enrol a hotword sound model even if the app is not set as the default digital assistant. Can be a substitute to the permission in (a).

(c) *CAPTURE_AUDIO_HOTWORD* – Allows an app to access the microphone through the hotword audio source. Required to access the microphone after the hotword has been detected;

(d) *MANAGE_HOTWORD_DETECTION* – enables hotword detection in a trusted process, i.e. an isolated process that cannot access the internet or other external resources;[633] and

(e) *SOUND_TRIGGER_RUN_IN_BATTERY_SAVER* – allows an app to use hotword detection when the phone is in battery saver mode.

(517) Provided that a pre-installation agreement between a developer and an OEM was concluded and the OEM granted the app access to these permissions (see Section 5.2), an app can implement the AOHD feature as follows:

(a) To enable AOHD, an app must first enrol a sound model and store the model on a persistent storage, in a registered models pool of the OS;[634]

(b) After an app has enrolled its sound model, it can use the *Always On Hotword Detector* system API. The app requests the OS to load an enrolled hotword sound model onto the DSP. The OS then loads the enrolled sound model onto the mobile device's DSP and starts the hotword detection. This API can only be used if the app is the default digital assistant;

(c) An app must also implement a *hotword detection service*[635] to handle the second stage hotword detection. In this service, the app receives the audio that triggered the hotword detection on the DSP. The app verifies that the hotword has been uttered and confirms the recognition.[636]

(518) The described implementation is based on the AOSP implementation, and it assumes that the Android device has a "DSP library" preinstalled, a software component that runs on the DSP and that can load and run the sound model that the app provides. As explained in Section 6.2.2.2, Alphabet's DSP library, which is preinstalled on all Google Android devices, is contractually prohibited from third-party usage. As the existing AOHD APIs are tied towards Alphabet's DSP library, third parties with OEM agreements had to implement a separate version of AOHD alongside the existing AOHD in Android (see Section 8.3.6.1). Therefore, the example above is

632 Alphabet's reply to RFI PO 3 of 13 February 2026, Doc ID […], question 22.

633 This is enabled by Private Compute Core, see Section 9.3.3.3.

634 See https://cs.android.com/android/platform/superproject/main/+/main:frameworks/base/media/java/android/media/voice/KeyphraseModelManager.java, Doc ID […], accessed on 4 February 2026.

635 The term service refers definition of a service in Android. An Android service is a piece of code that can run in the background. It can be part of an app to receive input from other apps or the OS. In this case the service receives input from the OS for the second stage hotword detection.

636 See https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/service/voice/HotwordDetectionService.java, Doc ID […], accessed on 4 February 2026.

hypothetical and has only been implemented in such a way in a released product by Alphabet as the provider of its assistant.

(519) The app then receives a confirmation that the hotword has been successfully detected. It can then start receiving audio from the microphone directly and process the user's request. This is usually paired with a user interface, showing that the app is active. Furthermore, the OS now activates the microphone indicator, demonstrating to the user that an app is actively listening and can use the microphone audio in any way (e.g. uploading it to a server for further processing).

(520) In Google's current implementation, the second stage runs in an isolated process.[637] This means that the audio received by this part of the app cannot leave this process before the second stage validation of the hotword detection. This is a privacy measure that ensures that all audio processing is happening in isolation on the device until the hotword has been validated. Therefore, the OS does not show the privacy indicator before both stages have validated a hotword detection.[638]

(521) Upon invocation using the Always On Hotword Detector API, the app receives contextual information. This information includes a screenshot of the device, the currently open apps, information provided by the currently used app (e.g. a URL opened in the browser), and a full low-level representation of the user interface of the currently used app. The user interface representation contains information presented on the screen, such as text, text fields, and buttons, in a machine-readable format. Google Android implements this functionality using the Assist API.

(522) [Describes Alphabet internal position regarding hotwords].[639] The actual number of concurrent apps that can run AOHD are limited by the memory and space available on the DSP.[640] The OS communicates with the DSP via the Sound Trigger Hardware Abstraction Layer (HAL), which can refuse to load additional sound models or stop running models, if the DSP does not have enough memory or processing resources available.[641] A typical optimized Alphabet sound model takes between [50-150] KB of space.[642] [third-party] explained that their current DSP models have up to [10-20] MB (i.e. [10 000-20 000] KB) of memory space available, which could therefore accommodate several sound models.[643]

8.3.3.2. Technical implementation of the always-on hotword detection on the digital signal processor

(523) To run the first stage hotword detection, a DSP sound model is required. The sound model is a machine learning model that can measure the probability that the hotword

---

637 Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID […], question 26, footnote 6.

638 Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID […], question 26, paragraph 61.a.

639 Alphabet's reply to RFI AI 4 of 28 November 2025, Doc ID […], question 8. For tasks beyond hotword detection, the CHRE is one example of an environment that enables additional tasks that run on the DSP (Section 9.4.3). Alphabet explained that on Pixel 10 devices, there [describes Alphabet's internal position regarding hotwords], suggesting that concurrency and capacity are not an insurmountable issue.

640 Agreed minutes of meeting with [third-party] [date], Doc ID […], paragraphs 15-20.

641 See Android AOSP source code, "The HAL implementation is free to reject loading of any model at any time by having the respective load*() method signal a ServiceSpecificException(RESOURCE_CONTENTION)", https://cs.android.com/android/platform/superproject/main/+/main:hardware/interfaces/soundtrigger/aidl/android/hardware/soundtrigger3/ISoundTriggerHw.aidl, Doc ID […], accessed on 20 March 2026.

642 The Commission described in the PFs its analysis of the Alphabet sound models found on a Google Pixel 9 Pro, a Google Pixel 10, a Samsung Galaxy S25. Alphabet confirmed the model sizes in Alphabet's response to the PFs, Doc ID […], paragraph 203.

643 Agreed minutes of meeting with [third party] [date], Doc ID […], paragraph 12.

has been uttered by the user; if this probability has exceeded a certain threshold, the hotword is considered as uttered and the second stage of hotword detection is launched. The DSP runs a piece of software, called the DSP library, that can interpret and execute the sound model. Alphabet maintains its own DSP library, called the Google Hotword Engine.[644] Alphabet's DSP library is part of the Google Android middleware (see Section 6.2.2.2) and must be preinstalled by all OEMs that integrate a DSP in their Android mobile device.[645]

(524) Since 2022, all Google Android devices that support the Google hotword use a Google Hotword Engine that supports the Tensor Flow Lite for Microcrontrollers model (TFLM)[646] format.[647] TFLM has been open sourced by Alphabet and has been specifically designed to run machine learning models on small microcontrollers and DSPs. This allowed Alphabet to standardise the sound models across DSPs from different chip vendors. The same sound model can be used for any DSP that supports TFLM.[648]

(525) Third-party and OEM apps possessing the relevant permissions could use the Google Hotword Engine to run their own sound models on the device's DSP. However, [describes Alphabet's commercial agreements].[649]

(526) According to Alphabet, the Google Hotword Engine can only run one sound model at a time.[650] However, likewise, Alphabet itself clarified that Google Pixel devices execute a second sound model on their DSP to detect if music is playing in the background. This feature does use a different DSP library, but only to execute different machine-learning operators needed for the music detection model.[651] In addition, Samsung is running multiple hotword [detail of Samsung's implementation] in parallel on (some of) its Google Android mobile devices (see Section 8.3.6.3). Therefore, there is no technical restriction to running several sound models in practice.[652]

*8.3.4. Feature falls within the scope of Article 6(7) DMA*

(527) The Commission considers that the AOHD feature as described in Section 8.3.2 – together with its functionalities – is controlled via Google Android and falls within the scope of Article 6(7) DMA.

*8.3.5. Current access for Alphabet's own services and hardware*

(528) The AOHD feature is available to and used by Alphabet via the preinstalled Google app. The Google app is preinstalled on [90-100]% of all Google Android

644 Alphabet's reply to RFI AI 2 of 23 September 2025, Doc ID […], DMA.100210, EMADA, Annex Q13.1, Doc ID […], Section 1.26.

645 Alphabet's reply to RFI AI 4 of 28 November 2025, Doc ID [...], [Alphabet internal document], Annex Q3A.18, Doc ID […], page 1.

646 See 'tensorflow/tflite-micro' (2026), tensorflow, https://github.com/tensorflow/tflite-micro,Doc, Doc ID […]. Tensor Flow Lite is also often named LiteRT, see LiteRT for Microcontrollers, https://ai.google.dev/edge/litert/microcontrollers/overview, Doc ID […], accessed on 27 April 2026.

647 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID […], question 29.

648 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID […], question 29, question 30.

649 Alphabet's reply to RFI AI 2 of 23 September 2025, Doc ID […], [Alphabet internal document], Annex Q13.1, Doc ID […], paragraph 4.11, page 17.

650 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID […], question 31.

651 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID […], question 33.

652 More broadly, Alphabet provides the Context Hub Runtime Environment (CHRE) for running small "nanoapps" on the DSP. See https://source.android.com/docs/core/interaction/contexthub, Doc ID […], accessed on 15 April 2026. On Pixel 10 devices, there are 25 nanoapps running in parallel. Alphabet's reply to RFI PO 16 of 1 April 2026, Doc ID […], question 9, paragraph 23.

smartphones that are actively used in the EU.[653] Alphabet's own AI services Google Assistant and Gemini are implemented via the Google app (see Section 5.3) and use AOHD. Both services have the capability to be invoked when the user utters one of the hotwords "OK Google" or "Hey Google".

(529) The access to AOHD feature to Google apps is enabled via the following mechanisms:

(a) As detailed in Section 5.4, through the EMADA, Google requires OEMs to preinstall Google's Core Applications and Core Services. The Google app is not an app whose preinstallation is compulsory, but if OEMs choose to do so, incentivised through placement payments, they must grant the app the permissions that Alphabet mandates;[654]

(b) Following the EMADA and GMS requirements,[655] OEMs must preinstall the following Core Services: Google Hotword Engine and the Google Play Services app. The Google Hotword Engine is required to enable AOHD for Alphabet's assistant apps on a Google Android mobile device;[656] and

(c) [confidential commercial agreement], Alphabet explains that OEMs must also preinstall the "Google Hotword Enrollment" app.[657] This app is used by Google Search to enrol the sound models with Google Android.

(530) For Google Pixel devices, the required apps and services are always preinstalled. There is no Pixel device that is sold without the Google app and the Google Hotword Engine preinstalled.

(531) The *Google Hotword Enrollment* app must be granted several privileged permissions by the OEM, including the permission to enrol sound models on behalf of another app.[658] The *Google Hotword Enrollment* app can enrol Google's DSP sound models in the OS.[659] However, current Android versions no longer require a separate enrolment app, as the APIs were updated to allow enrolment directly from a privileged and preinstalled app.[660] Although, it would not be necessary, Alphabet still uses the old model using an enrolment app.

---

653 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID […], Annex Q2, Doc ID […]. The numbers account for at least 90% for the "28-day" active Android smartphone device installed base. Multiple Android software builds may exist for the same device, as a device could be updated.

654 See Alphabet's reply to RFI AI 4 of 28 November 2025, Annex Q3a.1, Doc ID […]. Alphabet's reply to PO RFI 17 of 2 April 2026, Annex Q18.2, [Alphabet internal document]. Agreed minutes of the meeting with [OEM] on 09 March 2026, Doc ID […], paragraphs 7-9.

655 Alphabet's reply to PO RFI 17 of 2 April 2026, Doc ID […], Annex Q18.2, [Alphabet internal document], Doc ID […], 6.2 Preload GMS.

656 Alphabet's reply to RFI AI 2 of 23 September 2025, Doc ID […], Annex Q13.2, [Alphabet internal document], Doc ID […], Section 7.12, and Alphabet's reply to RFI AI 4 of 28 November 2025, Doc ID […], Annex Q3A.20, [Alphabet internal document], Doc ID […], page 1ff.

657 Alphabet's reply to RFI AI 4 of 28 November 2025, Doc ID […], Annex Q3A.20, Integration Guide: Android System Hotword Enrollment App, Doc ID […]. [Direct quote describing Alphabet's commercial guides and enforcement] page 10. [Direct quote describing Alphabet's commercial guides and enforcement] page 12.

658 Alphabet's reply to RFI AI 4 of 28 November 2025, Doc ID […], Annex Q3A.20, [Alphabet internal document], Doc ID […], page 4.

659 Alphabet's reply to RFI AI 4 of 28 November 2025, Doc ID […], Annex Q3A.20, [Alphabet internal document], Doc ID […], page 4.

660 Alphabet's reply to RFI AI 4 of 28 November 2025, Doc ID […], Annex Q3A.20, [Alphabet internal document], Doc ID […], page 7.

(532) Google offers the *Open Hotword Stresstest tool* that allow OEMs to perform testing of the Google Hotwords. However, OEMs are not required to perform these tests.[661]

(533) To improve the accuracy of the AOHD feature, Alphabet implements the Voice Match feature in their second stage hotword detection. Users can train the hotword detection based on their personal voice. This training can ensure that the assistant wakes only when the mobile device's owner speaks the hotword.

(534) Alphabet uses the same functionalities that are used for AOHD for other purposes. For example, on Google Pixel devices, Alphabet offers Now Playing to identify the artist and title of songs playing in the background.[662] This uses a sound model on the DSP to constantly check if music is playing in the background. If detected, the device will try to identify the song that is playing and save it for the user, so they can see which music they listened to throughout the day.

(535) AOHD is an important invocation point for Gemini, Alphabet's primary AI assistant and service. Until 2024, between [20-40 %] of all requests to Gemini on Google Android mobile devices (depending on device and OEM) were initiated by a hotword invocation, [describes Alphabet commercial and strategic discussions].[663]

(536) Alphabet noted that the distribution for invocations shows a downward trend for hotwords, with Gemini now mostly being invoked through the app icon.[664] However, this is a selective reading of the facts. Table 1 shows the invocations for the most used access points for Gemini. The numbers report an *increase* by [40-50]% to [number of invocations] invocations of Gemini via a hotword.[665] The trend of hotword invocation as a *percentage* of all invocations is indeed trending downwards,[666] but the *total* number of Gemini invocations is growing strongly, and the *absolute* number of invocations via AOHD still continues to grow.

661 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID […], question 32.

662 This is provided by Android System Intelligence (see Section 9.3).

663 Alphabet internal document, Presentation [Alphabet internal document], Doc ID […], page 5. Alphabet internal document, [Alphabet internal document], Doc ID […], page 3.

664 Meeting Alphabet/Commission of 24 March 2026, Doc ID […], slide deck page 19. Alphabet's reply to RFI 15 of 27 March 2026, Annex Q3, Doc ID […].

665 Alphabet's reply to RFI 15 of 27 March 2026, Annex Q3, Doc ID […].

666 Meeting Alphabet/Commission of 24 March 2026, Doc ID […], slide deck page 19.

**Table 1 shows the invocation mechanisms for the Gemini assistant. The first line shows the total invocations in millions, the second line the distribution share. Other invocation methods include *Corner Swipe, Widget, and unknown.***

| | App Icon | Hotword | Long-press Power | Long-press Home | Other |
|---|---|---|---|---|---|
| **2025 Q3** | […]<br>[20-30]% | […]<br>[10-20]% | […]<br>[20-30]% | […]<br>[10-20]% | […]<br>[20-30]% |
| **2025 Q4** | […]<br>[40-50]% | […]<br>[10-20]% | […]<br>[20-30]% | […]<br>[10-20]% | […]<br>[5-10]% |
| **2026 Q1** | […]<br>[50-60]% | […]<br>[5-10]% | […]<br>[10-20]% | […]<br>[5-10]% | […]<br>[5-10]% |

*Source: Alphabet's reply to RFI 15.*[667]

(537) Both invocation methods, app icon and hotword, serve different use cases for the end user. Hotword-based invocation is highly useful in scenarios where the user wants to quickly access Gemini, use it as an assistant and interact in a hands-free way. Opening Gemini through the app icon grants access to the full-fledged Gemini app, which offers a wide range of chatbot functionalities such as multimodal search (i.e. with images, audio and video), research, and programming, beyond assistant-like functionalities. The two invocation methods are therefore highly complementary.

(538) As the provider of Google Android, Alphabet could even ensure that its services, including Gemini, have access to this feature on all Google Android devices, including those where Gemini is not pre-installed, by simply changing the Google Android access rules applicable to the privileged APIs used to implement the feature (see Section 5.2.1).

*8.3.6. Current access for third-party services and hardware*

(539) Third parties currently do not have equally effective access to the AOHD feature. users cannot invoke a user-downloaded third-party app with a voice command with the same degree of effectiveness and efficiency, in particular in terms of power consumption and always-on availability.

(a) Third-party user-downloaded apps cannot access the AOHD feature and the linked APIs, since they are guarded by privileged permissions subject to the app being preinstalled by the OEMs. OEMs are also prohibited through the CDD to enable any sort of hotword detection for user-downloaded apps (see Section 8.3.6.1).

(b) Even preinstalled apps cannot rely on the Google Android APIs for AOHD, as the APIs are not standardised, but tied to Google's DSP library, the Google Hotword Engine.[668] OEMs [describes Alphabet's commercial agreement terms

667 Alphabet's reply to RFI 15 of 27 March 2026, Annex Q3, Doc ID […].

668 The Android source code is designed to be hardwired to the Google Hotword Engine. the code shows even if different modules are loaded (i.e. different hotword engines), it will always select the first module (see line 110, .findFirst()). Even if an OEM or DSP vendor would offer a replacement engine, the Android source code does not allow to use this engine. See Android source code

with OEMs], requiring preinstalled apps to build parallel implementations of AOHD (see Section 6.2.2.2 and Section 8.3.6.1).

8.3.6.1. Third-party services have very rarely gained the possibility to use hotword detection

(540) Under the current contracts with Alphabet, Android OEMs do not have the freedom to make available an API for user-downloaded third-party apps to use the AOHD feature. Only preinstalled apps may implement hotword detection. Indeed, this is explicit in the Android CDD (see Section 5.1.1) that all OEMs need to sign to market an Android certified device: "[i]f *Handheld device implementations support the System API HotwordDetectionService or another mechanism for hotword detection without mic access indication, they […] MUST NOT allow a user-downloaded application to provide the hotword detection service."*.[669] This seems to be a deliberate strategy by Alphabet not to make hotword accessible to third-party apps.

(541) OEMs have only rarely enabled third parties to use AOHD via the pre-installation mechanism. Third parties and OEMs provided data showing that during the last five years (from 2021 to June 2026) only *two* third parties excluding OEMs, being Amazon and Perplexity, had obtained access to the AOHD with a small amount of OEMs. Amazon has given up access to this feature in 2024.[670]

(542) It is also telling that for its own Pixel mobile devices, Alphabet has not granted access to the AOHD feature to *any* third-party service.[671]

(543) There have been only few cases where an OEM preinstalled a third-party or its own OEM app and granted this app the required privileged permissions for AOHD. The Commission is aware of the following examples:

(a) Amazon collaborated with five OEMs for the "Alexa built-in" program. In total 14 phone models with AOHD for Amazon Alexa have been released. Amazon stopped the program in March 2023 and AOHD for the "Alexa" hotword was disabled on those devices via a software update;[672]

---

https://android.googlesource.com/platform/frameworks/base/+/45034f0663f960d9ee5fb0a101a4732b71f6e2f4/media/java/android/media/soundtrigger/SoundTriggerManager.java#105, Doc ID […].

669 The hotword detection service refers to the AOHD feature of the Android OS. The CDD describes it as "*Android, through the System API VoiceInteractionService supports a mechanism for secure always-on hotword detection without mic access indication and always-on query detection, without mic or camera access indication."* Android 16 Compatibility Definition, https://source.android.com/docs/compatibility/16/android-16-cdd, Doc ID […], Section 9.8.

670 Amazon and Perplexity were able to use AOHD on selected Android phones, see [AI service provider's] reply to RFI of 20 February 2026, Doc ID […], question 2. See Bishop, T. (2022), '"Alexa Built-In" phones to lose key feature as Amazon cedes ground to Apple and Google', GeekWire, 17 November, https://www.geekwire.com/2022/alexa-built-in-phones-to-lose-key-feature-as-amazon-cedes-ground-to-apple-and-google/, Doc ID […], accessed on 27 February 2026.

671 Alphabet stated that any third-party could request Pixel to grant system privileges for its apps. These requests are rarely received and typically only by network operators. See Alphabet's response to Commission RFI of 23 September 2026, Doc ID […], question 4. The Commission did not find any instances where a third-party app on a Google Pixel phone was using the AOHD feature.

672 See Fatmi, N. S. and Maring, J. (2021), 'Best Alexa built-in phone 2022', Android Central website, 13 December, accessed 3 July 2026, https://www.androidcentral.com/best-alexa-built-phone, Doc ID […]. See Bishop, T. (2022), '"Alexa Built-In" phones to lose key feature as Amazon cedes ground to Apple and Google', GeekWire, 17 November, https://www.geekwire.com/2022/alexa-built-in-phones-to-lose-key-feature-as-amazon-cedes-ground-to-apple-and-google/, Doc ID […], accessed on 27 February 2026.

(b) The newest Samsung Galaxy S26 series of devices integrate with three assistants, all supporting AOHD simultaneously: Gemini, Bixby (from Samsung itself), and Perplexity;[673] and

(c) [A third party pursued the creation of a generalised AOHD API for third-party developers on Android phones. This third party has met with Google engineers to discuss this matter and offered solutions. They did not launch a working solution towards other third-party developers at the time of the adoption of the decision.][674]

(544) Each of the third-party solutions for AOHD did not rely on the existing AOHD APIs in Android for two reasons:

(a) The Android APIs for AOHD are tied to and use by default [describes Alphabet's commercial agreement terms with OEMs] Google Hotword Engine, Alphabet's DSP library (see Section 6.2.2.2).

(b) The Always On Hotword Detector API disallows concurrent hotword detection. As all third parties wanted to enable concurrent detection, e.g. being able to launch Alexa alongside Alphabet's assistants, they could not use what was available in Android.

8.3.6.2. Amazon Alexa always-on hotword detection

(545) Amazon's "Alexa built-in" program was a collaboration with five Android OEMs that would sell smartphones with the Alexa app preinstalled and AOHD for the "Alexa" hotword. Amazon maintained the AOHD on Android for about four years and ultimately stopped the support for it in 2023.[675] [Further information on Amazon Alexa][676]

(546) [Further information on Amazon's collaboration with several parties to implement their own version of AOHD on Android as access to Alphabet's implementation was not possible due to technical limitations][677]

(547) [Additional details on what efforts such an implementation entails. Android updates would require updates to Amazon's AOHD implementation.].[678][679]

(548) In 2019, Amazon launched the Voice Interoperability Initiative (VII) and released a series of whitepapers and design documents. This initiative had the goal to support multi-agent devices, where each agent would have a different hotword for voice

673 See Samsung (2026), 'Galaxy AI Expands Multi-Agent Ecosystem To Give Users More Choice and Flexibility', Samsung Newsroom, 22 February, https://www.samsung.com/ae/news/local/galaxy-ai-expands-multi-agent-ecosystem-to-give-users-more-choice-and-flexibility/, Doc ID […], accessed on 27 February 2026, Doc ID […], accessed on 27 February 2026. See also Samsung (2026), 'Samsung Unveils Galaxy S26 Series: The Most Intuitive Galaxy AI Phone Yet', Samsung Global Newsroom, 26 February, https://news.samsung.com/global/samsung-unveils-galaxy-s26-series-the-most-intuitive-galaxy-ai-phone-yet, Doc ID […], accessed on 27 February 2026.

674 Agreed minutes of meeting with [third party] of [date], Doc ID [...], paragraphs 15-20. Alphabet internal document, presentation [Alphabet internal document], Doc ID […], slide deck 1ff.

675 See Bishop, T. (2022), '"Alexa Built-In" phones to lose key feature as Amazon cedes ground to Apple and Google', GeekWire, 17 November, https://www.geekwire.com/2022/alexa-built-in-phones-to-lose-key-feature-as-amazon-cedes-ground-to-apple-and-google/, Doc ID […], accessed on 27 February 2026.

676 Agreed minutes of meeting with [third party] of [date], Doc ID […], paras 5, 6, 7, 13, 14.

677 Agreed minutes of meeting with [third party] of [date], Doc ID […], paras 5, 6, 7, 13.

678 Agreed minutes of meeting with [third party] of [date], Doc ID […], para 14.

679 Agreed minutes of meeting with [third party] of [date], Doc ID […], para 12.

activation.[680] More than 80 companies joined the initiative, including manufacturers of smart speakers, such as Sonos, Harman/Kardon; technology companies, such as Microsoft, Facebook, and Intel; DSP manufacturers, such as Qualcomm and MediaTek; and even some Android OEMs, such as Xiaomi, and Acer.[681] Importantly, Alphabet did not join this initiative,[682] but released a handful of smart speakers which supported Google Assistant and Amazon Alexa running simultaneously.[683] This multi-agent speaker was made to promote user choice, so the user can use each assistant for the tasks it can do best, or for households where different people prefer different assistants.[684]

(549) The example of Amazon's implementation of AOHD in Android shows that it required significant efforts to integrate this, mainly because the Android implementation lacks standardised APIs. In the end, without ever reaching a sustainable market share,[685] even one of the largest companies in the world had to abandon the project. While on the surface, the privileged system APIs in Android look like they could be used by third parties, they fail at the integration. The APIs, as they are built today are tailored to support Google's AI assistants and not third parties. While Alphabet creates a working DSP library could allow third parties to use AOHD today, if preinstalled, [describes Alphabet's commercial agreements],[686] rendering the AOHD implementation in Android useless for third parties.

8.3.6.3. Samsung always-on hotword detection for Bixby and Perplexity

(550) Samsung implements AOHD for their own Bixby assistant. The Bixby assistant's hotword ("Hi Bixby") can run simultaneously to the Google hotword and allows the user to use both assistants to their liking.[687] [Description of Alphabet's commercial agreement terms with OEMs and further details on Samsung's Bixby.][688]

(551) Samsung also offers a feature that allows the user to create a user-customisable hotword for Bixby. The feature is not enabled by default: the user must enable it as part of a set of "Bixby Labs" beta features.[689] The user-customised hotword, allowing a user to use almost any phrase as a hotword, continues to run concurrently

---

680 See Amazon (n.d.), 'Voice Interoperability Initiative', Amazon Alexa website, https://developer.amazon.com/en-US/alexa/voice-interoperability.html, Doc ID […], accessed 16 April 2026.

681 See Amazon (n.d.), 'Voice Interoperability Initiative Members', Amazon Alexa website, https://developer.amazon.com/en-US/alexa/voice-interoperability/members.html, Doc ID […], accessed 16 April 2026.

682 Alphabet internal document, Document [Alphabet internal document] of August-September 2023, Doc ID […], page 8.

683 Alphabet internal document, Document [Alphabet internal document] of August-September 2023, Doc ID […], page 1, 2. Alphabet internal document, Document [Alphabet internal document] of 30 September 2022, Doc ID […], page 1, 2.

684 Alphabet internal document, Document [Alphabet internal document] of August-September 2023, Doc ID […], page 1.

685 Amazon only reached preinstallation agreements with five OEMs, not including important such OEMs Samsung, Xiaomi, and Google. As the project was only launched on 14 smartphones, it could not reach a sustainable market share.

686 [Direct quote describing Alphabet's commercial agreements], excerptAlphabet's reply to RFI AI 2 of 23 September 2025, Doc ID […], EMADA, Annex Q13.1, Doc ID […], Section 4.11(a).

687 This feature was also tested by the Commission on a Samsung Galaxy S25. Agreed minutes of the meeting with [OEM] of [date], Doc ID […], paragraph 9.

688 [additional technical details]. Agreed minutes of the meeting with [OEM] of [date], Doc ID […], paragraph 2.

689 "You can set a custom wake word for Bixby on the Galaxy S25, but will anyone care?" by Mishaal Rahman, 06 February 2025, https://www.androidauthority.com/bixby-custom-wake-word-3523642/, Doc ID […], accessed 26 February 2026.

with the Google hotword. The final user experience is equal to the one provided by Gemini on Google Android mobile devices.[690]

(552) [Further information on custom hotword feature in Samsung devices.].[691]

(553) Samsung also granted Perplexity access to its AOHD feature for the Perplexity assistant. [Additional information on the collaboration between Perplexity and Samsung.].[692]

(554) Perplexity is today the only third-party assistant that is not made by an OEM, with access to the AOHD feature on Google Android mobile devices. Other AI service providers attempted to obtain access to the AOHD feature from OEMs but failed during the commercial negotiations. [Additional information about commercial negotiations.] .[693]

(555) [Additional information on Perplexity's hotword on Samsung devices.][694] Their experience shows that many third parties would be capable of creating a working sound model for AOHD on Google Android, if not restricted by the operating system.

(556) [Additional details on Perplexity's hotword on Samsung devices.].[695]

(557) The example of the AOHD integration with Samsung devices demonstrates that the third party could not rely on the existing system privileged APIs. Again, the third-party had to go the route of a parallel AOHD implementation, sitting next to the already available AOHD APIs of Android. As Samsung, the largest OEM in Europe, offered their own assistant Bixby in parallel to the Google AI assistants for a long time, Samsung had the experience for building a separate AOHD API and to maintain it over years. This considerable effort would usually not be done by other OEMs who do not maintain their own assistant. Therefore, third parties would have to go the same route as Amazon did for other OEMs (see Section 8.3.6.2).

8.3.6.4. [Third-party's] initiative to implement an open always-on hotword detection API for third-party developers

(558) [Third party] started an initiative to implement an open API for third-party Android developers for AOHD. This API would have been available on Google Android mobile devices [details on the eligible Android devices and how it was planned to be implemented][696]

(559) [Third party] planned to preinstall [a custom] app with access to privileged permissions on these devices.[697] Third-party app developers would then use an API exposed by [the app] to enrol sound models and register for AOHD. If the app's

690 "You can set a custom wake word for Bixby on the Galaxy S25, but will anyone care?" by Mishaal Rahman, 06 February 2025, https://www.androidauthority.com/bixby-custom-wake-word-3523642/, Doc ID […], accessed 26 February 2026.

691 Agreed minutes of the meeting with [OEM] of [date], Doc ID […], para 1-3.

692 Agreed minutes of meeting with [AI service provider] of [date], Doc ID […], paragraph 9, 10, 12. [AI service provider's] reply to RFI of 20 February 2026, Doc ID […], question 2.

693 [AI service provider's] reply to RFI of 20 February 2026, Doc ID […], question 3.

693 [AI service provider's] reply to RFI of 20 February 2026, Doc ID […], question 3.

694 Agreed minutes of meeting with [AI service provider] of [date], Doc ID […], paragraph 23.

695 Agreed minutes of meeting with [AI service provider] of [date], Doc ID […], paragraph 27.

696 Agreed minutes of meeting with [third party] of [date], Doc ID […], paragraphs 15-20. Alphabet internal document, Presentation [Alphabet internal document], Doc ID […], slide deck 1ff.

697 Agreed minutes of meeting with [third party] of [date], Doc ID […], paragraphs 15-20. Alphabet internal document, Presentation [Alphabet internal document], Doc ID […], slide deck 1ff.

hotword would be detected, the [third party's] app would invoke the third-party app to allow further interaction.

(560) According to the CDD, user-downloaded apps cannot get access to the AOHD feature on Android. Therefore, [third party] would not have been allowed to offer a public API to third parties without prior changes from Alphabet. In 2022, [third party] reached out to Google with the idea to implement this API for developers that could enable AOHD for third-party apps on Google Android mobile devices [details on the eligible devices].[698] [description of Alphabet's interaction with third parties].[699]

(561) [Third party] noted that the discussions with Alphabet did not lead to any public solutions in Android.[700] However, [third party] seemed to have continued their process to build an AOHD solution that could become available to third parties in the future. On some Android smartphones [details on eligible Android devices] [the app] is indeed preinstalled and granted with the privileged permission to capture audio after a hotword detection. However, while the app is available, there are no known APIs for third-party developers to use [third party's app] for AOHD.[701] [describes alphabet's commercial relationships with third parties][702]

(562) In practice, [third party] could release an interoperable implementation of AOHD for third-party developers. Again, [the third party] would not have implemented this feature using the existing Android AOHD APIs. [Third party] would also not have relied on Google's DSP library, [additional details on the implementation plan's of the third party]. [Third party] aimed to implement the entire AOHD through its [app] that would have offered an API for third-party developers. The main obstacle seems to be the lack of cooperation from Alphabet.[703] [Description of alphabet's commercial and strategic discussions]. Yet, without policy changes to the Android CDD from Alphabet, [the third party's] API would not be allowed.

8.3.6.5. Alternative solutions do not provide equally effective interoperability

(563) Alternative solutions do not provide equally effective interoperability to third-party downloaded apps. For example, using the public Android SDK APIs, an app could permanently listen to the device microphone and analyse the audio to detect a hotword.[704] This implementation comes with several weaknesses:

(a) It will use more battery, as it cannot rely on the battery-saving DSP. The phone would always stay awake and use the main application processor to analyse the

698 Agreed minutes of meeting with [third party] of [date], Doc ID […], paragraphs 15-20. Alphabet internal document, Presentation [Alphabet internal document], Doc ID […], slide deck 1ff.

699 In internal meeting notes Alphabet stated [direct quote describing Alphabet's commercial terms and strategic discussions], Alphabet internal document, [Alphabet internal document], Doc ID […], page 2.

700 Agreed minutes of meeting with [third party] of [date], Doc ID […], paragraphs 15-20.

701 The Commission analysed the public software packages of an [OEM] available on their website, see [URL], Doc ID […], accessed 09 April 2026. The Android software package contains an allowlist for the [third party's] app that grants it the CAPTURE_AUDIO_HOTWORD permission. The app is also listed on the Google Play Store, but it cannot be downloaded. See [URL], Doc ID […], accessed 09 April 2026.

702 Agreed minutes of meeting with [third party] of [date], Doc ID […], paragraphs 15-20.

703 Agreed minutes of meeting with [third party] of [date], Doc ID […], paragraphs 15-20.

704 See No Way, Google: Build Your Own Wake Word Service on Android | by Ian Lavery | Picovoice | Medium by Ian Lavery, 15 April 2026, https://medium.com/picovoice/no-way-google-build-your-own-wake-word-service-on-android-339a0189ff4c, Doc ID […], accessed 26 February 2026.

audio. After implementing hotword detection via these APIs, a third-party developer observed a 15x increase in their app's battery consumption;[705]

(b) It will show a privacy indicator, a small icon on the top right that shows that the app is permanently listening to the microphone. In contrast, AOHD using the DSP will not show a privacy indicator, as the app only gets access to the microphone audio when the hotword was detected. A privacy indicator may be appropriate when no privacy guarantees can be given. An always visible privacy indicator also devalues the user experience, as Alphabet acknowledges,[706] since the user can no longer distinguish whether the microphone indicator is always on due to AOHD, or due to an app accessing the microphone in the background for another purpose; and

(c) It requires running a foreground service, which will show a persistent notification to the user to inform them that the app is running in the background. The AOHD feature that is used by Alphabet does not show a persistent notification. A foreground service enables an Android app continued background execution. Depending on the OEMs implementation of battery optimisation, a foreground service may still be terminated to save battery, which would terminate access to the microphone.

8.3.6.6. Third-party services have no meaningful way for interoperability with always-on hotword detection on Android

(564) Currently, the only possibility for third parties to get access to the AOHD feature is to collaborate with OEMs directly through preinstallation and additional work due to a lack of standardised Android APIs.[707] This collaboration requires bespoke work from the third party with each individual OEM, and potentially with multiple DSP manufacturers, and separate commercial agreements, linked to payments from the third party to the OEMs and other parties. At the same time, this implementation has an important limitation from a commercial point of view because, even if successful, access has only ever been granted for a small set of Google Android mobile devices of each OEM thereby causing different experiences across Android ecosystem. Therefore, deploying a competitive product that uses AOHD across the Android user base has not been successful so far and will likely never be successful under the current model.

(565) As already established in Section 8.3.2 and Section 8.3.3, AOHD is a feature of Google Android, which includes the functionalities defined above and all implementations from the APIs in AOSP to the Google Hotword Engine, which is part of the Android middleware. The access for third parties could be more open, while the current system looks to be designed deliberately to not allow third parties to use AOHD. An interoperable solution could be implemented without changing much of the underlying implementation, see Section 8.3.8.

*8.3.7. Commission's assessment of Alphabet's response to the Preliminary Findings*

(566) In the response to the PFs and in the course of these proceedings, Alphabet first argued that it already allows for an interoperable solution for third parties and put forward a number of arguments related to the implementation of an interoperable solution for AOHD, intellectual property, and the claim that Android is already

705 Agreed minutes of the meeting with [third party] of 18 March 2026, Doc ID […].

706 Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID […], question 26, paragraph 61.a.

707 Agreed minutes of meeting with [AI service provider] of 26 March 2026, Doc ID […], para 5-15

interoperable as third-party service providers can implement AOHD by reaching out to OEMs.

8.3.7.1. Concerning the existing level of interoperability and integrity

(567) *First,* Alphabet argued that Google Android already "*allows interoperability for first-party and third-party services under equal conditions*"; because preinstalled apps have access to all functionalities of AOHD on Android subject to third-party developers working with OEMs and DSP chip developers.[708]

(568) As established in Section 6.3, the Commission considers that the model which Alphabet employs to delegate to OEMs the decision-making as regards whether a third-party obtains interoperability does not amount to effective interoperability within the meaning of Article 6(7) DMA.

(569) Even if OEMs agree to preinstall a third-party app, the issues and complexities for third parties to implement a version of hotword detection also boil down to the restrictions put in place by Alphabet. The Google Android AOHD APIs are hardwired to the Google Hotword Engine, as third parties are not allowed to use the engine, meaning they had to implement a custom version for hotword detection, in parallel to the existing Android APIs (see Section 8.3.6).

(570) Therefore, Alphabet is right when they state that the current system requires close collaboration between the OEM, and sometimes even the DSP manufacturer. However, this system is only in place because Android today does not provide an effective interoperability solution with standardised APIs, documentation, and access to third parties.

(571) *Second,* Alphabet submitted that the requirements for preinstallation for AOHD are strictly necessary and proportionate integrity measures, and the corresponding risk could not be addressed with less restrictive means, meaning that the current system whereby the third parties need to work with OEMs is necessary.

(572) As established in Section 6.3, the model which Alphabet employs to delegate to OEMs the decision-making as regards whether a third party obtains interoperability with any feature that is part of this Decision is not strictly necessary and proportionate, especially as it does not ensure that only strictly necessary and proportionate integrity measures are taken by the OEMs.

(573) Alphabet attempted to obtain support for the current implementation with [third parties] in the context of the public consultation. To this end, [details on Alphabet confidential exchanges with third parties whereby Alphabet highlighted to third parties its concerns on the draft measures] for them to address in their submission to public consultation.[709]

(574) For instance, [third party], [description of third party's interactions with Alphabet regarding how to reply to the public consultation] in its submission to the public consultation.[710] [Third party also shared its draft submission with Alphabet and with

---

708 Alphabet's response to the PFs, Doc ID […], paragraph 201.

709 On 7 May 2026, […], Google's public counsel, sent the following lines to [third party]: [direct quote from Alphabet's counsel's communications with third parties regarding the impact of the draft measures on third party developers], Alphabet's response to RFI PO 20, Annex Q1C.59, Doc ID […]. [Third party's] submission contained the following sentence: "*[direct quote from public consultation submission]*.", Doc ID […].

710 On 7 May 2026, […], Google's public council, sent the following lines to [third party]: [Direct quote from Alphabet's counsel's communications with third parties regarding the impact of the draft measures on third party developers], Alphabet's response to RFI PO 20, Annex Q1C.59, Doc ID […]. [Third

[third party],[711] [description of interactions with third party concerning its response to the Public Consultation], prior to addressing it to the Commission.[712]

(575) As a result, in the public consultation, [third party] submitted that the draft measures would enable [system level risks].[713]

(576) The basis of [third party's] assumptions as regards the draft measures is wrong. [Third party's] comments went beyond the proposed measures and [description of third party submission and Alphabet's communications with third party].[714] In contrast, the draft measures do not speak of [details of Alphabet's interactions with third party], but solely mention running third-party sound models, i.e. machine learning models, which are not code. A machine learning model can only run in a very constrained environment and works mostly using mathematical computations. In addition, the Commission's draft measures aimed to rely on Alphabet's DSP library, which exhibits the [details on Alphabet's interactions with third party] that [third party] demanded. [Third party's] submission criticises the draft measures, although they closely align with [details on third party interactions with Alphabet] (see Section 8.3.6.4).

(577) Importantly, the draft measures did not provide for full access to the DSP of the device. Loading a third-party sound model is a significantly less risky approach, [details of interactions between a third party and Alphabet]. Experts also confirmed that the integrity risks can be minimised when the loading third-party sound models to a DSP with an existing DSP library or hotword engine.[715] Similarly, in the expert report submitted by Alphabet, Prof. [professor] did not consider that measures regarding the AOHD feature were particularly sensitive and would require additional measures for user-downloaded apps.[716]

(578) Alphabet also misrepresented the implications on ambient microphone data collection. The draft and final measures seek to enable interoperability with the existing AOHD feature on Google Android. This feature does not allow an app to access any audio without showing a privacy indicator. To receive access to the audio, the second stage hotword validation needs to validate the hotword. As soon as the app receives full access to audio data, Android shows the privacy indicator. This is already explained in Section 8.3.3.

---

party's] submission contained the following sentence: "*Uncontrolled or insufficiently constrained DSP access risks system crashes, degraded device performance, intermittent audio failures, excessive battery drain, or sustained wake-lock scenarios, all of which can undermine both OEM and chipset supplier reputations.*", Doc ID […].

711 Google and [third party] are both members of [organisation].

712 Alphabet shared the [third party's] draft submission, received on 11 May 2026. See Alphabet's response to RFI PO 20, Annex Q1C.57, Doc ID […]. On 13 May 2026, two days after Alphabet received the draft from [third party], [representative of third party] wrote to […] at Google/Alphabet [direct quote from communications with third parties], see Alphabet's reply to RFI PO 20, Annex Q1C.10, Doc ID […].

713 [Third party's] submission to the public consultation, Doc ID […], section (i), (ii), (iii). [Third party's] submission has been quoted by Alphabet to reinforce Alphabet's arguments on the current interoperability solution. See Alphabet's response to the PFs, Doc ID […], paragraphs 202, 206.

714 On 7 May 2026, […], Google's public council, asked, among others, the following requests and questions to [third party]: [description of Alphabet's counsel's communications with third party regarding content of submission], Alphabet's response to RFI PO 20, Annex Q1C.59, Doc ID […].

715 See Agreed minutes of meeting with [academics] of 29 May 2026, Doc ID […], Section on Approach B and Approach C.

716 The expert report only refers to ambient data, proactive suggestions, Computer Control and App Functions as specific permission groups that are "are useful and risky" and it does not mention other measures. Annex 1 of Alphabet's response to the PFs, Doc ID […], Section I.B,

(579) *Third*, Alphabet argued that the measures for AOHD must be scoped to a restricted set of beneficiaries.[717] As further explained in Section 12.4.5, the Commission considers that placing a restriction on the beneficiaries that can access the AOHD feature is premature at this stage, since any integrity implications will depend on the actual implementation that Alphabet will choose for the interoperability solution (see Section 8.3.7.3). This Decision explicitly allows Alphabet to submit a reasoned request to add AOHD to the list of restricted features, should Alphabet demonstrate such restriction is strictly necessary and proportionate (see Section 12.4.5).

(580) In a later iteration of the measures, Alphabet requested to remove the explicit listing of the functionality to allow for matching the user's voice in the second stage of hotword detection. The Commission agreed to remove this, as it is not relevant where and when an app using hotword detection can match the user's voice. However, an effective interoperability solution requires that a third-party app will also be able to match the audio against the user's voice.

(581) *Fourth,* Alphabet argued that the Commission presented misleading facts in the PFs, basing the assessment of the integrity measures in place on wrong statements.[718]

(582) Alphabet argued that it is misleading to compare the available memory space on a DSP with the sound model size required, as other code and services would also run on the DSP.[719]

(583) The Commission notes that Alphabet did not substantiate their argument using concrete numbers or examples. It is correct that a DSP may run other functionalities, such as noise cancellation of microphone audio. However, Alphabet did not detail how much memory these features require and how much memory would remain available. In addition, third parties have worked on the same premise: [third party's proposal detailed in Section 8.3.6.4]. Samsung's smartphones can run up to three hotwords in parallel today [details on Samsung's solution].[720] Therefore, the Commission does not see any indication that, depending on the device and the DSP, there should not be enough memory space available to run additional sound models to support concurrent AOHD.

(584) In addition, Alphabet argued that the Commission wrongly compared nanoapps running in the Android Context Hub Runtime Environment (CHRE) to AOHD on the DSP. Alphabet does not contest that a Pixel phone runs [number of nanoapps] nanoapps in the CHRE, but considers that comparing it to AOHD would be incorrect, as they rely on different types of memory hardware.[721]

(585) While the DSP's hardware should result in better performance for hotword detection, Alphabet already uses the CHRE environment to execute sound models today for cough and snore detection, which demonstrates that the CHRE can support similar use cases.[722] In addition, Picovoice demonstrated that their hotword detection can run

---

717 Alphabet's first draft measures redline of 20 May 2026, Doc ID […].

718 Alphabet's response to the PFs, Doc ID […], paragraph 203.

719 Alphabet's response to the PFs, Doc ID […], paragraph 203, first bullet.

720 [OEM's] submission to the public consultation, Doc ID […], Section 1.2. [Details on solution].

721 Alphabet's response to the PFs, Doc ID […], paragraph 203, last bullet. Concretely, [describes Alphabet's product design choices regarding nanoapps on CHRE] (and therefore usually smaller) form of volatile memory.

722 Alphabet runs the sound models for cough and snore detection entirely in the CHRE. The DSP, which is optimised for always-on audio detection is not used for this feature. See Alphabet's internal document, [Alphabet internal document], […], Doc ID […], slide 8, slide 8 notes.

on a wide variety of different hardware from DSPs to more generic hardware.[723] Therefore, the Commission has not misrepresented the facts, but only compared two comparable technologies on Android that allow for always-on functionalities, including sound detection and hotword detection.

(586) Alphabet argued that Amazon's Voice Interoperability Initiative, allowing concurrent hotword detection on smart speakers, should not be compared to smartphones, as smartphone usually run on batteries, while smart speakers are plugged in.[724]

(587) As examples in Section 8.3.6 showed, several OEMs have implemented concurrent hotword detection on Android smartphones. The initiative is just another demonstration that there is industry and market demand for such implementations and that Alphabet has also participated in them.

(588) Therefore, the Commission has correctly analysed the facts at hand, and the Commission also based this decision on additional information that supports it (see Section 8.3).

(589) *Fifth*, during the public consultation some OEMs also suggested that the current interoperability model is the necessary one. [OEM's] submission stated that *"the DSP and its memory space belong to the area controlled by the manufacturer, not the OS provider"* therefore suggesting that all the cost linked to the interoperability solution would be on the manufacturer, i.e. OEM, not Alphabet.[725] [OEM] submitted to the public consultation that the AOHD feature is based on proprietary hardware, which would make a unified standard infeasible.

(590) It is understandable that OEMs are concerned that certain measures on Alphabet may lead to increased costs on their side. This would be especially detrimental for smaller OEMs. The Commission first notes that the burden to implement the interoperability solution is on the gatekeeper. In fact, Alphabet may implement a solution that does not change a single step in the OEMs' integration of Google Android on their devices. The measures for an interoperability solution would not require more memory on a DSP. Therefore, it would not increase the cost for OEMs. In addition, Alphabet has proven that a unified standard is feasible to implement for AOHD, its current DSP library runs across different DSP architectures and supports to run the same sound model on different DSPs.[726]

(591) The Commission observes that memory space and chipsets are routinely made available to third parties by OEM.[727] The DSP is not significantly different from other chipsets that are available and interoperable on Android devices today, as illustrated by the fact that Qualcomm, a major Android chipset vendor, integrates the DSP alongside other components (CPU, GPU, NPU) on its mobile device System on Chip (SoC).[728]

---

723 The Picovoice hotword Engine Porcupine supports to run on a Raspberry Pi, which uses DRAM, see https://github.com/Picovoice/wake-word-benchmark, Doc ID […].

724 Alphabet's response to the PFs, Doc ID […], paragraph 203, second bullet.

725 [OEM's] submission to the public consultation, Doc ID […], Section 1.2.

726 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID […], para 141-143.

727 For example, today a third-party app can run a large language model using more than 2GBs of system memory (RAM) and the NPU of the Samsung device.

728 Qualcomm's SoC includes a Sensing Hub. The Sensing Hub is an always-on processor that can also act as a DSP to process hotword detection. See here https://www.qualcomm.com/smartphones/products/8-series/snapdragon-8-gen-5-mobile-platform, Doc ID […].

(592) Therefore, the Commission considers that the current interoperability solution does not meet the standard of effective interoperability within the meaning of Article 6(7) DMA.

8.3.7.2. Generic hotword proposal

(593) [Alphabet proposed offering a generic hotword on Android that would be utilised for any third-party assistant].[729] [Alphabet proposed offering a generic hotword on Android that would be utilised for any third-party assistant].

(594) The generic hotword feature does not provide equally effective interoperability for the following reasons:

(a) A generic hotword is not the same Google Android feature as the AOHD used by Google.[730] The Google Android feature includes the functionality to load a custom sound model for a custom hotword on the DSP. The current AOHD feature that Alphabet uses allows Alphabet to fully customise the experience, including the user interface, the onboarding, and the hotword phrase.

(b) The generic hotword would not work concurrently with the Google hotwords. The generic hotword would only be enabled when the default assistant is set to an assistant that is not made by Alphabet. However, it has been shown that concurrent hotwords work on Google Android.[731] Concurrent hotwords are also required to allow third-party apps to compete with Alphabet's services.

(c) Generic hotword would by nature be available to only one alternative service and would therefore not deliver on the objective of multi-homing (Section 5.2.4).

(d) Third parties have told the Commission that a generic hotword would not be a preferred solution, as they could not build a brand around their assistant.[732] Assistants could not introduce a hotword phrase that would represent their brand. Today, most assistants use a name, such as Siri, Alexa, or Claude.

(595) As a result, the Commission considers that the generic hotword implementation would not amount for effective interoperability within the meaning of Article 6(7) DMA.

8.3.7.3. Interoperable implementations of AOHD

(596) Based on the information in the Commission's possession, the AOHD feature may be implemented in various ways. Besides the two implementations set out in the PFs, being third-party sound models and user-customisable hotwords, there are other

729 Alphabet submission of 7 April 2026, Hotword Solutions, Doc ID […], page 6. Solutions meeting Alphabet/Commission of 24 March 2026, Doc ID […], slide deck 14-19.

730 Cf. Section 4.2.

731 Samsung has three concurrent hotwords on their current Galaxy S26 devices. Amazon was able to have a concurrent "Alexa" hotword on selected Android phones between 2020 and 2023.

732 Agreed minutes of meeting with [AI service provider] of 11 March 2026, Doc ID […], paragraph 5-6. Agreed minutes of meeting with [third party] of 18 March 2026, Doc ID […]. As described by Amazon: "*The wake word is a strong brand indicator, and the choice of wake word is an important part of the experience.*" and "*The wake word, along with the agent's voice, is also a significant component of persona and brand.*", see Amazon (n.d.), 'Multi-Agent Design Guide - Discovery and Education', Amazon Alexa website, accessed 16 April 2026, https://developer.amazon.com/en-US/alexa/voice-interoperability/design-guide/discovery-and-education.html, Doc ID […].

viable implementations to enable effective interoperability for third-party developers.[733]

8.3.7.3.1. Third-party sound models

(597) Today, Alphabet creates custom sound models for AOHD on Google Android mobile devices. Alphabet currently deploys multiple TFLM-based sound models to Android 16 devices.[734] To run the sound models on the DSP, Google uses a DSP library, called Google Hotword Engine.[735]

(598) One option to implement a solution that provides effective interoperability would be to enable third parties to run their third-party custom sound models on the DSP using the same Google Android components that are used by Gemini: the AOSP APIs to load sound models onto the DSP, the middleware DSP library to execute sound models on the DSP, the Google Android APIs for the second stage validation, the Google Android APIs for the invocation.

(599) In its response to the PFs Alphabet *first* argued that the Google Hotword Engine is not part of the Google Android operating system.[736]

(600) The Commission explained in Section 6.2.2.2 that the Google Hotword Engine is part of the designated Google Android middleware and therefore an integral part of Google Android as it contributes to controlling basic functions of the hardware and software. This is because, amongst others, the Android APIs for AOHD are hardwired to the Google Hotword Engine. A third party app, even when preinstalled can implement AOHD using the available Android APIs, only by using the Google Hotword Engine (see Section 6.2.2.2 and Section 8.3.6).

(601) *Second,* Alphabet submitted that if the Google Hotword Engine were considered an "OS-mediated feature", Alphabet would only be required to allow access to the "OS mediation layer". The OS mediation layer would be the API that allows to communicate with the DSP firmware/library, but not the DSP library itself. Under these conditions, third parties would not be able to replicate Google's "implementation of hotwords".[737]

(602) Alphabet first misrepresents the Google Hotword Engine as an "OS-mediated feature". The Google Hotword Engine is a software component that contains part of the implementation of the AOHD feature into the designated Google Android (see Section 6.2.2.2). Alphabet then misunderstands how, even if Google Hotword Engine were to be an "OS-mediated feature" (*quod non*), interoperability could be implemented. The AOHD feature, as described in Section 8.3.2, includes the functionality to load sound models on the DSP and execute them on the DSP for hotword detection. However, without a DSP library, a sound model can neither be loaded onto the DSP, nor can it be executed. Access to the AOHD feature therefore requires access to a DSP library and it would not be sufficient for Alphabet to only offer access to the OS mediation layer. Otherwise, this would amount to granting access to an API that in practice does not work (by lack of the underlying implementation), which would not actually grant effective access to the feature.

(603) To provide effective interoperability, third-party developers must have access to a level that allows them to implement AOHD in their Android apps. To this end, they

733 Agreed minutes of the meeting with [AI service provider] of 3 November 2025, Doc ID […], para 10.
734 Alphabet's reply to RFI PO 15 of 27 March 2026, Doc ID […], question 5.
735 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID […], question 31, 32.
736 Alphabet's response to the PFs, Doc ID […], para 199
737 Alphabet's response to the PFs, Doc ID […], para 199

need to be able to implement their sound models in a way that they can run on the DSP chip using a DSP library. As the Google Hotword Engine is part of the Android middleware and provided with Google Android, it could be made available as a DSP library to be used by third parties to obtain effective interoperability.

(604) The Commission also notes that under the measures specified in this Decision, Alphabet is free to implement the interoperability solution for AOHD in a way that does not entail the use of the Google Hotword Engine. For example, Alphabet may implement a new DSP library that is built for interoperability and third-party access and distribute this DSP library as part of Google Android, similar to how it distributes the Google Hotword Engine today.

(605) *Third,* Alphabet submitted that the draft measures would require Alphabet to recreate and transfer its own technology to third parties without compensation. Alphabet owns [Alphabet internal data] patents worldwide (including more than [Alphabet internal data] in Europe) related to the technologies, architectures, and methodologies required to implement the AOHD features. The draft measures would require Google to replicate these technologies for third parties without compensation in breach of Alphabet's rights under Article 16 and 17(2) of the Charter.[738]

(606) Article 17 of the Charter and Article 1 of the First Protocol to the European Court of Human Rights grant everyone the right to own, use, and dispose of their lawfully acquired or created property, including their intellectual property, to the exclusion of third parties, subject to the limitations in the Article.

(607) It is upon Alphabet to bring to the Commission's attention whether the draft measures would give rise to interferences with specific intellectual property rights (IPRs) and to substantiate its claims. Alphabet is best placed to substantiate which patents protected in the Union it holds, which would be interfered with by the draft measures, and explain the scope, importance and effects of the alleged interferences with its patents. Alphabet has in-depth knowledge of its technology and patent portfolio.

(608) Such substantiated claims by Alphabet are necessary to allow the Commission:

(a) to evaluate whether those alleged interferences would amount to a restrictive regulation or deprivation of the patents invoked by Alphabet,

(b) to assess whether Alphabet could implement measures to mitigate or minimise any interference with its patents while still achieving the goals of Article 6(7) DMA and

(c) to make an informed assessment of the proportionality of the envisaged interoperability measures, taking into account the level of interference with specific patents substantiated by Alphabet, the need to achieve the DMA's legitimate public interest objectives and the legislative weighing of the public and private interests enshrined in Article 6(7).

(609) Alphabet did not meet the standard of substantiation. It did not substantiate which patents read on which technologies and why these technologies are required to implement the AOHD feature. Nor did Alphabet explain the scope, importance and effects of the alleged interferences with its patents.

(610) The Commission further notes that the measures do not prescribe a particular implementation of the AOHD feature. The technologies used may depend on the

738 Alphabet's response to the PFs, Doc ID […], paragraph 200.

implementation which Alphabet chooses. This may impact which, if any, patents are interfered in the sense that the patent reads on the particular technology.

(611) Alphabet has also not substantiated why any interference with its patents by this Decision would be disproportionate. In particular, Alphabet has not explained why the interference with its IPR by this Decision is of such nature and weight to cast aside, in this particular case, the Union legislator's weighing of the public interests against the wording of Article 6(7) DMA, which does not provide for a justification based on IPR.

(612) The Commission considers the following relevant in this respect. Alphabet decided to add the AOHD feature to the Google Android operating system and to allow third-party OEMs to make it in principle accessible to any third party (subject to limited exceptions). Alphabet benefits from the inclusion of the AOHD feature (and any IPRs reading on the underlying inventions) in the Google Android operating system through, inter alia, substantial licensing fees (see Section 5.1.3) and the sale of its own Google Android mobile devices.

(613) Lastly, Alphabet claims, without substantiation, that the measures would deprive it of patents and IPRs. Provided that Alphabet's patents and IPRs are interfered with (which Alphabet has not substantiated), the obligation to allow effective interoperability regulates the use of property. Alphabet is not deprived of its patents and IPRs nor of the possibility to use them. Alphabet can continue to use the patented inventions for any product, service and purpose. In addition, Alphabet will continue to financially benefit from the inclusion of these inventions in Google Android (see prior recital).

8.3.7.3.1.1. Concerning integrity risks related to an interoperable solution

(614) *First,* Alphabet submitted in its response to the PFs that access to the AOHD feature as envisaged in the PFs would raise unsolved integrity risks. Concretely, the interoperability solution would not solve compatibility issues between third-party sound models and the DSP that could result in integrity risks.[739]

(615) The Commission recalls that under Article 6(7) DMA, Alphabet is allowed to implement strictly necessary and proportionate integrity measures.

(616) The Commission considers that such measures could include for instance a compatibility definition that third-party sound models must adhere to. This definition could be technically validated every time before a sound model is loaded, guaranteeing that the sound model is compatible with the DSP. The compatibility definition can be defined in a way that each sound model essentially resembles the topology of the neural network that the current Google hotword sound model uses (see Section 8.3.7.3.1.2). The Commission's understanding is that this solution should not lead to integrity issues.[740]

(617) *Second,* Alphabet argued that an independent validator would not mitigate OS integrity risks of over-the-air sound model deployment. Alphabet argues that a sound model would run on many DSPs for the first time, without being tested on each separate chip in advance, which could lead to a higher-than-tested false activation rate.[741] Alphabet also submitted that an independent validation could not test all environments for sound models, including "*thousands of distinct Android device*

739 Alphabet's response to the PFs, Doc ID […], para 204

740 See Agreed minutes of meeting with [academics] of 29 May 2026, Doc ID […], Section on Approach C.

741 Alphabet's response to the PFs, Doc ID […], para 205

*models with different SoC and DSP architectures*"[742] and that an independent validation could not mitigate against the security risk of a malicious third-party sound model that would intentionally listen for another hotword phrase.[743]

(618) Concerning false activation rates, these should not differ between different DSPs. Alphabet's argument is purely hypothetical they did not substantiate why this could or would happen. By contrast, Alphabet confirmed that one sound model can run across all supported DSP architectures that support TFLM, not requiring different models for different DSPs. Alphabet also confirmed that a newly released DSP does not require Alphabet to bring any changes to its sound model.[744] Therefore, Alphabet's models are not tied to a specific DSP. Another example is the company [third party], which offers a hotword SDK. Their sound models perform equally well on all supported platforms, including DSPs.[745] It follows that Alphabet's contention that the risk of false activation rates would increase materially as a result of the measure is not substantiated.

(619) Concerning the capability of an independent validator to mitigate alleged integrity risked, due to too many configurations and alleged security risks, due to malicious models, the Commission observes from the outset, that Alphabet currently supports [10-20] DSP architectures for AOHD[746] not "*thousands*".[747] A validator could use a so-called development board for each of the DSP architectures in questions to assess the third-party models in an automated manner.[748] Testing on each individual device would not be not necessary and is not done by Alphabet today.[749] However, an implementation that closely resembles the one described in Section 8.3.7.3.1.2 would not require such an intensive testing because it relies on an already tested compatibility definition and could be verified technically. In this case, testing could be limited to detecting malicious or misbehaving models, without requiring testing for compatibility.

(620) Additionally, Alphabet may implement the following measures to detect malicious or misbehaving sound models:

(a) Before a sound model is loaded onto a DSP, an independent validation based on clear and fair criteria could detect potential issues upfront and only validate models that pass the required tests. Alphabet may, in coordination with the Commission and external experts, develop the required testing criteria and work on establishing an independent validation. The testing of sound models could be performed through independent testing labs and Alphabet.

(b) The same validation could also test for a series of terms that should not result in a hotword detection to ensure that the model does not detect common words. It could also test against existing hotword phrases to ensure that no model resembles another hotword phrase.

---

742 Alphabet's response to the PFs, Doc ID […], para 205

743 Alphabet's response to the PFs, Doc ID […], para 205

744 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID […], questions 29.30

745 Agreed minutes of meeting with [third party] of 27 May 2026, Doc ID […], para 4, 9.

746 Today Alphabet supports [DSP architectures]. Alphabet's reply to AI RFI 4 of 28 November 2025, Annex Q9, Doc ID […].

747 Alphabet's response to the PFs, Doc ID […], para 205

748 Development boards for microcontrollers with a DSP are available, see for example: https://estore.st.com/en/nucleo-f411re-cpn.html, Doc ID […].

749 Testing the sound model integration is optional for OEMs:"*[…] this step is optional*.". See Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID […], questions 32, paragraph 154.

(c) To protect sound models against manipulation, each sound model could be verifiable through cryptographical means, such as a signature, by the validating instance. Google Android should then only enrol and load sound models that have been validated and have been signed.

(d) If a sound model has been reported to misbehave, e.g. frequent false positive detections, its signature could be revoked. The third-party developer would then have to update their model and go through the validation process again before it can be used.

(e) Additional measures in Google Android could identify sound models that falsely activate more often than usual and warn the user. The user could then decide if they would want to keep the hotword active or to deactivate it.

(621) For example, independent validators already perform similar testing for the profiles offered by the App Defence Alliance.[750] A company that would want to obtain a certification for the security of their mobile application, would have to be tested by an independent validator. A similar program could be used for testing sound models.

(622) *Fourth,* Alphabet argued that concurrent sound models could cause interference between them when third parties train their own hotword sound models. In addition, some OEMs have demanded that concurrent hotword detection should not be mandated.[751]

(623) On the other hand, as regards the risk of interference when running multiple sound models concurrently, the Commission observes it is for third-party developers to implement a well-working sound model that does not interfere with other sound models. As for any software that is provided, developers have to ensure to their software is free of bugs, behaves as intended, and does not interfere with other apps. Otherwise, users might disable the hotword detection for this app or stop using their service altogether.

(624) In addition, OS-level detection mechanisms could be implemented to discover if concurrent hotwords would interfere with each other. Such mechanisms could alert the user if they would register that multiple hotword phrases activate at the same time.

(625) At the same time, the validation process is apt to check for interference by testing against every existing hotword. Interfering models could either be rejected, or they could be assigned with a warning, so that users are informed that some models could cause interference with others. The users would then still be free to configure the app for hotword detection. In any case, it is not up to the Commission to prescribe a model for testing against interference. This pertains to the specific implementation, which may or may not, depending on circumstances, require testing in the first place. For example, a user-customisable hotword solution (see Section 8.3.7.3.2) would require far less testing.

(626) On the other hand, contrary to certain OEMs stated concerns, the measures specified in this Decision do not mandate concurrent sound models if the DSPs do not have the available memory and compute resources available for implementation.

---

750 App Defense Alliance, Certification Scheme Overview, version 1.2, 26 March 2026, https://drive.google.com/file/d/1pOr_HcAIb9PqeYaYO05PEbYqagKxM2dR/view, accessed on 30 June 2026, Doc ID […].

751 Alphabet's response to the PFs, Doc ID […], para 207.

(627) It follows from the above, that any potential integrity risks as raised by Alphabet and incompatibilities associated with this implementation can be solved through technical or contractual means.

8.3.7.3.1.2. An implementation that could resolve most integrity risks raised by Alphabet

(628) In response to the PFs, Alphabet requested to be granted more flexibility as to the implementation of the AOHD feature and started an internal research project to develop a viable implementation plan.[752]

(629) Nevertheless, the Commission outlines below a possible solution that should resolve many raised integrity risks by design and offer a fallback solution to Alphabet if the research project turns out to be unsuccessful.

(630) This implementation would rely on the existing implementation that Alphabet is using today for AOHD in Gemini. The implementation is based on the idea that the existing sound models could be adapted for different hotword phrases. One would have to retrain the model using updated training data that represents the new phrase, i.e. recordings of people uttering the new hotword phrase. The topology of the model, i.e. its neural network architecture, would stay the same. The difference between the models are the weights. The weights define what will be detected, i.e. specific patterns, and features of a hotword phrase, while the neural network topology defines how it will be detected.

(631) The following high-level steps would be necessary to implement this solution on Google Android:

(a) Enable the AlwaysOnHotwordDetector and related SoundTrigger APIs to be accessible publicly. The APIs should enable all functionalities described in Section 8.3.2. Currently these APIs are system APIs and require a privileged permission;

(b) Implement a new Android permission that an app can request through a permission prompt to gain access to the AOHD implementation;

(c) Publish the topology of the sound model used to detect the Google hotwords.[753] Document on how a model with the same topology can be trained. Define which input can be expected and which output should be given;

(d) Allow third-party apps to use process isolation on Google Android for the second stage hotword validation, to ensure that a third-party app cannot access audio before the OS privacy indicator is shown;

(e) Allow apps that are not the default assistant, i.e. the active Voice Interaction Service, to use the Always On Hotword Detector API;[754] and

(f) Implement a sound model validation that validates if a sound model adheres to the topology before it can be enrolled in Google Android.

---

752 Alphabet's submission, "revised mark-up of the draft measures", Doc ID […], Section 1.2.

753 Already the existing sound models do not hide the internal architecture. Interested developers could download Google's Hotword Enrollment APK from their Google Android device using developer tools like adb. The APK contains the current sound models in the Tensor Flow Lite for Microarchitectures format. It is trivial to inspect the neural network architecture of the model using a model visualiser (e.g. https://netron.app/).

754 Currently only the active Voice Interaction Service can start hotword recognition. See the comment in the AOSP source code: "Caller must be the active voice interaction service via Settings.Secure.VOICE_INTERACTION_SERVICE", see https://android.googlesource.com/platform/frameworks/base/+/94b4c163b7dfe5ce3607f7bb8456f9573f7de57d/core/java/android/service/voice/AlwaysOnHotwordDetector.java#1148, Doc ID […], line 1148.

(632) The changes above would not enable concurrent hotword detection yet. AOSP does not allow to run multiple Always On Hotword Detectors in parallel. Therefore, the following additional measures would have to be made to implement concurrent hotword detection:

(a) Allow multiple Always On Hotword Detectors to run concurrently. The maximum number of concurrent detectors should be limited by the number of supported sound models on the DSP;[755] and

(b) Potentially, the DSP library must be updated to support to run multiple sound models.[756]

(633) [Describes Alphabet's internal product development processes and systems].[757]

(634) Alphabet submitted that changing the topology could corrupt the DSP, while keeping the same topology should not cause issues on the DSP itself.[758] This has also been confirmed by two experts. A model with updated weights could not cause any integrity issues, as long as it is validated technically prior to loading it on the DSP. Such a model would not require any external validation.[759]

(635) Perplexity has demonstrated that third-party developers are capable of creating a sound model that can detect a custom hotword phrase ("Hey Plex", later changed to "Hey Perplexity"[760]) on a Google Android mobile device. [further comments on the implementation of the Perplexity hotword on a Google Android device].[761] [further comments on the implementation of the Perplexity hotword on a Google Android device].

(636) The Perplexity AOHD on the Samsung Galaxy S26 devices also runs concurrently with Google hotwords and Samsung's Bixby hotwords. This example demonstrates that modern Google Android smartphones can use at least three hotword sound models concurrently. Furthermore, the implementation on Samsung phones also runs independently of the assignment of the default AI assistant on Google Android. The user may choose Google Gemini as their default assistant and still invoke Perplexity's assistant and Bixby with AOHD.[762]

(637) The Commission considers that the other risks that Alphabet identified (see Section 8.3.7.3.1.1) can be resolved with the appropriate mitigating measures in place.

(638) To conclude, the Commission is confident that Google Android can be changed to support third-party AOHD for user-downloaded apps using the proposed implementation. The proposed solution itself already mitigates many integrity risks

---

755 Currently the AOSP source code of Voice Interaction Service only allows one Always On Hotword Detector "There is already an active AlwaysOnHotwordDetector. It must be destroyed to create a new one.", see core/java/android/service/voice/VoiceInteractionService.java - platform/frameworks/base - Git at Google, Doc ID […], line 744.

756 According to Alphabet's responses the code that handles audio distribution must be updated: "While Google's library can, in theory, support multiple models running on the DSP concurrently, the DSP chipset's firmware, including the custom code, must be updated to distribute audio across all models", Alphabet's reply to RFI PO of 13 February 2026, para 73.

757 Alphabet's reply to RFI PO 19 of 17 April 2026, Doc ID […], question 9.

758 Alphabet's reply to RFI PO 19 of 17 April 2026, Doc ID […], question 9.

759 See Agreed minutes of meeting with [academics] of 29 May 2026, Doc ID […], para 20.

760 See Schenck, S. and AssembleDebug (2026), 'Perplexity is changing its wake word from "Hey Plex," and here's the proof', Android Authority website, 7 April, https://www.androidauthority.com/hey-perplexity-3655439/, Doc ID […], accessed 13 April 2026.

761 Agreed minutes of meeting with [AI service provider] on [date], Doc ID […], paragraphs 1-15.

762 See Samsung (2026), 'Enable Perplexity on your Samsung Galaxy phone', Samsung US website, https://www.samsung.com/us/support/answer/ANS10010357/, Doc ID […], accessed on 13 April 2026.

raised by Alphabet as a model with updated weights cannot cause integrity issues on the DSP. The solution does not give direct access to the DSP, protecting the environment from untrusted code. An independent validator can mitigate additional risks that could occur from malicious parties that would try to mimic hotword phrases of other parties.

8.3.7.3.2. User-customisable hotwords

(639) Alternatively, there could be a solution where a user freely chooses a hotword for an app. The required sound model for the according hotword would be trained based on the user's input. For example, the user could specify a phrase, e.g. "Hey Aquarius" and then speak this phrase several times into their microphone. The solution would then create a new custom sound model that can detect this phrase if uttered by the user. The entire implementation of the model creation would be under Alphabet's control, reducing any potential integrity risks to the same level that Alphabet has today. Third-party developers would only be required to implement functionalities for the user to input the hotword phrase and the recordings.

(640) Based on internal documents, Alphabet has pursued to implement such user-customisable hotwords to allow users of the Gemini app to decide for a custom hotword phrase. However, it seems that to date, the custom hotword project has only been worked on by a small number of engineers. The internal documents describe that [direct quote describing Alphabet's commercial and strategic discussions].[763] Still, a substantial amount of work would have to be done, which would require more staff and engineers on this project. [Description of Alphabet's commercial and strategic discussions].[764]

(641) [Description of Alphabet's commercial and strategic discussions].[765]

(642) Public research has also assessed the feasibility of user-customisable hotwords. In 2022, Indian researchers developed a new hotword training method based on few-shot learning.[766] This technology allowed them to create customisable hotword models that can run on a low-power device. Their architecture achieves an accuracy of over 94%. The research has been made available as an open-source project.[767]

(643) In the response to the PFs, at first, Alphabet argued that the draft measures on user-customisable hotwords set out in the PFsconstitute a new feature that Alphabet has not implemented. As a result, Alphabet argued that the implementation of user-customisable hotwords would go beyond implementing interoperability for the AOHD feature and, therefore, exceed the scope of Article 6(7) DMA.[768]

(644) The Commission notes that the operating system feature subject to these proceedings is AOHD, and that user-customizable hotwords is merely one implementation of that feature. The purpose of the user-customisable hotword measure in the PFs was to make the AOHD feature available to third parties.

---

763 Alphabet internal document, [Alphabet internal document], Doc ID […], page 16.

764 Alphabet internal document, [Alphabet internal document], Doc ID […], page 19.

765 Alphabet internal document, [Alphabet internal document], Doc ID […], page 68, 82, 94.

766 See R, C., Rangapur, A. and Sethuraman, S. C. (2022), 'EfficientWord-Net: An Open Source Hotword Detection Engine based on One-shot Learning', *Journal of Information & Knowledge Management*, Vol. 21, Issue 04, p. 2250059, https://doi.org/10.1142/S0219649222500599, accessed on 20 April 2026, Doc ID […].

767 See 'Ant-Brain/EfficientWord-Net' (2026), ANT-BRaiN, https://github.com/Ant-Brain/EfficientWord-Net, accessed on 20 April 2026, Doc ID […].

768 Alphabet's response to the PFs, Doc ID […], para 196, 197.

(645) As explained above, user-customisable hotwords would not increase any integrity risks as the entire implementation would remain fully in control with Alphabet. It would not require any additional integrity measures, such as an independent validator. Therefore, user-customisable hotwords could be one way to bridge the interoperability gap in a way that already mitigates any integrity risks by design. In addition, user-customisable hotwords could also result in a high adoption from third-party developers, as developers would only have a minor engineering effort to enable hotword functionality for their services.

(646) *Second,* Alphabet argued that a user-customised hotword solution would likely not work across the entire ecosystem and that they ran into several issues when working on user-customised hotwords. As a result, Alphabet does not know today if this solution could be implemented as an effective interoperability solution across the entire ecosystem.[769]

(647) Alphabet did not substantiate why this implementation should not be technically feasible, in contrast, internal documents show that the project seems to be technically feasible. Internal documents show that [description of and direct quotes from Alphabet internal documents and Alphabet strategic decisions] [770] [description of and direct quotes from Alphabet internal documents and Alphabet strategic decisions] .[771] [description of and direct quotes from Alphabet internal documents and Alphabet strategic decisions] ,[772] but this does not preclude that improvements are possible. Other parties have successfully implemented user-customisable hotwords, suggesting that the engineering problems can be solved (see below).

(648) In addition, the Commission understands that a user-customised hotword solution may demand more hardware resources as the current solution. Therefore, it may not work equally well on all devices. Android devices have a broad price range and more expensive devices, usually come with better hardware, and potentially a better DSP chip.[773] Alphabet should strive to implement a solution that would work on as many devices as possible, but it may be that some lower tier devices or older devices will not be capable to implement the required changes due to hardware limitations, as is the case for any feature (e.g. the inability to run on-device LLMs reliably on some older Google Android mobile devices).

(649) The Commission has assessed the feature and showed the feasibility of several interoperability solutions. This notwithstanding, Alphabet retains the right to ask to waive certain measures should they be demonstrably impossible to implement (Section 12.11).

(650) *Third*, in the context of these proceedings, Alphabet has argued that [Description of Alphabet's commercial and strategic discussions]. Therefore, there is an issue of technical feasibility. Alphabet claimed that the engineers could not reach the

769 Alphabet's response to the PFs, Doc ID […], para 206

770 Alphabet's response to the PFs, Doc ID […], para 196-198

771 Alphabet internal document, [Alphabet internal document], Doc ID […], page 24.

772 Alphabet internal document, [Alphabet internal document], Doc ID […], page 19.

773 Android OEMs usually integrate one System on Chip (SoC) in their smartphones. The SoC encompasses multiple distinct components on one integrated circuit. These components include the processor (CPU), the graphics chip (GPU), and often also includes an always-on processors, like a DSP. Qualcomm's SoC includes a Sensing Hub, see here https://www.qualcomm.com/smartphones/products/8-series/snapdragon-8-gen-5-mobile-platform, Doc ID […]. The Sensing Hub is an always-on processor that can also act as a DSP to process hotword detection.

[required quality].[774] Alphabet argued that additional staff might not solve the issue to bring user-customisable hotwords to the entire Android ecosystem.

(651) Alphabet did not further substantiate, why the issues could not be solved within an ongoing implementation process.[775] Alphabet did not define the [required quality] that they have referred to. According to internal documents, [description of Alphabet's commercial and strategic discussions].[776] [Description of Alphabet's commercial and strategic discussions] as such incorrect detections are dealt with in the second stage validation.

(652) Internal documents reveal that [Alphabet internal documents].[777] In contrast, alternative hotword implementations do achieve a better quality, i.e. a lower false acceptance rate. During the public consultation, the hotword SDK Porcupine, offered by Picovoice, was mentioned.[778] Picovoice's sound models have a very low false acceptance rate of only one false activation within ten hours, 200 times lower compared to the Google sound models.[779] As Picovoice's sound models can be generated on demand, essentially allowing user-customisable hotwords, this example demonstrates that a high-quality bar can be achieved.[780]

(653) However, there is room for further improvements. Picovoice's sound models are lightweight hotword sound models that only require around [20-30]KB of memory, less than half the size of Alphabet's models.[781] Picovoice can train new models for new hotword phrases within minutes, allowing for user-customised sound models, and the hotwords engine can run an unlimited number of sound models concurrently.[782] The Picovoice SDK is compatible with the same architecture that is used by DSPs.[783] This technology demonstrates that concurrent, user-customisable hotwords with a low false acceptance rate can be implemented. These examples also prove that sound models can achieve even lower model sizes, allowing for running many concurrent hotwords.

(654) As described in Section 8.3.6.3, Samsung has implemented a user-customisable hotword for the Samsung Bixby assistant. Currently, Samsung's implementation [further information on Samsung's implementation].[784] Nevertheless, the implementation as-is is working and resulting in a reliable detection of the user-customised hotword. Samsung also implemented [further information on Samsung's

---

774 Solutions meeting Alphabet/Commission of 08 April 2026, [Alphabet employee] highlighted orally that they tried to implement it, but could not reach the quality bar. Solutions meeting Alphabet/Commission of 24 March 2026, Doc ID […], slide deck 14-19.

775 Alphabet's response to the PFs, Doc ID […], paragraph 197.

776 Document provided as part Alphabet's reply to RFI PO 18 of 10 April 2026, [Alphabet internal document], Doc ID […], pages 9-10.

777 Document provided as part Alphabet's reply to RFI PO 18 of 10 April 2026, [Alphabet internal document], Doc ID […], page 10.

778 See [third party's] contribution to the Commission's public consultation, Doc ID […], page 5.

779 See open-source benchmark of Picovoice's hotword engine Porcupine: https://github.com/Picovoice/wake-word-benchmark, Doc ID […].

780 On Picovoice's website, one can generate a new hotword model on demand within seconds, see https://picovoice.ai/products/voice/wake-word/, Doc ID […].

781 Alphabet's sound models use between [50-150] KB, see Section 8.3.3. Picovoice's sound models use under [20-30]KB per hotword / keyword to detect it. See https://picovoice.ai/products/voice/wake-word/, Doc ID […].

782 Picovoice allows for theoretically an unlimited amount of hotwords, public examples show up to four hotwords on a DSP-like microchip. See https://picovoice.ai/products/voice/wake-word/, Doc ID [..].

783 Picovoice's hotword SDK can run on ARM Cortex-M4, https://picovoice.ai/docs/quick-start/porcupine-microcontroller/, Doc ID […]. This is one of the supported DSP architectures.

784 Agreed minutes of meeting with [OEM] of [date], Doc ID [...], para 6.

implementation]. Samsung's implementation demonstrates the technical feasibility of a user-customisable hotword on a Google Android mobile device.

(655) A customisable hotword would also increase user privacy: different users are likely to pick different hotwords, and the training process inherently tunes the detection to the user's voice, making accidental activations less common.[785]

(656) Given the presented evidence, the Commission believes that the technology for concurrent user-customisable hotwords exists already today, and Alphabet would have the means to implement it as an effective interoperability solution for AOHD:

(a) the Picovoice SDK demonstrates highly accurate, low-power, low-memory and customisable models that can run concurrently on DSPs;

(b) Samsung offers a working solution for user-customisable hotwords for its own Bixby assistant; and

(c) [Describes Alphabet's internal product development processes regarding hotwords].

(657) Therefore, the Commission considers that there are various possible implementations of the AOHD feature that would amount to effective interoperability within the meaning of Article 6(7) DMA. In any event, pursuant to the measures attached to this Decision, Alphabet is free to decide on the exact implementation if it enables effective interoperability with the AOHD feature within the meaning of Article 6(7) DMA.

8.3.7.4. Concerning proportionality

(658) Alphabet argued in its response to the PFs that the measures for always-on hotword detection are disproportionate. However, Alphabet does not substantiate why the measures would be more restrictive than necessary to achieve the objectives of Article 6(7) DMA. In particular, Alphabet fails to propose any less restrictive, yet equally effective means of complying with that provision. Nor does Alphabet explain, let alone substantiate, why the feature and/or a specific measure would impose a disproportionate burden on Alphabet. Instead, its arguments focus on questioning the scope of Article 6(7) and the actual need for interoperability – a line of reasoning which does not meet the legal standard for a valid proportionality claim, as explained in Section 13 below.

(659) *First*, Alphabet argues that the hotword is no longer the primary access point for Gemini (the app icon is now that primary access point) and therefore implementing a complicated interoperability solution for this feature would be disproportionate within the meaning of Article 8(7) DMA.[786]

(660) The Commission notes that interoperability obligation of Article 6(7) DMA does not have a qualifier for usage or importance of a feature for the gatekeeper. Even a feature that is not important to the gatekeeper, it can lead to major innovations in third-party apps.

(661) In addition, Alphabet's statement that hotword would be a declining invocation method on mobile is incorrect.[787] The Commission's statement in the PFs correctly

---

785 See Martin Biéri, Félicien Vallet, Isabelle Corbara, Juliette Hirtz, Pauline Faget et al., 'ON THE RECORD - Exploring the ethical, technical and legal issues of voice assistants', Commission nationale de l'informatique et des libertés, November 2020, accessed 20 March 2026, https://www.cnil.fr/sites/cnil/files/atoms/files/cnil_white-paper-on_the_record.pdf, Doc ID […].

786 Alphabet's response to the PFs, Doc ID […], para 209.

787 Alphabet's response to the PFs, Doc ID […], para 209.

reflects the invocation numbers provided by Alphabet. Between 2025 Q4 and 2026 Q1, the hotword invocations increased by [40-50]% to […] invocations, even if the invocations via the app icon grew even stronger (see Section 8.3.5).[788] However, this only illustrates the overall growth of the AI assistants' interactions but by no means takes away the importance of hotword invocations on mobile, which in the future might even grow as agentic experiences become mainstream on mobile.

(662) Alphabet's statements also lack the context. The app icon invocation and the hotword invocation serve different use cases of Gemini. While the predecessor of Gemini, Google Assistant did not have an app interface and was purely acting as an assistant, invoked via hotword or long-press on the power button, Gemini is built around a chatbot functionality. Therefore, the use cases for Gemini, compared to Google Assistant, dramatically expanded. Indeed, the app icon invocation and the other direct invocation mechanisms (including hotword and long-press power) serve different use cases of Gemini.[789] In addition, there is a high demand for this type of invocation amongst developers of AI assistants and the hotword invocation was labelled as a crucial access point for AI assistants.[790]

(663) *Second,* Alphabet argued that the draft measures could have harmful consequences for end users, as their device could have shorter battery life, that the risk of system bugs increases and that mid-range phones could not handle some parts of the implementation, such as concurrent hotwords. Therefore, Alphabet argues that an interoperable solution would not be proportionate.

(664) The Commission notes that the final implementation of the interoperability solution depends on Alphabet. Therefore, it is also to a large degree in Alphabet's hands to create a solution that will not impact the battery life of Android users or introduces system bugs. The Commission believes that Alphabet can manage to implement an interoperability solution that is on par with the existing solution in Google Android.[791] In fact, the interoperability solution could be implemented nearly identically to the existing solution in a way that should have minimal additional impact on the user's battery life (see Section 8.3.7.3.1).

(665) Alphabet did not substantiate in their response why a third party should not be equally capable to create a sound model or implementation that is battery efficient. In fact, the false activation for Alphabet's sound model is […] times per hour, which should already have significant impact on the battery.[792] By contrast, third parties have managed to create hotword detection that only has one false activation per ten

---

788 Meeting Alphabet/Commission of 24 March 2026, Doc ID […], slide deck page 19. Alphabet's reply to RFI PO 15 of 27 March 2026, Annex Q3, Doc ID […].

789 Hotword invocation launches the app in a hands-free mode (also called Floaty) that allows the user to send a request using voice. Opening the app through the icon launches the full chatbot experience. [Direct quote from Alphabet internal document]. See Alphabet's internal document, [Alphabet internal document], Doc ID […], page 2.

790 Agreed minutes of the meeting with [AI service provider] on 28 October 2025, Doc ID […], para 1-10. Agreed minutes of the meeting with [AI service provider] of 3 November 2025, Doc ID […], para 3-12. Agreed minutes of the meeting with [AI service provider] of 11 March 2026, Doc ID […], para 1. [Third party's] submission to the Public Consultation, Doc ID […], para 20. [Third party's] submission to the Public Consultation, Doc ID […], page 2. [AI service provider's] submission to the Public Consultation, Doc ID […], Section 2.

791 [Third party], a company that creates SDKs for voice-driven interactions explained that even running concurrent sound models does not lead to significant increases power usage. [Third party] explained that even running concurrent sound models does not lead to significant increases power usage, see agreed minutes of the meeting with [third party] of 27 May 2026, Doc ID […], para 10.

792 Document provided as part Alphabet's reply to RFI PO 18 of 10 April 2026, [Alphabet internal document], Doc ID […], page 10.

hours (see Section 8.3.7.3.2).[793] Therefore, Alphabet is not justified in holding that third party access would lead to user harm and less battery life. As the example shows, the contrary could be the case, as a third-party assistant may result in longer battery life.

(666) Alphabet did not substantiate why the risk for system bugs should increase. Following the measures of the PFs and the measures specified this Decision, the risk for system bugs should not be larger than today. It depends on the final implemented interoperability solution, but the solutions presented in this section does not increase the risk for system bugs: third-party sound models that follow a compatibility definition cannot cause any bugs that could not be caused by Alphabet's sound models today. User-customised hotword detection would be fully under the control of Alphabet, and third parties would not have access in a way that could cause system bugs.

(667) The Commission therefore concludes that there is an interoperability gap with respect to always-on hotword detection feature of Google Android. As a result, third parties and in particular user-downloaded apps, cannot perform important functionalities which places them sub-par in comparison with Google's own AI-powered services. As a result, the Commission considers that it should specify how Alphabet must fill this gap to comply with Article 6(7) DMA.

*8.3.8. Measures that Alphabet should implement*

(668) To meet the requirements of Article 6(7) DMA, Alphabet should implement the measures set out in Section 1.2 of the Annex to this Decision. The interoperability solution for the AOHD feature, as described in Section 8.3.2 of this Decision, must be equally effective as that available to any of Alphabet's own services.

(669) Following discussions with Alphabet, the measures pertaining to AOHD feature attached to this Decision leave significant room for discretion to Alphabet as to the exact implementation of the features, as long as the outcome allows for equally effective interoperability. Alphabet is best placed to decide on an interoperability solution that will result in the least impact on available resources. The Commission has proposed two workable solutions in Section 8.3.7.3.

(670) To devise an interoperability solution, Alphabet started a research project.[794] If Alphabet's internal research project does not yield any results, the Commission recommends that Alphabet should implement the solution presented in Section 8.3.7.3.1, as it is fully based on the existing implementation of AOHD in Google Android.

(671) In addition, to have a true level playing field, it is important that several AI assistants can be invoked concurrently. This is because a user who would want to try a new assistant app to potentially replace Alphabet's Gemini as the default assistant in the future, needs to be able to compare the app with Gemini and its capabilities without switching back and forth the default assistant role, a multi-step process in the Android settings. Therefore, AOHD cannot be solely enabled for a default assistant, which in the out-of-the-box experience in most cases is set to Gemini.

(672) At the same time, concurrent AOHD will allow for true multi-homing, allowing users to have multiple, specialised assistants running in parallel. For example, the

---

793 See open-source benchmark of Picovoice's hotword engine Porcupine: https://github.com/Picovoice/wake-word-benchmark, Doc ID […].

794 Alphabet's submission, "revised mark-up of the draft measures", Doc ID […], Section 1.2.

user could run a dedicated assistant to control their smart home through voice commands, while keeping Gemini active for other tasks.

(673) For the purpose of ensuring that effective interoperability continues in the future, Alphabet should also provide effective interoperability with any future updates, including new functionalities, of the AOHD feature insofar as they are available to Alphabet's own services or hardware. To this end, the general measures in Section 12 of this Decision apply, including in particular the measures concerning future updates and new functionalities set out in Section 12.9 in this Decision.

*8.3.9. Timing for implementation of the measures*

(674) To meet the requirements of Article 6(7) DMA, Alphabet should implement the measures in the Annex (Section 1.2) according to the timelines indicated therein.

(675) Alphabet requested an implementation timeline of two years,[795] as they are expecting that the interoperability solution would require extensive engineering work and that Alphabet is currently at the early stage of a research project to explore viable implementation options.[796] While the Commission allows for flexibility in the implementation, as hotword detection continues to be a major access point for Gemini and will be an important access point for third-party services, the Commission considers that an implementation timeline of one year better achieves the objective of effective interoperability without further undue delay to third parties' right to interoperability under Article 6(7). The Commission offers one possible solution that should only result in minor engineering efforts in Section 8.3.7.3.1.2.

(676) The Commission considers that the implementation of a solution that allows concurrent AOHD for multiple apps may require more time. Therefore, Alphabet should be granted additional implementation time to develop a solution that supports concurrency in line with Alphabet's request that is two years.

## 9. MEASURES ON FEATURES FOR CONTEXT

### 9.1. Introduction

(677) Context consists of the ability for AI services to engage with what the user is currently doing, by gathering inputs and data from what the user or the device are observing (e.g. screen or sensor data). This includes all the historical context (also called memory) that has been built up, either inside the service or centrally on-device, including when provided by other apps. Context and memory are what make an AI service knowledgeable and tailored, to let it understand and anticipate the user's needs. Context complements and refines the query and answer: a service uses this context, either together with an explicit prompt from the user or on a proactive basis, to decide on the relevant actions to take or suggest.

(678) Alphabet sees context and memory as crucial to unlocking the next generation of AI services and assistants, including Gemini. [Direct quote describing Alphabet's commercial and strategic discussions on context and memory][797] With the upcoming Android Context Engine (ACE), Alphabet is looking to build a much deeper

---

795 Technical meeting Alphabet/Commission of 12 June 2026, slide deck Restricted Features and Implementation Timelines, Doc ID […], slide deck page 13. Alphabet's reply to RFI 21 of 8 June 2026, Doc ID […], question 6, paragraph 17.

796 Alphabet's submission "revised mark-ups on always-on hotword detection", Doc ID [...], para 8, comment.

797 Alphabet internal document, internal email of 24 June 2024, with subject [Alphabet internal document], Doc ID […], page 5.

integration with Android for contextual AI services. [Describes Alphabet's commercial and strategic discussions].[798]

(679) Equivalent access to context is crucial to allow third-party services to deliver as rich an experience as Alphabet's services, without being disadvantaged purely by a lack of access. Even if a third-party service offers the best models, user experience, etc., if Alphabet's service has exclusive or preferential access to certain elements of context, such as continuous capture of sensor data or of what the user sees on the screen, or has the exclusive or better ability to be a proactive companion to the user, a third-party service can never match the first-party service when it comes to knowing and supporting the user. This would make users more likely to return to the more knowledgeable gatekeeper incumbent service rather than stay with or explore a third-party alternative.

(680) A lack of equivalent access to context may reduce the incentive for third parties to create innovative solutions at an important inflection point for the development of AI services and assistants towards more personalised and agentic behaviour.[799] This could exclude otherwise novel solutions from being developed and ultimately reaching end users simply because third parties are unable to rely on access to context, which is necessary for innovation to happen in these areas.[800]

(681) Based on the assessment developed hereafter, the Commission considers that there is currently an interoperability gap when it comes to features for context for third-party services on Google Android mobile devices.

(682) The Commission therefore specifies how Alphabet must fill this gap, to comply with Article 6(7) DMA. The measures cover both access to data and the ability to provide contextual services. Centralised access to apps' data stored on-device (Section 9.2) and access to ambient data (Section 9.4) are particularly important for third parties when accessing the context-aware intelligence feature to provide innovative proactive suggestions and intelligent experiences (Section 9.3).

## 9.2. Measures for centralised access to apps' data stored on-device

### *9.2.1. Introduction*

(683) Data from the apps that users use on their devices is an essential input for AI services to understand the user's context and deliver rich and personalised services. The ability to access such data in a centralised manner via on-device databases or data sharing platforms provides AI services with efficient cross-app data access, search and retrieval capabilities. This allows those services to efficiently use that data, for instance, as valuable context to users' queries or as input for innovative use cases such as proactive suggestions, in an equally effective way as Alphabet's services.[801]

(684) Indeed, access in a centralised manner, as opposed to on an app-by-app and siloed basis, eliminates the need to engage in one-on-one relationships, integration or agreements with every app provider. In addition to not being scalable, such siloed solutions would in practice generally be easier to obtain by the gatekeeper (due to its

---

798 Alphabet internal document, Presentation [Alphabet internal document] of January 2026, Doc ID […], page 7.

799 See in this regard: https://www.oecd.org/content/dam/oecd/en/publications/reports/2026/02/the-agentic-ai-landscape-and-its-conceptual-foundations_a9d4b451/396cf758-en.pdf, Doc ID [...].

800 For instance, third parties need to be able to integrate with some lower-level features enabling equivalent access to user's context in order to offer alternative services. See for instance agreed minutes of meeting with [AI service provider] of 20 March 2026, Doc ID […].

801 Agreed minutes of meeting with [AI service provider] of 6 March 2026, Doc ID […], and agreed minutes of meeting with [AI service provider] of 11 March 2026, Doc ID […].

position and large installed base) than by third parties, who would therefore often be at a disadvantage, even if preferred by the end user. Access to app data which is exclusively stored on-device is of particular importance, as this data is only stored on the device and cannot therefore be accessed via any other means (e.g. via the cloud).

(685) The relevance of centralised access to apps' data stored on-device will continue to grow, especially in the context of future developments around personalised agentic AI, including more sophisticated proactive experiences and improved abilities for task delegation by users.[802]

(686) Alphabet has developed solutions permitting centralised access to apps' data stored on-device which enable these capabilities and support Alphabet's own services. However, the same level of access is not available to all third parties, in particular as access is limited to one app at a time, which the user must set as the default assistant, rather than allowing access by multiple apps in support of multi-homing (Section 5.2.4).

*9.2.2. Description of the feature*

(687) Users may use many different apps on their devices, including OEM apps, apps that come pre-installed on the device and apps that the users themselves decide to download and install. These apps may store their data on-cloud and/or on-device.

(688) Storing data on-device has several advantages compared to on-cloud storage, including enhanced privacy, lower latency, more efficient cross-app search and retrieval capabilities, and the ability to access the data without the need for an internet connection. App developers may therefore need or choose to store their apps' data on-device, and some app data may exclusively be stored on-device and hence not available via any other means: for example, some OEM apps' data or privacy-sensitive data may always be kept on-device.[803]

(689) There is an interest from OEMs, app developers and Alphabet itself in use cases where a "recipient app" can access the data that a "donor app" is storing on-device. For instance, this data can be used by services to offer users a relevant assistive experience.[804] There is also an interest from apps to donate (some of) their data for use by other apps.

(690) When donor apps donate their data for access by other apps, they can indicate which recipient apps are eligible to access that data. Access to that app data may be obtained on a siloed, one-on-one basis (i.e. the donor app can share its data with the recipient app by specifically allowlisting[805] it and/or integrating with it), but this requires a significant effort from developers to identify all apps they might want to share data with (currently or in the future) and/or integrate with each of them in a tailored manner. Given the friction associated with one-on-one negotiations, this also limits the ability, in particular of smaller developers, to negotiate integrations and may thus provide incentives to stick to the incumbents (Section 5.4).

---

802 Alphabet describes [direct quote describing Alphabet's commercial and strategic discussions]. See Alphabet internal document, Document [Alphabet internal document], Doc ID […], page 2.

803 See Alphabet internal document, [Alphabet internal document], Doc ID […], page 1: "[direct quote describing Alphabet's commercial and strategic discussions].

804 Agreed minutes of meeting with [AI service provider] of 6 March 2026, Doc ID […]: "*For instance, access to the messages app's data is important for offering users a relevant assistive experience*".

805 Allowlisting should be understood as the action of a party (e.g. an app) including another party (e.g. another app) in a list or category of parties, which enables those listed parties to carry a specific action (e.g. a donor app allowlisting a recipient app enables the latter to access the former's data).

(691) The centralised access to apps' data stored on-device feature addresses disadvantages of this siloed approach on Android devices. Contrary to one-to-one sharing, it enables donor apps to choose to share data, via an on-device database or data sharing platform, with a group of apps, usually managed via roles such as the SMS or default assistant role,[806] while recipient apps can then easily discover all donor apps and collectively access their data. This allows, first, donor apps to donate their data to any app in that group without having to identify, allowlist and/or integrate with every single app which users might choose for that role, currently or in the future; and second, the creation of a pool of data which can be accessed and used by any role holder on equal terms in a centralised manner.[807]

(692) Donor apps can also donate their data for access by the operating system, but this operates with a different paradigm.[808] This Section 9.2 primarily addresses the opt-in role-based centralised access, where apps consciously and explicitly choose to share data for broad use with other apps when those are chosen by the user as the holder of a default role (e.g. the default assistant role). The opt-out access, which is granted to the operating system for integration with operating system surfaces and which may be less apparent to donor apps as it is enabled by default, is addressed in Section 9.3.

*9.2.3. Technical implementation of centralised access to apps' data stored on-device in Google Android*

(693) The current implementation of the centralised access to apps' data stored on-device feature in Google Android is via AppSearch, an on-device database suited for efficient indexing and querying of data.[809] This includes AppSearch's ability to store vector embeddings and enable semantic search, which is particularly useful for AI services.[810]

(694) AppSearch currently offers three storage solutions:[811]

(a) Local Storage: the donor app manages its own specific index (akin to a repository). The data is only accessible by the donor app itself (i.e. data is not accessible by any system service or other app).

(b) Platform Storage: the donor app contributes to a system-wide central index, allowing it to share data with other apps, which can also search over its data. The data can also be displayed on system UI surfaces (e.g. the home screen), unless the donor app opts out.

(c) Play Services Storage: this mirrors the Platform Storage conditions, except that the central index is managed by Google Play Services, not in the system server;

---

806 See https://developer.android.com/reference/androidx/appsearch/app/SetSchemaRequest.Builder, Doc ID […], accessed on 10 April 2026, method "addRequiredPermissionsForSchemaTypeVisibility".

807 This data access is read-only: the recipient app cannot alter the donor apps' data. See https://developer.android.com/develop/ui/views/search/appsearch, Doc ID […], accessed on 20 April 2026.

808 See https://developer.android.com/reference/androidx/appsearch/app/SetSchemaRequest.Builder, Doc ID [...], accessed on 10 April 2026, method "setSchemaTypeDisplayedBySystem".

809 See https://developer.android.com/develop/ui/views/search/appsearch, Doc ID [...], accessed on 20 April 2026. See also Alphabet internal document, [Alphabet internal document], Doc ID […], pages 3, 8 to 10 and Alphabet internal document, [Alphabet internal document], Doc ID […], pages 4 to 6, .

810 See https://developer.android.com/reference/kotlin/android/app/appsearch/EmbeddingVector, Doc ID […], accessed on 27 June 2026.

811 See https://developer.android.com/develop/ui/views/search/appsearch, Doc ID […], accessed on 10 March 2026. See also Alphabet internal document, [Alphabet internal document], Doc ID […], page 36.

and that indexed data is not displayed on system UI surfaces. [Describes Alphabet's position regarding Play Services storage].[812]

(695) Of these three solutions, it is mainly Platform Storage that is relevant to centralised access to apps' data stored on-device. As noted above, within Platform Storage, donor apps may grant access to their data in two different ways.

(696) *First,* on a one-on-one basis, i.e. donor apps grant third parties access to their data by specifically allowlisting the apps they want to share their data with. This allows the recipient app to access all data shared with it by the relevant donor apps, but the issues resulting from a siloed approach remain (i.e. need for granular and exhaustive identification and allowlisting of recipient apps by the donor app). This is particularly detrimental for smaller operators and new entrants, which might be less known by other developers and hence less likely to be explicitly allowlisted.

(697) *Second,* to address these shortcomings,[813] AppSearch's implementation also foresees a role-based access model that facilitates the granting of permissions and privileges to user-chosen apps, in particular role-specific access to data. This allows donor apps to share data with the holder(s) of a given role, without needing to identify, allowlist or integrate with every one of the potential role holders individually.[814] AppSearch defines three permissions that enable data sharing with particular roles:[815]

(a) Default assistant role via the *READ_ASSISTANT_APP_SEARCH_DATA* permission. Donor apps can opt into sharing their data with the default assistant role.[816] Any third-party app may hold the default assistant role (and thus obtain this permission and access the associated data) but only one app can hold that role at a time.[817]

(b) System intelligence role via the *READ_GLOBAL_APP_SEARCH_DATA* permission. Donor apps are opted in by default into sharing their data with this role and may opt out.[818] Only pre-installed intelligence components such as Alphabet's Android System Intelligence may hold this role and thus obtain such permission, see Section 9.3.

(c) Home role via the *READ_HOME_APP_SEARCH_DATA* permission. Donor apps can opt-in into sharing their data with the home role.[819] Any third party

---

812 Alphabet's reply to RFI PO 7, Doc ID […], question 1, paragraph 2.

813 These shortcomings are well-known to Alphabet. See, for instance, https://source.android.google.cn/docs/core/permissions?hl=en, Doc ID […], accessed on 5 June 2026: "*Android 10 introduced the concept of a role, a unique name within the system associated with certain requirements and privileges. Assign roles to apps to grant them permissions for a specific purpose, and configure default roles using platform config resources*". See also https://developer.android.com/reference/android/app/role/RoleManager, Doc ID […], accessed on 9 April 2026: "*Upon becoming a role holder, the application may be granted certain privileges that are role specific. When the application loses its role, these privileges will also be revoked*".

814 Alphabet's reply to RFI PO 7, Doc ID […], question 1, paragraphs 3 to 6.

815 Alphabet's reply to RFI PO 7, Doc ID […], question 1, paragraphs 3 to 6.

816 Alphabet's reply to RFI PO 7, Doc ID […], question 1, paragraph 14.

817 See https://cs.android.com/android/platform/superproject/+/android-latest-release:packages/modules/Permission/PermissionController/res/xml/roles.xml, Doc ID […], last accessed on 20 April 2026, lines 105, 109, 110.

818 Alphabet's reply to RFI PO 7, Doc ID […], question 1, paragraph 14. See also Technical meeting Alphabet/Commission of 19 February 2026, Doc ID […], slide deck page 16.

819 Alphabet's response to RFI PO 7, Doc ID […], question 1, paragraph 14.

may hold the home role (and thus obtain this permission and access the associated data) but only one app can hold that role at a time.[820]

*9.2.4. Feature falls within the scope of Article 6(7) DMA*

(698) The Commission considers that the centralised access to apps' data stored on-device feature as described in Section 9.2.2 – together with its functionalities – is controlled via Google Android and falls within the scope of Article 6(7) DMA.

*9.2.5. Current access for Alphabet's own services and hardware*

(699) The centralised access to apps' data stored on-device feature is available to and used by Alphabet.

(700) The current implementation of the centralised access to apps' data stored on-device feature is done via AppSearch. AppSearch is designed and developed by Alphabet, part of AOSP, and shipped as a non-modifiable component of Google Android.[821] As explained above, Alphabet controls the features that form part of Google Android and under which conditions those features can be accessed and by whom. Alphabet decides, for instance, how AppSearch is accessed and used, including what role-specific permissions exist, who can hold those roles, whether roles are made available to a single or various concurrent holders, whether data is shared with each of those roles on an opt-in or opt-out basis, etc.

(701) Gemini currently uses the centralised access to apps' data stored on-device feature. When a service is set as the default assistant, it gains the *READ_ASSISTANT_APP_SEARCH_DATA* permission, and therefore has access to all data shared by donor apps with the default assistant role. Alphabet's Gemini comes as the default assistant on Pixel (Pixel 9 and later devices)[822] and other OEMs' devices[823] since 2024,[824] thereby having out-of-the-box access to, and/or using, that feature and the associated data in all of those devices.

(702) The centralised access to apps' data stored on-device feature, in its current implementation via AppSearch, is also currently powering different Alphabet AI-related projects and use cases. For instance, Alphabet has confirmed that [Describes Alphabet's product design choices regarding Magic Cue].[825] [Describes Alphabet's

---

820 See https://cs.android.com/android/platform/superproject/+/android-latest-release:packages/modules/Permission/PermissionController/res/xml/roles.xml, Doc ID […], accessed on 10 April 2026, lines 437, 440, 441.

821 See https://source.android.com/docs/core/ota/modular-system/appsearch#appsearch-architecture, part of the AOSP documentation website, Doc ID […], accessed on 20 April 2026, "*AppSearch […] was shipped as an APEX in Android 12. In Android 13, AppSearch is shipped as a Mainline module*". See also https://source.android.com/docs/core/ota/modular-system, Doc ID […], accessed on 20 April 2026, "*Mainline modularizes some Android system components and enables them to be updated outside of the normal Android release cycle. [...] Mainline converts selected system components into modules*". [Description of confidential commercial agreement]. Annex Q13.2 - DMA.100210 - EC AI RFI #2 of 23 September 2025 (8 October 2025), Doc ID […], submitted in Alphabet's reply to RFI AI 2 of September 2025. [Description of confidential commercial agreement].

822 See https://support.google.com/pixelphone/answer/15283615?hl=en, Doc ID […], accessed on 24 April 2026.

823 Alphabet's internal document, internal email of 7 September 2024 at 6:39:23 AM, with subject [Alphabet internal document], Doc ID […].

824 For completeness, OEMs currently set Gemini as the default assistant on almost all Google Android mobile devices – see Section 5.4.2.

825 Alphabet's response to RFI PO 7, Doc ID […], question 1, paragraph 17(ii).

product design choices regarding Magic Cue].[826] [Describes Alphabet's product design choices regarding Magic Cue].[827] [Describes Alphabet's product design choices regarding Magic Cue].[828]

(703) [Describes Alphabet internal documents regarding importance of this access for AI-related projects]. Internal documents refer to AppSearch as fundamentally supporting [describes Alphabets' product design choices].[829] [Describes Alphabet's product design choices].[830] [Describes Alphabet's product design choices].[831]

(704) Alphabet argued that [direct quote describing Alphabet's commercial and strategic positions].[832] [Description of how Alphabet is sharing data from its apps on-device with its services].

*9.2.6. Current access for third-party services and hardware*

(705) Third parties do not enjoy equal access to the centralised access to apps' data stored on-device feature, including in its current implementation via AppSearch, in an equally effective manner as Alphabet's own services.[833]

(706) Alphabet allows any app to be appointed as the default assistant role holder and thus be granted the associated *READ_ASSISTANT_APP_SEARCH_DATA* permission. However, this permission is role-specific and can only be granted to and used by one holder at a time, see Sections 5.2.3 and 5.2.4.[834] In other words, access to the feature is limited to one app only (i.e. the default assistant role holder) and cannot be concurrently obtained by all assistants that may be installed on a particular device even if the user so wishes.[835]

(707) As a result, only one app (the default assistant role holder) can access the centralised access to apps' data stored on-device feature at a time.[836] This limits the user value of all but the default assistant, further locking-in users to the incumbent default assistant and preventing effective multi-homing.

(708) Finally, third parties do not have equal access to Alphabet first-party apps' data stored on-device. The parallel implementation enabling centralised access to Alphabet's apps' data stored on-device is only accessible to Alphabet itself.

---

826 Alphabet internal document, [Alphabet internal document], Doc ID […], pages 2 and 3. Alphabet internal document, [Alphabet internal document], Doc ID […]. Alphabet internal document, [Alphabet internal document], Doc ID […], page 40.

827 Alphabet internal document, [Alphabet internal document], Doc ID […], page 40.

828 Alphabet internal document, presentation [Alphabet internal document], Doc ID […], pages 26, 27 and 29.

829 Alphabet internal document, Presentation [Alphabet internal document], Doc ID […].

830 Alphabet's internal document, Document [Alphabet internal document], Doc ID [...], pages 2 and 9. Alphabet's internal document, [Alphabet internal document], Doc ID […], page 3.

831 Alphabet's internal document, Presentation, [Alphabet internal document], Doc ID […], page 69.

832 Alphabet's reply to RFI PO 7 of 27 February 2026, Doc ID […], question 3c, paragraph 19.

833 As regards access via the *READ_GLOBAL_APP_SEARCH_DATA* permission, Section 9.3 applies.

834 For completeness, the same applies to the home role, which can only be held by a single package at a time.

835 Alphabet's reply to RFI PO 7 of 27 February 2026, Doc ID […], question 1, paragraph 7.

836 In other words, if, on a device, App A is the default assistant role holder, only App A has access to the centralised access to apps' data stored on-device feature. For App B to have access to the feature, the user must change the default role holder from App A (who loses access to the feature as a result) to App B (who will have access to the feature only as long as it stays the default assistant role holder). This is the case irrespective of whether an app has been preinstalled or downloaded by the user. Even if an app is preinstalled it cannot access the feature unless it is appointed as the default assistant role holder.

*9.2.7. Alphabet's response to the Preliminary Findings*

(709) *First*, Alphabet argues that the measures for the centralised access to apps' data stored on-device feature would be unlawful because they are based on a wrongly-defined feature. Alphabet submits that the feature does not exist and is therefore not available to Alphabet. According to Alphabet, imposing interoperability on a feature that does not exist requires Alphabet to create a new feature. Alphabet also claims that the mischaracterisation of the feature affects developers' control over their data and its alleged misdescription in some of the RFIs addressed to third parties undermines the evidence gathered by the Commission.[837]

(710) *Second*, Alphabet argues that the measures would constitute an unlawful data-sharing obligation, and that the Commission wrongly treats the data itself as a feature of the OS. Alphabet submits that the measures are arbitrarily asymmetric, and there is no basis under Article 6(7) DMA to apply unequal conditions to first-party and third-party services so that only Alphabet's first-party app data is subject to such data-sharing obligations.[838]

(711) *Third*, Alphabet argues that the measures for the centralised access to apps' data stored on-device feature would be unlawful because equally effective interoperability already exists. Alphabet claims that AppSearch leaves data access in developers' hands, and that first-party and third-party services can obtain AppSearch's read permissions under the same conditions.[839] Alphabet also argues that, to the extent a delta in the access exists, associating the *READ_ASSISTANT_APP_SEARCH_DATA* permission with specific roles is a strictly necessary and proportionate integrity measure. According to Alphabet, allowing an unknown and unbound set of apps to simultaneously hold the assistant permission would lead to developers revoking their opt-in, destroying the feature.[840]

(712) *Fourth*, Alphabet argues that the measures are disproportionate in the specific circumstances of Alphabet and Google Android. Alphabet claims that the measures would have harmful consequences for developers and users. They would allegedly diminish the role of the donor apps in deciding which apps are able to read their data, with the consequence that end users would have less control over which apps are able to access their data.[841] Alphabet contends specifically that the measures to make data that is stored on device by Alphabet's first-party apps accessible, are disproportionate because they create an onerous and unjustified obligation on first-party apps to share data with any third party.[842]

*9.2.8. The Commission's assessment*

9.2.8.1. Concerning the existence and description of the feature

(713) Alphabet's arguments that no feature of Google Android fits the description of the centralised access to apps' data stored on-device feature and that AppSearch is wrongly defined, because it "*is not a feature for "centralised access to data stored on device"*",[843] are contradictory and based on a mischaracterisation of the Commission's findings.

---

837 Alphabet's response to the PFs, Doc ID […], paragraphs 29-31, 212-220.
838 Alphabet's response to the PFs, Doc ID […], paragraphs 212 and 221.
839 Alphabet's response to the PFs, Doc ID […], paragraphs 212, 222-224 and 226.
840 Alphabet's response to the PFs, Doc ID […], paragraphs 212, 225 and 226.
841 Alphabet's response to the PFs, Doc ID […], paragraphs 212, 227 and 228.
842 Alphabet's response to the PFs, Doc ID […], paragraphs 212, 227 and 228.
843 Alphabet's response to the PFs, Doc ID […], paragraph 215.

(714) It is clear from the PFs[844] and from this Decision[845] that it is AppSearch's opt-in role-based centralised access via the default assistant role permission and not the whole of AppSearch that is considered to be the current implementation of the centralised access to apps' data stored on-device feature.

(715) Indeed, Alphabet's response to the PFs confirms that this was well understood by Alphabet,[846] as Alphabet in fact concedes that the centralised access to apps' data stored on-device feature does exist: "*AppSearch is, therefore, a mechanism for […] app-to-role data sharing […] The only "feature" of AppSearch that could be seen as affording some degree of more centralised access logic is the ability to hold a role that receives AppSearch data categorically […] rather than just by app name*".[847] This confirms the Commission's position that the feature does exist and is available to Alphabet.

(716) Compliance with these measures therefore does not require Alphabet to build a new feature. Alphabet does not substantiate why compliance with the measures would require it to do so. The measures make it clear that effective interoperability with this feature may be enabled by allowing user-chosen third-party apps to concurrently access the data that developers have chosen to share on an opt-in basis via the *READ_ASSISTANT_APP_SEARCH_DATA* permission by, for instance, making it possible for that permission (which is currently available to the default assistant role holder only) to be granted to various apps concurrently. Alphabet does not argue that this would be impossible to achieve, nor does it explain how this would entail the creation of a new feature.

(717) Alphabet's claim that the measures affect developers' control over their data is unsubstantiated: it does not explain how the measures interfere with or reduce developers' agency over their data. Alphabet's claim is also incorrect: the measures do not interfere with developers' choice to donate their data. Developers are still free to decide whether to donate data for the feature and what data to donate, if any. The measures have no effect on this. Developers' agency over their data and their ability to decide whether to donate their data to the feature and thereby share it with user-chosen apps remains unchanged. The change brought about by the measures is limited to enabling users to make that same data that developers have decided to share with user-chosen apps to be concurrently accessible to all apps they choose.

(718) Alphabet's claim that the evidence gathered on this feature is undermined by a mischaracterisation of AppSearch in some of the RFIs sent to third parties is also misguided. Alphabet does not substantiate what evidence has been undermined, how or why. Even if some evidence was undermined (*quod non*), Alphabet does not substantiate how that affects the Commission's findings on the centralised access to apps' data stored on-device feature in this case. In any event, the Commission's description of AppSearch in the RFI to AI Assistants[848] closely mirrors Alphabet's publicly available description of AppSearch[849] and the responses by third parties

---

844 Notably paragraphs 432, 436 and 438 of the PFs and paragraph 25(a) of the Annex to the PFs.

845 Notably paragraphs (691), (695) and (697) of this Decision and paragraph 25(a) of the Annex to this Decision.

846 Alphabet's response to the PFs, Doc ID […], paragraphs 222-226.

847 Alphabet's response to the PFs, Doc ID [...], paragraph 216.

848 In Alphabet's response to the PFs, Doc ID [...], paragraph 219, it claims that [direct quote describing Alphabet's commercial and strategic positions] is a misleading way to characterise the feature.

849 See https://developer.android.com/develop/ui/views/search/appsearch, Doc ID [...], accessed on 20 April 2026: "*With […] PlatformStorage […] your application contributes to a system-wide central index. […] Data access within these central indexes is restricted to data your application has contributed […]*".

(including [AI service provider's] response, which Alphabet cites)[850] demonstrate a correct understanding of the feature, evidencing that any potential misdescription of the feature (*quod non*) did not have any undermining effect on the evidence gathered.

9.2.8.2. Concerning access to Alphabet's apps' data stored on-device

(719) Alphabet's argument that the measures in relation to Alphabet's first-party apps' data stored on-device constitute an unlawful data-sharing obligation[851] is incorrect.

(720) At present, Alphabet does not share its apps' data stored on-device with third parties via the AppSearch implementation of the centralised access to apps' data stored on-device feature (i.e. the *READ_ASSISTANT_APP_SEARCH_DATA* permission) but has in parallel recreated a version of the feature that it makes available to its first-party services only.[852] Internal documents show Alphabet's [description of Alphabet's intention regarding access to Alphabet's first-party app data by other assistants].[853] For the reasons explained in Section 4.5.3, Alphabet should be prevented from using a parallel implementation of the centralised access to apps' data stored on-device feature accessible only to Alphabet's apps, such as Gemini, as this would undermine the effectiveness of the measures attached to this Decision in relation to this feature. Contrary to Alphabet's claims, these measures do not treat the data itself as a feature of the OS,[854] but are rather an effectiveness safeguard necessary to ensure that the interoperability solution for the centralised access to apps' data stored on-device feature is equally effective as the solution available to Alphabet.

(721) Alphabet's apps include Android flagship apps such as Gmail, Calendar, Docs, Drive, Messages, Dialer, YouTube, Maps, Chrome, Meet, and Google Health. Many of them come pre-installed on Android devices and are in fact themselves core platform services listed in the Designation Decision, corroborating their relevance and gateway status from the perspective of end users. This places Alphabet in an unparalleled position: no other developer is in a position to gate such fundamental parts of the Android ecosystem.

(722) In this light, equal effectiveness would not be achieved if Alphabet could allow its own first-party services to access Alphabet first-party apps' data stored on-device while preventing third parties from doing so. Restricting access to these apps' data would prevent third-party AI assistants from meeting basic user expectations. For example, Gemini could use on-device data from the Google Chrome or Google Maps apps to quickly answer a user query about a recently visited website or restaurant by accessing the browsing history or the location history, respectively, while third-party AI assistants would have no way to do so, even if the user so wishes. At the same time, where set as the default assistant, Gemini would still be able to access all third-party apps' data made accessible by third-party developers via the centralised access to apps' data stored on-device feature.

(723) In the absence of access to Alphabet first-party apps' data stored on-device, any interoperability solution for this feature would not provide equally effective interoperability to third parties. Therefore, for equally effective interoperability, the

---

850 [AI service provider's] reply to RFI 1 of 20 February 2026, Doc ID [...], question 4.

851 Alphabet's response to the PFs, Doc ID [...], paragraph 221.

852 See paragraphs 447 to 449 of the PFs, and paragraphs (702) to (704) of this Decision.

853 Alphabet's internal document, [Alphabet internal document], Doc ID […], pages 1, 2 and 4: [Direct quote describing Alphabet's commercial and strategic discussions].

854 Alphabet's response to the PFs, Doc ID […], paragraph 221.

same Alphabet first-party apps' data should be accessible to user-chosen third-party service(s), via the applicable permission settings.[855]

9.2.8.3. Concerning the existing level of interoperability and integrity

(724) Alphabet claims that effective interoperability with the centralised access to apps' data stored on-device feature already exists because AppSearch's conditions apply equally to first-party and third-party apps, and third parties can also become the default assistant role holder (thereby obtaining access to the *READ_ASSISTANT_APP_SEARCH_DATA* permission and the associated data). This however does not amount to equally effective interoperability. The feature being accessible to various services on a concurrent basis is necessary to ensure equally effective interoperability.

(725) Alphabet determines AppSearch's design and functioning, including whether AppSearch permissions are tied to a particular role. It is also Alphabet who defines the relevant roles and the applicable conditions, including whether a role is exclusive (i.e. can only be held by one app at a time). It is therefore Alphabet's decisions to, first, tie access to this feature to the default assistant role and, second, make the default assistant role exclusive, that limit equally effective interoperability.

(726) The Commission also considers that linking access to the feature to the exclusive default assistant role is not a strictly necessary and proportionate integrity measure.

(727) *First,* Alphabet puts forward arguments which in fact relate to the protection of the user's privacy (or even developer buy-in) rather than protection of operating system integrity. The Commission nonetheless further addresses Alphabet's arguments in this respect.

(728) *Second,* Alphabet emphasises the importance of user choice and consent.[856] However, access to the feature is currently not subject to user consent but is automatically granted to the default assistant role holder at the moment it is set as the default. The user is not asked to explicitly consent to this access. Notwithstanding this, the Commission agrees on the importance of user choice and consent. This is reflected in the measures, which foresee the possibility for access to the feature being subject to user consent (see Section 12.2).

(729) *Third,* Alphabet emphasizes the alleged risks stemming from "*an unbounded set of self-declared "assistants"*" and "*an unknown and unbounded set of apps [...] simultaneously hold[ing] the assistant permission*".[857] This is however in contradiction with Alphabet's current approach to this permission. As noted above, Alphabet currently allows any app to hold the default assistant role and thus obtain this permission. Any "unknown self-declared assistant" can currently be made the default assistant role holder and hence obtain access to the permission. In the same vein, there are currently no limits preventing users from changing the default assistant role holder as many times as they see fit, effectively allowing an "unbounded set of apps" to obtain access to the permission. In other words, the alleged risks flagged by Alphabet would already exist, but Alphabet has to date not implemented any mitigating measure. This indicates that the alleged risks either do not exist or are not sufficiently significant to justify mitigating measures.

---

855 For example, Alphabet could implement this by setting the data's visibility to grant access to those holding the *READ_ASSISTANT_APP_SEARCH_DATA* permission.

856 Alphabet's response to the PFs, Doc ID [...], paragraph 225.

857 Alphabet's response to the PFs, Doc ID [...], paragraph 225.

(730) In light of the above, the main modification to the *status quo* brought about by the measures has to do with the feature being concurrently accessible to more than a single app at a given moment, instead of access being limited to a single app at a time. In any event, Alphabet has not explained how the concurrency factor itself would affect integrity, or why is it that concurrency alone would exacerbate any alleged integrity risks.

(731) Notwithstanding this, in light of Alphabet's submissions and of the input received in the context of the public consultation on this point,[858] the Commission has adjusted the measures related to the centralised access to apps' data stored on-device feature. Due to the sensitivity of this feature, this Decision explicitly allows Alphabet to limit access to this feature only to third parties that meet certain eligibility criteria, which include criteria relating to integrity, security, and privacy (see Section 12.4). This reflects the nature and purpose of the feature (i.e. to enable centralised access for user-chosen apps to all apps' data stored on-device that developers have decided to share with user-chosen apps, and in particular assistant apps) while ensuring equally effective interoperability.

9.2.8.4. Concerning proportionality

(732) Alphabet argued in its response to the PFs that the measures for centralised access to apps' data stored on device are disproportionate. However, Alphabet does not substantiate why the measures would be more restrictive than necessary to achieve the objectives of Article 6(7) DMA. In particular, Alphabet fails to propose any less restrictive, yet equally effective means of complying with that provision. Nor does Alphabet explain and substantiate why the feature and/or a specific measure imposes a disproportionate burden on Alphabet. Instead, its arguments focus on questioning the scope of Article 6(7) and the actual need for interoperability – a line of reasoning which does not meet the legal standard for a valid proportionality claim, as explained in Section 13 below.

(733) In particular, Alphabet's claim that the measures are disproportionate is limited to stating that the measures would have harmful consequences for developers and end users. But Alphabet does not explain what the harmful consequences would be or how those alleged harmful effects would be disproportionate against achieving equally effective interoperability and, ultimately, the DMA's objectives of contestability and fairness.

(734) In any event, the measures for the centralised access to apps' data stored on-device feature enable concurrent access to the feature (which is necessary to ensure equally effective interoperability – see Section 9.2.8.3) and their scope is limited to what is strictly necessary to achieve that end.

(735) The Commission nonetheless addresses each of Alphabet's arguments in this respect.

(736) *First,* Alphabet argued that access to the feature should be limited[859] to AI assistants to account for the fact that developers choose to share their data with the feature on

---

858 See, for instance, [third party's] contribution to the Commission's public consultation, Doc ID […], particularly paragraph 52; [OEM] contribution to the Commission's public consultation, Doc ID […], page 4, recommending "*a tiered access framework*"; [OEM's] contribution to the Commission's public consultation, Doc ID […], page 4, proposing "*a third-party service admission standard or a third-party service whitelist established by an independent committee*".

859 Alphabet's submission of 11 May 2026, Doc ID [...], pages 4 to 6; Alphabet's submission of 19 May 2026, Doc ID [...]; Alphabet's submission of 20 May 2026, Doc ID [...], pages 9 and 10; Alphabet's submission of 28 May 2026, Doc ID [...], pages 9 and 10; and Alphabet's email submission of 15 June 2026, Doc ID [...] and Doc ID [...].

the expectation that it is accessible to assistants alone. This is however not the case: as noted above, Alphabet currently allows any user-chosen app (i.e. not only assistants) to hold the default assistant role, thereby obtaining access to the feature. There is therefore no reason to depart from the current approach by now limiting the feature beneficiaries to AI assistants only, simply as a result of the measures specified in this Decision. This is especially so given that, as described above, the measures preserve developers' choice whether to donate data to the feature.

(737) Notwithstanding this, taking account of both Alphabet's submissions and the input received from third parties in the context of the public consultation on this point,[860] the Commission has adjusted the measures. Due to the sensitivity of this feature, this Decision explicitly allows Alphabet to limit access to this feature only to third parties that meet certain eligibility criteria, which include criteria relating to integrity, security, and privacy (see Section 12.4). In the same vein, as noted above, the Commission agrees with Alphabet on the importance of user consent,[861] and the measures therefore also foresee the possibility for access to the feature being subject to user consent, to ensure that only user-chosen apps gain access to the feature. The measures therefore allow to restrict access to the feature to trusted third parties without unnecessarily reserving it to assistants, helping to prevent the misuse of the feature, and reinforcing the protections for developers and end users while ensuring equally effective interoperability for the purposes of Article 6(7) DMA.

(738) *Second,* Alphabet argued that the measures should be adapted to limit access to what is available to the Gemini assistant and similar services.[862] This is however excessively restrictive, especially considering the rapid evolution of the AI industry, which may lead to new services soon being deployed that, despite not being similar to Gemini, could access, and benefit from access to, the feature. In light of this, the Commission concludes that the scope of "Alphabet's AI-powered services, such as Gemini" is to be maintained.

(739) *Third,* as regards the measures foreseen in relation to Alphabet's first-party app's data stored on-device (in particular the measures in paragraph 25(b) of the Annex to this Decision), Alphabet's claim that they are disproportionate is limited to stating that these allegedly create an onerous and unjustified data-sharing obligation requiring to share data with any third party.[863] Alphabet does not explain what the harmful consequences would be.[864] The arguments pertaining to Alphabet's first-party apps are in any event addressed in general in Section 4.5.3 and specifically for this feature in Section 9.2.8.2 above.

---

860 Various contributions to the Commission's public consultation point at potential security- and privacy-related concerns resulting from data access – see for instance [OEM's] (Doc ID [...]), [OEM's] (Doc ID [...]) and [third party's] (Doc ID [...]). Various contributions also argue that these concerns may be addressed through measures ensuring that data access is subject to appropriate requirements – see for instance the [third party's] (Doc ID [...]), [third party's] (Doc ID [...]) and [third party's] (Doc ID [...]) contributions to the Commission's public consultation.

861 Alphabet's submission of 20 May 2026, Doc ID [...], pages 9 and 10.

862 Alphabet's submission of 20 May 2026, Doc ID [...], pages 9 and 10, and Alphabet's submission of 17 June 2026, Doc ID [...].

863 Alphabet's response to the PFs, Doc ID [...], paragraph 227.

864 Alphabet quotes [OEM's] contribution to the Commission's public consultation, Doc ID [...], page 6, but [OEM's] statement simply reflects a generic data protection concern with no connection to Alphabet's first-party apps.

(740) Notwithstanding this, and notably in light of Alphabet's submissions in this regard,[865] the Commission has adjusted the measures in the Annex to this Decision to, first, explicitly allow Alphabet to limit access to the feature only to third parties that meet certain eligibility criteria, which include criteria relating to integrity, security, and privacy (see Section 12.4) (which is complemented by the possibility of access being subject to user consent), and second, to limit the scope of the data to be made accessible for third parties to Alphabet's first-party apps' data stored on-device that is shared with Alphabet's AI-powered services, such as Gemini, for the purposes of answering a user query. This guarantees that the data captured by the measure is limited to Alphabet's first-party apps' data stored on-device which is made available to and accessed by Alphabet's AI-powered services, such as Gemini, for the primary use case that the measures for the centralised access to apps' data stored on-device feature are meant to address (i.e. answering a user query).

(741) The Commission therefore concludes that there is an interoperability gap with respect to the centralised access to apps' data stored on-device feature of Google Android. As a result, third parties cannot perform important functionalities that are expected of an AI assistant which places them sub-par in comparison with Google's own AI-powered services. As a result, the Commission considers that it should specify how Alphabet must fill this gap to comply with Article 6(7) DMA.

*9.2.9. Measures that Alphabet should implement*

(742) To meet the requirements of Article 6(7) DMA, Alphabet should implement the measures set out in Section 2.1 of the Annex to this Decision. The interoperability solution for the centralised access to apps' data stored on-device feature, as described in Section 9.2.2 of this Decision, must be equally effective as that available to Alphabet's AI-powered services, such as Gemini.

(743) The measures that Alphabet should implement are threefold.

(744) *First,* third-party services must be capable of understanding the user's context and efficiently use it to deliver rich, personalised services under the same conditions as Alphabet's AI-powered services, such as Gemini. Alphabet must therefore ensure that third-party services can access, search and retrieve the on-device-stored apps' data that apps choose to share on an opt-in role-based basis in a centralised manner via on-device databases or data-sharing platforms.

(745) *Second,* Alphabet should be prevented from recreating and using a parallel version of the centralised access to apps' data stored on-device feature for sharing and accessing Alphabet's first-party apps' data where such parallel version is available to Alphabet's first-party services only. This would otherwise shield Alphabet's first-party apps' data from third-party services and undermine the effectiveness of the interoperability solution in relation to this feature. Alphabet must therefore ensure that third-party services can also access Alphabet's first-party apps' data stored on-device which is shared with Alphabet's AI-powered services, such as Gemini, for the purposes of answering a user query, and that they can do so under the same conditions as Alphabet's AI-powered services, such as Gemini.

---

865 See Alphabet's submission of 11 May 2026, Doc ID [...], pages 4 to 6; Alphabet's submission of 20 May 2026, Doc ID [...], pages 9 and 10; Alphabet's submission of 28 May 2026, Doc ID [...], pages 9 and 10; Alphabet's submission of 3 June 2026, Doc ID [...]; Alphabet's submission of 15 June 2026, Doc ID [...]; Alphabet's email of 15 June 2026, Doc ID [...]; Alphabet's submission of 18 June 2026, Doc ID [...]; and Alphabet's email of 18 June 2026, Doc ID [...].

(746) Finally, Alphabet should be prevented from limiting access to the feature, for instance by tying access to the holding of exclusive roles, in a way that only a single service can access the feature at a time. This hinders equally effective interoperability, limits the user value of all but the one service accessing the feature, locks in users into the incumbent default apps and prevents effective multi-homing (Section 5.2.4). Alphabet must therefore ensure that the feature can be accessed and used by multiple services on a concurrent basis (i.e. access to and use of the feature is not tied to an exclusive role or in any way limited to a single app at a time).

(747) As explained in Section 12.4.5, the centralised access to apps' data stored on-device feature meets the exceptionality conditions that warrant limiting access to third parties that meet certain eligibility criteria. As explained in Section 12.2, access to the feature may also be subject to user consent.

(748) For the purpose of ensuring that effective interoperability continues in the future, Alphabet should also provide effective interoperability with any future updates, including new functionalities, of the centralised access to apps' data stored on-device feature insofar as they are available to Alphabet's own services or hardware. To this end, the general measures in Section 12 of this Decision apply, including in particular the measures concerning future updates and new functionalities set out in Section 12.9 in this Decision.

(749) To the knowledge of the Commission, [describes Alphabet's product design choices],[866] future updates or implementations of the centralised access to apps' data stored on-device feature may include [describes Alphabet's product design choices].[867]

*9.2.10. Timing for implementation of the measures*

(750) To meet the requirements of Article 6(7) DMA, and taking into account Alphabet's proposal and the available evidence,[868] Alphabet should implement the measures in the Annex (Section 2.1) according to the timelines indicated therein.

## 9.3. Measures for context-aware intelligence

*9.3.1. Introduction*

(751) AI service providers can offer users a wide variety of inconspicuous but helpful "intelligent experiences", personalised to the user's context and continuously present to enhance the user experience. Some are almost invisible, and serve a utility function: for example, "Screen Attention" keeps the screen on while a user is looking at their device. However, many experiences are directly user-facing, such as real-time subtitles (Live Caption).[869]

(752) These intelligent experiences also cover proactive suggestions, which represent an important step in the transition from reactive AI services to fully agentic AI capabilities. These experiences autonomously ("proactively") surface relevant

866 Alphabet internal document, presentation [Alphabet internal document], Doc ID […], pages 18 to 23, and 26; Alphabet internal document, Document [Alphabet internal document] of July 2025, Doc ID […], pages 3-6.

867 Alphabet internal document, presentation [Alphabet internal document], Doc ID […], page 1. Alphabet internal document, presentation [Alphabet internal document], Doc ID […], page 36. Alphabet internal document, Presentation [Alphabet internal document], Doc ID […], page 6.

868 Technical meeting with Alphabet of 12 June 2026, Doc ID [...], slide deck page 13. Alphabet's reply to RFI 21 of 8 June 2026, Doc ID [...], question 6, paragraph 17.

869 Alphabet's reply to RFI AI 6 of 23 February 2026, Doc ID [...], question 37, paragraph 169; Alphabet internal document, [Alphabet internal document] Presentation of July 2025, Doc ID […], page 25.

information and recommend helpful actions across apps.[870] As a result, services can skip the need for a user to manually initiate an interaction: the user does not need to find, open and search inside and across different apps to pull the needed information. The information or action just "magically" appears in the right place at the right time.

(753) These suggestions are based on the user's data collected from apps, recently viewed content, and the user's context in general.[871] In their current form, these proactive suggestions appear on surfaces inside, on top of, and outside of apps.[872] Suggestions usually consist of a short text and the icon of the app to which the suggestion leads.[873] For example, the proactive suggestion can be a helpful action such as "View calendar" when a friend sends an invitation to dinner via a message. The suggestions can also be a richer combination of data. For example, the user can get a proactive suggestion about a relevant flight booking when calling an airline, showing the flight number, destination, date and time, and booking confirmation for the flight.[874]

(754) Together, these proactive suggestions and intelligent experiences make the use of AI effortless for users and give a sense of true assistance and deep integration with what the user is doing, at any time.

(755) On the Google Android operating system, these proactive suggestions and intelligent experiences rely on the functionalities of the context-aware intelligence feature. These functionalities are necessary to let Alphabet and third parties implement their own intelligence solutions to the benefit of their services. This is enabled by access to resources within the operating system, in particular physical and digital context with a level of continuous access required to enable proactivity as well as the ability to surface the experience across apps or at the operating system level. For instance, *Now Playing*, which automatically displays information about the current song playing nearby, always listens to the microphone audio, and when it detects a song via an on-device database, it proactively displays the song's information on the device's lock screen.

(756) Alphabet's internal documents confirm that context-aware intelligence is an important driver to offer "*innovative new AI services*" in the future.[875] For example, access to ambient data (i.e. core sensor data, see also Section 9.4) is [quotes from Alphabet internal document discussing importance of context-aware intelligence for building proactive agents and creating truly intelligent experiences].[876] Alphabet has been steadily developing more intelligent experiences, including Pixel-exclusive experiences such as *Now Playing*.[877] The lessons Alphabet learned from developing these experiences, coinciding with the overall evolution of AI, paved the way for more advanced experiences. The latest embodiment of this is Alphabet's Magic Cue,[878] which uses the functionalities of the context-aware intelligence feature to

---

870 See https://support.google.com/pixelphone/answer/16508057?hl=en, Doc ID […], accessed on 15 April 2026.

871 Alphabet internal document, [Alphabet internal document] of August 2025, Doc ID […], pages 3, 5.

872 Alphabet's reply to RFI PO 3 of 13 February 2026, Doc ID [...], Annex Q24.8, Doc ID [...], page 8.

873 Alphabet's reply to RFI PO 3 of 13 February 2026, Doc ID [...], Annex Q24.8, Doc ID [...], page 6.

874 Alphabet's reply to RFI PO 3 of 13 February 2026, Doc ID [...], Annex Q24.8, Doc ID [...], page 7.

875 Alphabet internal document, Presentation [Alphabet internal document] of Q4 2025, Doc ID […], page 2.

876 Alphabet internal document, Document [Alphabet internal document] of January 4, 2026, Doc ID […], page 3.

877 Alphabet's reply to RFI AI 6 of 23 February 2026, Doc ID […], question 37, paragraph 169.

878 Wherever Magic Cue is mentioned in this section, it is discussed as an implementation of a solution that uses the context-aware intelligence feature as described.

offer proactive suggestions on Pixel devices, and which is deeply integrated with Alphabet's first-party apps, including Gemini.

(757) Alphabet's framework for context-aware intelligence has been crystallizing with [description of Alphabet's product design choices and commercial and strategic discussions],[879] [description of Alphabet's product design choices and commercial and strategic discussions].[880] [Description of Alphabet's product design choices and commercial and strategic discussions].[881] [Description of Alphabet's product design choices and commercial and strategic discussions],[882] [description of Alphabet's product design choices and commercial and strategic discussions]. One internal document describes how the evolution of OS-based proactive suggestions makes them [direct quote describing Alphabet's commercial and strategic discussions].[883]

(758) Access to the functionalities of the context-aware intelligence feature, such as ambient data, personal context and relevant surfaces, is very important for this vision. Currently, Alphabet only makes the context-aware intelligence feature accessible to OEMs and Alphabet itself, via privileged permissions and thus OEM pre-installation, with very limited support via public APIs.[884] Third parties do not have access to this feature in an equally effective manner, even though they seek better interoperability, as there is great interest in offering solutions for proactive suggestions,[885] as well as for other intelligent experiences, such as live captions or live translation, including to offer better privacy guarantees.[886] This lack of feature access puts third parties at a competitive disadvantage, as it prevents them from offering alternative proactive suggestions and intelligent experiences (including experiences with equal or better performance, e.g. better speech recognition, greater language support, or more relevant proactive suggestions) or innovative capabilities on an equal footing with Alphabet.[887] This lack of feature access also prevents third parties from offering alternative services in a manner that is equally accessible to users (e.g. on the lock screen).

---

879 Alphabet internal document, Presentation [Alphabet internal document] of January 2026, Doc ID […], page 11.

880 Alphabet internal document, internal email of 29 January 2026 [Alphabet internal document], Doc ID […], page 3.

881 Alphabet internal document, [Alphabet internal document], Doc ID […], page 6.

882 Alphabet internal document, Presentation [Alphabet internal document] of July 2025, Doc ID […], pages 12 and 40 ([describes Alphabet's product design choices]). Alphabet internal document, Presentation [Alphabet internal document] of 2025, Doc ID […], pages 2, 9, 12. Alphabet internal document, Presentation [Alphabet internal document], Doc ID […], pages 4, 5, 8, 20, 22.

883 Alphabet internal document, [Alphabet internal document], Doc ID […], page 2.

884 Agreed minutes of meeting with [AI service provider] of 10 February 2026, Doc ID […], paragraphs 11, 13, 15.

885 For example, Alphabet has told a third party that [description of commercial discussion between Alphabet and third party on possibility to offer proactive suggestions]. Agreed minutes of meeting with [AI service provider] on 10 February 2026, Doc ID […], paragraph 11, 13. See also Agreed minutes of meeting with [AI service provider] on 12 March 2026, Doc ID […], paragraph 22. Agreed minutes of meeting with [AI service provider] on 2 March 2026, Doc ID […], paragraph 15. [AI service provider's] submission to the public consultation, Doc ID [...], page 19. [AI service provider's] submission to the public consultation, Doc ID […], page 3. [AI service provider's] submission to the public consultation, Doc ID […], page 2.

886 Agreed minutes of meeting with [AI service provider] of 2 March 2026, Doc ID [...], paragraph 13. Agreed minutes of meeting with [AI service provider] of 2 April 2026, Doc ID [...], paragraphs 9, 11, 15, 16.

887 Agreed minutes of meeting with [AI service provider] of 10 February 2026, Doc ID [...], paragraphs 11, 13, 15. Agreed minutes of meeting with [AI service provider] of 6 March 2026, Doc ID [...], paragraph 29.

(759) Specifically for proactive suggestions, this need for interoperability with the context-aware intelligence feature is reinforced by a lack of interoperability with existing proactive suggestions solutions on Android. Magic Cue, being Alphabet's implementation of proactive suggestions on Pixel devices, heavily favours Alphabet's ecosystem of apps and services, largely lacking interoperability for third parties.[888] In its PFs, the Commission considered imposing measures for interoperability with the proactive suggestions feature, which would have provided for high-level participation by third parties in solutions such as Magic Cue (see also Section 4.2).[889] Alphabet raised that "*proactive suggestions features* [sic] *are still at a nascent stage*"[890] and that "*proactive suggestions is a developing concept, and Google does not yet have a viable proactive suggestions product that meets Google's quality expectations and users' needs. This feature is therefore insufficiently mature*".[891] Moreover, as explained above, [description of Alphabet's commercial and strategic discussions]. In this light, the Commission considers that it is premature to specify the high-level proactive suggestions feature,[892] and instead imposes measures to focus Alphabet's efforts on enabling interoperability at a lower level, with the context-aware intelligence feature. This enables third parties to develop their own solutions for proactive suggestions and intelligent experiences, as an alternative to Magic Cue. These alternative solutions can offer the integrations with third-party apps that Alphabet's solution lacks, next to offering improvements over Alphabet's solution on other qualities, thereby creating competitive pressure in the high level. This is particularly important to enable context-aware intelligence in other services and ecosystems other than Alphabet's own, e.g. retrieving flight information from email services other than Gmail.

(760) Alphabet has access to all the necessary functionalities that serve as the building blocks for innovative proactive suggestions and intelligent experiences. As Alphabet is working on putting together these building blocks to offer the proactive AI services of the future, third parties can only properly compete with, or out-innovate, Alphabet in developing such innovative, proactive, and context-aware experiences on Android, if they also gain the same access to the context-aware intelligence feature as is available to Alphabet.

*9.3.2. Description of the feature*

(761) The context-aware intelligence feature breaks down into functionalities that let services build solutions in the form of software components (Section 9.3.5.1). These solutions take user context as inputs, then process it to derive relevant proactive

888 Only a few third-party apps, that Alphabet unilaterally selects, can surface Magic Cue suggestions. Alphabet's reply to RFI PO 7 of 27 February 2026, Doc ID […], paragraph 40. Moreover, Magic Cue will never suggest querying a third-party AI assistant like it does for Gemini. The surfacing of Gemini in Magic Cue even depends on Gemini being set as default. If a third-party AI assistant is set as default instead of Gemini, Magic Cue will not suggest to users actions in either Gemini or the non-Gemini default assistant. See https://blog.google/products-and-platforms/devices/pixel/march-2026-pixel-drop/#footnote-5, Doc ID […], accessed on 17 April 2026.

889 Section 9.3 of the PFs and Section 2.2 of the Annex to the PFs.

890 Alphabet's response to the PFs, Doc ID […], paragraph 240, second bullet.

891 Alphabet submission "Alphabet Second Mark-up of the draft measure" of 29 May 2026, Doc ID […], page 11.

892 This is without prejudice to the Commission maintaining its position that proactive suggestions is a feature in the meaning of Article 6(7) DMA and that several related features (even those at "different" layers") may need to be made interoperable (Section 4.2). Under Article 8(9) DMA, the Commission may decide to reopen proceedings where there has been a material change in any of the facts on which the decision was based or the measures as specified in the decision are not effective.

suggestions and intelligent experiences, and finally output those suggestions and experiences to a variety of surfaces in the operating system and in apps.

(762) The context-aware intelligence feature includes, amongst others, the following functionalities:

(a) **Access to inputs**: access to the user's context on a continuous basis and in the background as needed for proactivity.[893] This includes:

(1) Access to the user's physical context, which includes ambient data, such as data from the camera, microphone, accelerometer, proximity sensor, GPS, and other location signals (see also Section 9.4).[894]

(2) Access to the user's digital context, which includes, but is not limited to, device audio, screen contents and context, screenshots, relevant information on the foreground app, notifications, app launches, contacts, shortcuts, SMS, and app package details.[895]

(3) The ability to retrieve data provided ad-hoc by an app, service, or the operating system to the proactive suggestion or intelligent experience.[896] This data is provided as a one-off in the specific current user context. [Alphabet's internal system processes and components].

(4) Global access to apps' data and actions stored on-device that apps choose to donate, including through an on-device database or data sharing platforms allowing for centralised data access and sharing (see also Section 9.2).[897]

(b) **Access to processing resources**: access to processing resources to interpret user context and derive proactive suggestions and intelligent experiences. This includes:

(1) Ability to use on-device models to process the input data and generate the outputs.[898] Such on-device models can be both those already present on the device (see Section 11.2) and those implemented by third parties (see Section 11.3).

(2) Ability to use communication channels to send and receive data to the solution's cloud-based models to process the input data and generate the

---

893 Technical meeting Alphabet/Commission of 19 February 2026, Doc ID [...], slide deck pages 5, 9, 10, 22.

894 Technical meeting Alphabet/Commission of 19 February 2026, Doc ID [...], slide deck pages 5, 9, 10, 22. Alphabet internal document, Document [Alphabet internal document] of 8 February 2021, Doc ID [...], page 3. See Marchiori et al., "Private Compute Core Architecture", Google, 23 September 2022, https://arxiv.org/pdf/2209.10317v2, Doc ID [...], Table 1. Alphabet's reply to RFI 9 of 4 March 2026, Doc ID [...], Annex to question 28, second column.

895 Technical meeting Alphabet/Commission of 19 February 2026, Doc ID [...], slide deck pages 5, 22. Alphabet internal document, [Alphabet internal document] Document of 8 February 2021, Doc ID [...], page 3. See Marchiori et al., "Private Compute Core Architecture", Google, 23 September 2022, https://arxiv.org/pdf/2209.10317v2, Doc ID [...], Table 1. Alphabet internal document, [Alphabet internal document] Document, Doc ID [...], page 2.

896 Alphabet internal document, Document [Alphabet internal document] of 22 September 2025, Doc ID [...], page 1.

897 Technical meeting Alphabet/Commission of 19 February 2026, Doc ID [...], slide deck page 5.

898 Technical meeting Alphabet/Commission of 19 February 2026, Doc ID [...], slide deck page 5.

outputs.[899] This does not entail the ability for third parties to use Alphabet's cloud-based models.

(3) Ability to use protected execution environments allowing to process particularly sensitive data, such as ambient data, in isolation.[900]

(c) **Access to output channels and display surfaces**: access to output channels and display surfaces to present or make available the generated proactive suggestion or intelligent experience to the user. This includes:

(1) The ability to present proactive suggestions or intelligent experiences on a surface visible or audible to the user in the user interface, mediated by the OS. Such surfaces can take many forms. For instance, such surfaces may appear inside Alphabet's apps and third parties' apps, as part of system interfaces (e.g. lock screen or notifications), as "chips" or small insets in keyboards, as a screen overlay, etc.[901]

(2) The ability to transmit data from the proactive suggestion or intelligent experience to an app, service, or the OS.[902]

(3) The ability to trigger integrations and interactions with apps and the operating system from the proactive suggestion or intelligent experience, such as redirecting to another app, opening an app in an overlay, or deep linking to parts of the operating system such as settings.[903]

(763) These functionalities are combined and used to offer a variety of proactive suggestions and intelligent experiences, also described as "Device Intelligence" by Alphabet.[904] All these experiences rely, to one degree or another, on the context-aware intelligence feature and its functionalities. As examples:

(a) *Now Playing* automatically shows the name of a song that is currently playing near the user, without the need for the user to actively trigger the song detection. To achieve this, Now Playing captures the device's ambient microphone input, processes it to detect that music is playing and then to determine the concrete song, and finally surfaces the name of the song on the lock screen and notification shade.[905]

(b) *Live Translate* lets a user see a live translation of text conversations, audio, or video. This can be fully automatic, being triggered as soon as Live Translate detects the use of a foreign language (e.g. in a chat). For this, Live Translate accesses screen content, microphone input, and the camera, processes it with a translation model, and surfaces the translated text in a screen overlay.[906]

---

899 See https://blog.google/innovation-and-ai/products/google-private-ai-compute/, accessed on 17 April 2026, Doc ID [...].

900 Technical meeting Alphabet/Commission of 19 February 2026, Doc ID [...], slide deck page 25.

901 Technical meeting Alphabet/Commission of 19 February 2026, Doc ID [...], slide deck page 9 (keyboard).

902 Alphabet's response to RFI PO 7 of 27 February 2026, Doc ID [...], question 6.d, paragraph 38. Alphabet internal document, Presentation [Alphabet internal document], Doc ID [...], page 15. Alphabet internal document, Presentation [Alphabet internal document], Doc ID [...], page 28.

903 Alphabet's reply to RFI PO 3 of 13 February 2026, Doc ID [...], Annex Q24.8, Doc ID [...], page 13.

904 Alphabet internal document, Presentation [Alphabet internal document] of August 2025, Doc ID [...], page 29.

905 Alphabet's reply to RFI 9 of 4 March 2026, Document [Alphabet internal document], Doc ID [...], row "Now Playing" submitted in Alphabet's reply to RFI 9 of 4 March 2026, Doc ID [...].

906 Alphabet's reply to RFI PO 9 of 4 March 2026, Document "Annex Q28", Doc ID [...], row "Live translate" submitted in Alphabet's reply to RFI 9 of 4 March 2026, Doc ID [...].

(c) *Magic Cue* offers experiences such as [describes Alphabet's internal systems and processes regarding Magic Cue],[907] [describes Alphabet's internal systems and processes regarding Magic Cue].[908]

*9.3.3. Technical implementation of context-aware intelligence in Google Android*

9.3.3.1. Access by intelligence components

(764) Before explaining how the operating system provides the context-aware intelligence feature, it is helpful to explain how this feature is used and deployed in practice on Android.

(765) The context-aware intelligence feature is generally accessed by "intelligence components" (in this Section also referred to as simply the "components"), which are responsible for the reasoning and actual generation of proactive suggestions and intelligent experiences, using the functionalities of the context-aware intelligence feature.

(766) Currently, a monolithic "system intelligence" component, also called the "intelligent service",[909] is used for offering proactive suggestions and intelligent experiences on Android.[910] This component was first released in 2018 in Android 9.[911] OEMs must integrate this intelligence component into the Google Android version shipping on their device.[912] Most OEMs use an "off-the-shelf" implementation of this component from Alphabet, called Android System Intelligence (ASI).[913] Alphabet in its role as OEM for Pixel devices uses an additional intelligence component called Pixel System Intelligence (PSI) for Magic Cue and uses ASI for all other experiences.[914]

(767) Certain functionalities of the context-aware intelligence feature are subject to privileged permissions, including by being marked as system APIs (see Section 5.2.3).[915] Certain permissions are further tied to specific system intelligence roles[916]

---

907 Alphabet internal document, Document [Alphabet internal document] of April 2025, Doc ID [...], page 45.

908 Alphabet internal document, Document "Annex Q28", Doc ID [...], row "Magic Cue", submitted in Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID [...]. Technical meeting Alphabet/Commission of 19 February 2026, Doc ID [...], slide deck page 30.

909 Technical meeting Alphabet/Commission of 19 February 2026, Doc ID [...], slide deck page 34.

910 Technical meeting Alphabet/Commission of 19 February 2026, Doc ID [...], slide deck page 5; Alphabet internal document, Presentation [Alphabet internal document] of 2025, Doc ID [...], page 34.

911 See https://play.google.com/store/apps/details?id=com.google.android.as, accessed on 15 April 2026, Doc ID [...]. See https://www.androidauthority.com/android-system-intelligence-3325187/, Doc ID [...], accessed on 7 June 2026.

912 Technical meeting Alphabet/Commission of 19 February 2026, Doc ID [...], slide deck page 6. [content of commercial agreements], Alphabet internal document, Document "Annex Q13.2 – […] - EC AI RFI #2 of 23 September 2025 (8 October 2025) (Confidential - Contains Business Secrets).pdf", Doc ID [...], page 269 submitted in Alphabet's reply to RFI AI 2 of September 2025.

913 Also codenamed "*AiAi*". Technical meeting Alphabet/Commission of 19 February 2026, Doc ID [...], slide deck page 7; Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID [...], question 37, paragraph 172; Alphabet internal document, [Alphabet internal document] Document of 3 September 2025, Doc ID [...], pages 2, 3: [direct quote describing Alphabet's commercial and strategic discussions]. This component offers implementations of, among others, Live Caption and Smart Reply. See Marchiori et al., "Private Compute Core Architecture", Google, 23 September 2022, https://arxiv.org/pdf/2209.10317v2, Doc ID [...], Sections 3.1 and 3.2.

914 As another example of an OEM-specific intelligence component, [OEM] implements their own additional component called Personal Data Intelligence. Technical meeting Alphabet/Commission of 19 February 2026, Doc ID [...], slide deck pages 6, 30; Alphabet's reply to AI RFI 6 of 23 February 2026, Doc ID [...], paragraphs 174, 175.

915 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID [...], paragraphs 177.b, 181, 182.

916 These roles are *SYSTEM_UI_INTELLIGENCE* for system UI features (e.g. app prediction), *SYSTEM_AMBIENT_AUDIO_INTELLIGENCE* for processing ambient audio (e.g. Now Playing),

or to system services (see Section 5.2.4).[917] OEMs assign each role and bind system services to a given intelligence component,[918] which must be a system app and can only be a single app per role or service.[919] This assignment then automatically grants these (system) permissions.[920]

(768) [Descriptions of confidential commercial agreement].[921] [Description of confidential commercial agreement].[922] [Description of confidential commercial agreements].[923] [Description of Alphabet commercial and product design choices].[924]

(769) [Description of Alphabet's commercial and strategic discussions].[925] [Alphabet's internal system processes and components].[926] [927] [Description of Alphabet's commercial and strategic discussions].[928] [Description of Alphabet's commercial and strategic discussions].[929] Even under this evolution with ACE, the fact remains that the ability to provide proactive suggestions and intelligent experiences will still be dependent on access to the same context-aware intelligence feature and its functionalities, as outlined in Section 9.3.2, such as ambient data and operating system surfaces.[930]

---

*SYSTEM_AUDIO_INTELLIGENCE* for audio (e.g. Live Caption), *SYSTEM_NOTIFICATION_INTELLIGENCE* for notifications (e.g. suggesting replies and actions for notifications), *SYSTEM_SETTINGS_INTELLIGENCE* for surfaces in the Settings app, *SYSTEM_TEXT_INTELLIGENCE* for text (e.g. Live Translation, Autofill, Smart Reply), and *SYSTEM_VISUAL_INTELLIGENCE* for camera data (e.g. Screen Attention). See https://source.android.com/docs/core/permissions/android-roles, Doc ID [...], accessed on 8 April 2026. There is also a vendor intelligence role for an additional OEM intelligence component, being *SYSTEM_VENDOR_INTELLIGENCE.* https://cs.android.com/android/platform/superproject/+/android-latest-release:packages/modules/Permission/PermissionController/res/xml/roles.xml?q=SYSTEM_VENDOR_INTELLIGENCE, Doc ID [...], accessed 15 April 2026.

917 Such services include the *AugmentedAutofillService*, *ContentCaptureService*, and *TextClassifierService*. Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID [...], question 38.b, paragraph 182.

918 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID [...], question 37, paragraph 177.

919 See tags for SYSTEM_UI_INTELLIGENCE: "exclusivity=user", "static=true" and "systemOnly=true". See https://cs.android.com/android/platform/superproject/+/android-latest-release:packages/modules/Permission/PermissionController/res/xml/roles.xml, accessed on 20 April 2026, Doc ID [...]. Applies *mutatis mutandis* to the other system intelligence roles.

920 Alphabet internal document, [Alphabet internal document] Presentation of 2025, Doc ID […], page 43.

921 [Confidential commercial agreement], Alphabet internal document, Document [Alphabet internal document], Doc ID […], pages 268, 269, submitted in Alphabet's reply to RFI AI 2 of September 2025.

922 Alphabet internal document, Document [Alphabet internal document], Doc ID […], pages 8-10 submitted in Alphabet's reply to pre-opening RFI 4 of 28 November 2025, Annex Q3a.8, Doc ID […].

922 Alphabet internal document, Document [Alphabet internal document], Doc ID […], pages 8-10 submitted in Alphabet's reply to pre-opening RFI 4 of 28 November 2025, Annex Q3a.8, Doc ID […].

923 [Confidential commercial agreement], Section 7.21, Alphabet internal document, Document [Alphabet internal document], Doc ID […], page 270 submitted in Alphabet's reply to RFI AI 2 of 23 September 2025.

924 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID […], paragraphs 174, 175.

925 Alphabet internal document, Presentation [Alphabet internal document] of January 2026, Doc ID […], page 32.

926 Alphabet's reply to RFI AI 6 of 23 February 2026, Doc ID 1229, paragraph 194.d.

927 Alphabet internal document, Document [Alphabet internal document], Doc ID […], page 3.

928 Alphabet's reply to RFI AI 6 of 23 February 2026, Doc ID [...], paragraphs 37, 38; Alphabet internal document, Presentation [Alphabet internal document], Doc ID […], pages 3 and 13.

929 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID [...], paragraphs 43, 44, 45.

930 Alphabet internal document, Presentation [Alphabet internal document] of January 2026, Doc ID […], page 18.

9.3.3.2. Access to inputs and outputs

(770) The intelligence components that generate and provide proactive suggestions and intelligent experiences do so by accessing inputs, applying certain logic to process inputs, and generating and surfacing outputs, which form the functionalities of the context-aware intelligence feature, which is part of Google Android.

(771) Access to the user's current physical and digital context is necessary to observe what the user is doing, seeing, and hearing, to be able to provide an experience or suggestion that is relevant to that context, which must then be presented to the user. For example, if a chat conversation is shown on screen, a relevant reply can be suggested in the keyboard. This physical and digital context includes ambient data, as described in Section 9.4. The components that generate proactive suggestions and intelligent experiences call upon Google Android APIs that mediate access to certain hardware and software features in order to gather this physical and digital context; the same applies for the surfaces where the experience is then provided.[931] Some of these APIs are system APIs, i.e. restricted to components that have privileged permissions, which are available to Alphabet's ASI component, as it is granted such privilege, but not to downloaded apps (see Section 5.2).[932]

(772) Certain experiences, in particular smart replies and proactive suggestions with information, generate content that is destined to be transmitted to an app, service, or the OS, mediated by the OS. For example, if a friend asks in a chat for the name of a recently discussed restaurant, the user could be shown a suggested reply with the name of that restaurant, intended to be inserted into the text field of the chat app. The surfacing app does not immediately receive or see the contents of the suggestion, especially when it is based on a user's personal data.[933] Instead, after having reviewed the suggestion, the user must interact with it and "accept" it, for example by tapping the surface.[934] Only at that point, the suggestion will "egress" or be transmitted to the app, e.g. copied to the text field.[935] Similarly, a proactive suggestion can offer an interaction with another app or the OS, for example via intents, to be executed upon the user's acceptance of the suggestion.[936] This requires access to apps' actions on-device, for example, the actions discoverable and executable via App Functions (Section 10.2).[937]

(773) Certain UI surfaces inside apps are managed via "delegated rendering", which allows that app to (optionally) indicate where a suggestion or experience is best positioned in relation to the app's own UI elements. [Alphabet's internal position regarding delegated rendering].[938] On Pixel devices, APIs for delegated rendering are

---

931 [Description of Alphabet's internal systems regarding Android APIs]. Alphabet's reply to RFI PO 9 of 4 March 2026, Document "Annex Q28", Doc ID […], second column submitted in Alphabet's reply to RFI 9 of 4 March 2026, Doc ID […]. [Description of Alphabet's internal systems regarding Android APIs]. Alphabet's reply to RFI PO 7 of 27 February 2026, Doc ID […], question 6.b, paragraph 36. Alphabet's reply to RFI PO 9 of 4 March 2026, Document "Annex Q28", Doc ID […], last column submitted in Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID […]. Alphabet's reply to RFI PO 15 of 27 March 2026, Doc ID […], question 7, paragraphs 29, 30.

932 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID […], paragraph 181.

933 Alphabet's reply to RFI PO 13 of 20 March 2026, Doc ID […], paragraph 8.

934 There may be experiences where the suggestion is not meant to egress to the app at all. For example, if a flight number is shown during a call with an airline in the dialer, this is meant for the user to read over the phone to the airline, but not to be provided to the dialer.

935 Alphabet's reply to RFI PO 7 of 27 February 2026, Doc ID […], question 6.d, paragraph 38, 40.

936 Alphabet internal document, Document [Alphabet internal document], Doc ID […], page 1.

937 Alphabet internal document, Presentation [Alphabet internal document], Doc ID […], row "Magic Cue & Nudges".

938 Alphabet's reply to RFI PO 13 of 20 March February 2026, Doc ID […], paragraphs 6, 7.

[Alphabet's internal position regarding delegated rendering] only available to Alphabet.[939]

(774) Next to the user's current context, broader context available on the device as data is highly important to provide relevant experiences and suggestions. For example, while the current screen context can provide information that the user is calling an airline, it is only possible to suggest the flight number of the user's upcoming flight through access to the on-device app data that contains that flight number (e.g. an email). To enable this, the apps that hold this relevant information share this data through an on-device database or data sharing platform allowing for centralised data access and sharing, mediated by the OS.

(775) The current implementation of such data revolves around AppSearch, as described in Section 9.2. The system intelligence role holder enjoys global access to apps' data stored on-device in AppSearch platform storage.[940] As opposed to specific data that apps explicitly share with an assistant (Section 9.2), this data access is much broader: all apps are opted *in* by default into sharing their data stored in AppSearch platform storage with this role.[941] Only Alphabet's ASI or OEMs' intelligence components may hold this role and thus obtain such permission and broad, global data access for providing intelligent experiences.[942] Apps do have means to opt out,[943] which Alphabet claims "anecdotally" to be common, despite saying it has no telemetry that would allow it to reliably make that claim.[944]

(776) Alphabet's internal documents show that [Alphabet's internal system processes and components],[945] [direct quote describing Alphabet's commercial and strategic discussions][946] [Alphabet's internal system processes and components.[947] [Alphabet's internal system processes and components].[948] [Alphabet's internal system processes and components],[949] [Alphabet's internal system processes and components];[950] [Alphabet's internal system processes and components].[951] Simultaneously, Alphabet envisages that [description of Alphabet's commercial and

939 Alphabet's reply to RFI PO 13 of 20 March 2026, Doc ID […], question 1.f, paragraph 9.

940 Through the *READ_GLOBAL_APP_SEARCH_DATA* permission.

941 Alphabet's reply to RFI PO 7 of 27 February 2026, Doc ID […], question 2.c, paragraph 14. See also TM 2 PPT, page 16.

942 [Alphabet's internal system processes and components]. Alphabet internal document, Document "Annex Q28", Doc ID […], third column submitted in Alphabet's reply to RFI 9 of 4 March 2026, Doc ID […]. RFI 7, Q11a, para 74b.

943 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID […], question 35, paragraph 168. RFI 7, Q1d, paragraph 4.

944 Alphabet's reply to RFI AI 7 of 27 February 2026, Doc ID [...], question 2.c, paragraph 14.

944 Alphabet's reply to RFI AI 7 of 27 February 2026, Doc ID [...], question 2.c, paragraph 14.

945 Alphabet internal document, Presentation [Alphabet internal document], Doc ID […], pages 3, 23. Alphabet internal document, Document [Alphabet internal document], Doc ID [...], page 1. Alphabet internal document, Document [Alphabet internal document], Doc ID [...], pages 3-6. Alphabet internal document, Document [Alphabet internal document], Doc ID [...], pages 2-3.

946 Alphabet internal document, Presentation [Alphabet internal document], Doc ID [...], page 26.

947 Alphabet internal document, Presentation [Alphabet internal document], Doc ID [...], page 27.

948 Alphabet internal document, Document [Alphabet internal document], Doc ID [...], page 4. Alphabet internal document, Document [Alphabet internal document] Doc ID [...], page 3. Alphabet internal document, [Alphabet internal document] email of 12 January 2026, Doc ID [...], page 7.

949 Relevant code names are *pContext* or *Sian* for the cached data and *Geller* for the synchronization mechanism. Alphabet internal document, Document [Alphabet internal document], Doc ID [...], page 1.

950 Alphabet internal document, Document [Alphabet internal document], Doc ID [...], page 1. Alphabet internal document, Presentation [Alphabet internal document], Doc ID [...], page 6. Alphabet internal document, Document [Alphabet internal document] of 21 August 2025, Doc ID [...], page 2.

951 Alphabet internal document, Document [Alphabet internal document], Doc ID [...], page 6.

strategic discussions],[952] [description of Alphabet's commercial and strategic discussions],[953] [description of Alphabet's commercial and strategic discussions].

9.3.3.3. Access to processing resources

(777) The processing logic that actually provides the intelligence for the intelligent experience or proactive suggestion relies on relevant machine learning models.[954] Given the need for a proactive experience with low latency, these models often run on-device, requiring access to device resources mediated by the OS. [Alphabet's internal system processes and components],[955] [Alphabet's internal system processes and components],[956] [Alphabet's internal system processes and components].[957] Magic Cue uses machine learning models in its "memory generation" process, i.e. to derive insights from raw data (e.g. emails) to later suggest and surface.[958] Some intelligent experiences have evolved to using the Gemini Nano on-device model, accessed via AICore,[959] which may already be present on the device and is accessed via the operating system (see Section 11.2). [Description of Alphabet's product development].[960]

(778) At the same time, when more computational power is needed, it may be necessary to use and communicate with cloud-based models. For example, some data processing for Magic Cue is conducted in the cloud via Private AI Compute.[961] This infrastructure uses a cloud-based "protected space", which provides certain security and privacy guarantees that are usually reserved to on-device processing, such as inaccessibility to Alphabet or any other party.[962]

(779) When processing of sensitive data such as ambient data, it may be desirable to offer strong guarantees on privacy and security, e.g. the adoption of process isolation, encryption, anonymous and aggregated telemetry, and user control and awareness, which may be enabled by the OS. The CDD requires that intelligence components process such sensitive data in a "protected execution environment".[963] Data is not allowed to leave such an environment (or the device) unless the user intends so, e.g. a proactive suggestion is transmitted to an app only when the user accepts it (e.g. the chat app receiving the proposed chat message reply), or if the data is processed with

---

952 Alphabet internal document, Document [Alphabet internal document], Doc ID [...], page 2. Alphabet internal document, Document [Alphabet internal document], Doc ID [...], pages 4, 5.

953 Alphabet internal document, Document [Alphabet internal document], Doc ID [...], page 3. Alphabet internal document, Document [Alphabet internal document], Doc ID [...], page 6.

954 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID [...], question 40, paragraph 185.

955 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID [...], question 40, paragraph 185.

956 Alphabet internal document, Document [Alphabet internal document], Doc ID [...], page 2. Alphabet internal document, Presentation [Alphabet internal document], Doc ID [...], page 4. Alphabet internal document, Presentation [Alphabet internal document], Doc ID […], page 28.

957 Alphabet internal document, Document [Alphabet internal document], Doc ID […], page 2.

958 Alphabet's reply to RFI PO 7 of 27 February 2026, Doc ID [...], question 11.b, paragraphs 70.b, 76.

959 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID [...], paragraphs 185, 189, 193.

960 Alphabet internal document, Presentation [Alphabet internal document], Doc ID [...], pages 3, 6, 27; Alphabet internal document, Presentation [Alphabet internal document], Doc ID [...], page 6.

961 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID [...], question 41, paragraph 187. See https://blog.google/innovation-and-ai/products/google-private-ai-compute/, accessed on 15 April 2026, Doc ID [...].

962 See https://blog.google/innovation-and-ai/products/google-private-ai-compute/, accessed on 15 April 2026, Doc ID [...].

963 See https://source.android.com/docs/compatibility/16/android-16-cdd, accessed on 20 April 2026, Doc ID [...]. In particular, Sections 9.8.6 (OS-level and ambient data) and 9.8.15 (Sandboxed API Implementations) are relevant, though other relevant requirements appear throughout, being Sections 9.1 [C-4-1]; 9.8.12. Music Recognition (for Now Playing); 9.8.16. Continuous Audio and Camera data.

a privacy-preserving mechanism.[964] The CDD further requires user affordances and opt-outs, and imposes limits on network access for intelligence components.[965]

(780) For Alphabet's intelligence components, ASI and PSI, these requirements are met by processing data in an environment that complies with its "Private Compute Core" (PCC) policies.[966] While PCC is not embodied by a single set of APIs, the policies are supported by a variety of infrastructural components, including sandboxing and isolated processes (containment of data such that is inaccessible to other processes), controlled data flows notably through Private Compute Services (PCS),[967] and processing user data in the cloud via Private Inference / Private AI Compute.[968] For ACE, [description of Alphabet's product development].[969] [Description of Alphabet's commercial and strategic discussions].[970] [Confidential commercial agreements].[971] [Direct quote describing Alphabet's commercial and strategic discussions],[972] reflecting that intelligence experiences (based on ambient data) are highly relevant for AI services.

*9.3.4. Feature falls within the scope of Article 6(7) DMA*

(781) The Commission considers that the context-aware intelligence feature as described in Section 9.3.2 – together with its functionalities – is controlled via Google Android and falls within the scope of Article 6(7) DMA.

*9.3.5. Current access for Alphabet's own services and hardware*

(782) The context-aware intelligence feature is available to and used by Alphabet.

9.3.5.1. Access to the context-aware intelligence feature via ASI and PSI

(783) Alphabet's services benefit from, and therefore use, the context-aware intelligence feature through Android System Intelligence (ASI) and Pixel System Intelligence (PSI). As will be explained below, the access to inputs and context, processing, and output surfaces, enable Alphabet's services to offer the end user relevant contextual experiences. As a simple example, Gmail can proactively suggest the relevant flight number when calling an airline, due to access to screen context (that makes apparent that the end user is calling an airline) and an on-screen output surface (e.g. in the Dialer app), making Gmail more attractive to the end user.

(784) Alphabet's services generally do not directly call the APIs that represent the context-aware intelligence feature, and instead ASI and PSI do so. However, this direct calling of APIs, or any type of direct integration with the feature for that matter, is not the relevant test to determine whether a feature is used by or available to a service (Section 4.2). ASI and PSI are not services (nor features) within the meaning

964 Technical meeting Alphabet/Commission of 19 February 2026, Doc ID [...], slide deck page 25.

965 See https://source.android.com/docs/compatibility/16/android-16-cdd, accessed on 20 April 2026, Doc ID [...], Sections 9.8.6 and 9.8.16.

966 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID [...], paragraphs 179, 180, 183.

967 See Marchiori et al., "Private Compute Core Architecture", Google, 23 September 2022, https://arxiv.org/pdf/2209.10317v2, Doc ID [...], Sections 2.2.2, 2.3.2 and 3.

968 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID [...], paragraph 187.

969 Alphabet internal document, Presentation [Alphabet internal document] Presentation of December 2025, Doc ID [...], pages 12, 14. Alphabet internal document, Presentation [Alphabet internal document], Doc ID [...], page 63.

970 Alphabet internal document, Presentation [Alphabet internal document] Presentation of December 2025, Doc ID [...], page 19.

971 Alphabet internal document, Document [Alphabet internal document], Doc ID [...], page 78. Alphabet internal document, Document [Alphabet internal document], Doc ID [...], pages 33-35.

972 Alphabet internal document, Document [Alphabet internal document], Doc ID [...], page 79.

of the DMA, but instead software components.[973] ASI and PSI represent an implementation detail, where these components facilitate how Alphabet's services use the context-aware intelligence feature, by handling the access to the various functionalities of that feature on behalf of the services. This reflects an architecture where responsibilities are divided between software components:[974] rather than making every Alphabet service individually integrate the logic to generate proactive suggestions or intelligent experiences and therefore require each service to individually access the feature. Alphabet abstracts this logic and access away into the ASI and PSI components, which all of its services then integrate with instead.

(785) Which component is concretely responsible for the direct access to a feature, and where those components are then concretely incorporated (in the service's app, in a middleware app package, or in the operating system image), does not affect the delineation of the operating system nor the identification of a feature or of a service.[975] In theory, the Gmail app could have directly incorporated the logic for accessing and processing inputs to display, for example, its suggested flight number on relevant output surfaces. The choice to abstract this away into the ASI/PSI components that are reusable across Alphabet's apps is mainly one of efficiency.[976] This does not alter how that app's service may use and benefit from the context-aware intelligence feature, except that there is a level of indirection. By analogy, every single app could incorporate its own on-device model, such as Gemini Nano, and directly use the on-device model implementation feature (Section 11.3), but it is more efficient to install it centrally and make it accessible to services, who then still indirectly use that on-device model implementation feature.

9.3.5.2. The use of the context-aware intelligence feature by Alphabet's services

(786) Alphabet's ecosystem of services benefits from, and therefore use, the context-aware intelligence feature in many ways.

(787) Alphabet's services, including Google first-party apps, integrate with, benefit from, and thus use the context-aware intelligence feature through intelligent experiences provided by the Android System Intelligence component, on Pixel devices and on devices from other OEMs. Examples of such experiences, and the functionalities of the context-aware intelligence feature that they use, include, but are not limited to:

(a) Alphabet apps that output audio and video, such as YouTube, integrate with Live Caption, which provides captions for the current system audio. [Alphabet's internal system processes and components].[977]

---

973 See also Alphabet's response to the PFs, Doc ID [...], paragraph 162-163, which is falsely premised on the presumption that ASI and PSI must be classified monolithically as either a service or as part of Google Android.

974 This is further shown by [Alphabet's internal system processes and components]. Theoretically, nothing would prevent ASI from taking care of the same tasks – such as generating proactive suggestions – as PSI.

975 This applies just as much to the Gemini service, whose boundary is not limited to its 'app' (or APK package for that matter), despite Alphabet's assertion to that effect in Alphabet's response to the PFs, Doc ID [...], paragraph 254, [Alphabet's internal system processes and components]. See also Section 4.2.

976 Android's current requirements on the use of the context-aware intelligence feature, such as system privileges and role assignment, as explained in Section 9.3.3, would complicate the ability for the Gmail app to directly access the feature. However, as explained in Section 4.4, Alphabet controls these Android requirements and could adapt them if needed.

977 Alphabet internal document, Document "Annex Q28", Doc ID [...], row "Live caption" submitted in Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID [...].

(b) Alphabet apps, such as Google Messages, integrate with Live Translate, which translate text and audio. [Alphabet's internal system processes and components].[978]

(c) Alphabet apps, such as Google Messages, integrate with Smart Reply, i.e. keyboard suggestions,[979] which generates suggested replies in a conversation. [Alphabet's internal system processes and components];[980] [Alphabet's internal system processes and components];[981] [Alphabet's internal system processes and components];[982] [Alphabet's internal system processes and components].[983]

(d) Alphabet apps, such as Digital Wellbeing, integrate with Cough and snore detection, which detects coughing and snoring while the end user is sleeping. [Alphabet's internal system processes and components].[984]

(788) Alphabet's services, including Google first-party apps and Gemini, integrate with, benefit from, and thus use the context-aware intelligence feature through proactive suggestions on Pixel devices, marketed as Magic Cue and provided by the Pixel System Intelligence component. Alphabet's first-party services that enjoy such a benefit include Dialer/Phone, Messages, Keep, Contacts, Calendar, Gmail, Tasks, and Gemini.[985] Alphabet is considering adding support for Wallet and Tasks.[986] Alphabet's ecosystem of services benefits from Magic Cue, by surfacing suggestions inside its apps, which encourages end users to use those services to see these suggestions, and/or by providing data from its services for suggestions, which encourages users to use those services (e.g. as their email provider) such that richer suggestions can be generated.

(789) Alphabet's services' use of the context-aware intelligence feature is enabled by Pixel System Intelligence's and Magic Cue's access to the functionalities of the context-aware intelligence feature as follows:

---

978 Alphabet internal document, Document "Annex Q28", Doc ID [...], row "Live translate" submitted in Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID [...].

979 Smart Reply is powered by Android System Intelligence. Alphabet's reply to RFI 9 of 4 March 2026, Doc ID [...], question 25, paragraph 59.a. Smart Reply and keyboard suggestions appear to be considered interchangeable terms, see Alphabet internal document, Presentation [Alphabet internal document], Doc ID [...], page 49.

980 Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID [...], question 25, paragraph 59.a; Alphabet internal document, Document [Alphabet internal document], Doc ID [...], row "Keyboard suggestions" submitted in Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID [...].

981 Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID [...], question 25, paragraphs 59.d and 59.e.

982 Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID [...], question 25, paragraph 59.c.

983 Alphabet internal document, Document [Alphabet internal document], Doc ID [...], row "Keyboard suggestions" submitted in Alphabet's reply to RFI 9 of 4 March 2026, Doc ID [...].

984 Alphabet internal document, Document [Alphabet internal document], Doc ID [...], row "Cough & snore" submitted in Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID [...].

985 Alphabet's reply to RFI PO 7 of 27 February 2026, Doc ID [...], question 10, paragraph 66.b, 68.b, and question 11.a, paragraph 74.a. https://9to5google.com/2026/01/19/pixel-10s-magic-cue-feature-seems-to-be-preparing-google-tasks-wallet-integration/, accessed on 17 April 2026, Doc ID [...]. https://blog.google/products-and-platforms/devices/pixel/march-2026-pixel-drop/#footnote-5, Doc ID [...], accessed on 17 April 2026.

986 See https://9to5google.com/2026/01/19/pixel-10s-magic-cue-feature-seems-to-be-preparing-google-tasks-wallet-integration/, Doc ID [...], accessed on 17 April 2026.

(a) Magic Cue accesses the screen content (including of Alphabet's first-party apps) on a continuous basis and in the background to start the generation of a proactive suggestion and observe the current end user activity on-screen.[987]

(b) Magic Cue accesses apps' data stored on-device in AppSearch to gather the context. [Alphabet's internal system processes and components][988]), [Alphabet's internal system processes and components][989]), [Alphabet's internal system processes and components].[990] [Alphabet's internal system processes and components].[991]

(c) Magic Cue uses processing resources, including [Alphabet's internal system processes and components].[992]

(d) Magic Cue uses output channels and display surfaces (including in Alphabet's first-party apps), [Alphabet's internal system processes and components].[993]

(790) Gemini integrates with and benefits from the context-aware intelligence feature through Magic Cue. When a user asks a friend via a message about restaurant ideas, Magic Cue generates a relevant prompt and proactively suggests to the user to "search restaurants in Gemini" (without showing the prompt to the user).[994] If the user accepts the suggestion, Gemini opens in an overlay, prepopulated with the generated prompt.[995] This experience is tied to Gemini being set as the default assistant.[996] According to internal documents, [Alphabet forward-looking strategy].[997]

(791) Alphabet's characterization of this integration functioning [direct quote describing Alphabet's commercial discussions][998] glosses over the use of multiple functionalities of the context-aware intelligence feature to achieve this, including the access to screen content (which ultimately leads to the use of Gemini), the transmission of data to Gemini, and the interaction of opening Gemini in an overlay. By virtue of Alphabet's access to the context-aware intelligence feature, the end user is encouraged to use Gemini.

---

987 Alphabet's reply to RFI PO 7 of 27 February 2026, Doc ID [...], question 11, paragraph 70.a, question 11.a, paragraph 72.a, question 11.c, paragraph 78, question 14, paragraph 84. Alphabet internal document, Document "Annex Q28", Doc ID [...], row […] submitted in Alphabet's reply to RFI 9 of 4 March 2026, Doc ID [...].

988 Alphabet's reply to RFI PO 13 of 20 March 2026, Doc ID [...], question 1l, paragraph 16.

989 Alphabet's reply to RFI AI 7 of 27 February 2026, Doc ID [...], question 11.b, paragraph 75.a.

990 Alphabet's reply to RFI PO 7 of 27 February 2026, Doc ID [...], question 10, paragraph 66.b, 68.b, and question 11.a, paragraph 74.a.

991 Alphabet's reply to RFI PO 18 of 10 April 2026, Doc ID [...], question 7a and 7b, paragraph 31, 32.

992 Alphabet internal document, Document "Annex Q28", Doc ID [...], row […] submitted in Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID [...]. Alphabet's reply to AI RFI 7 of 27 February 2026, Doc ID [...], question 11.b, paragraph 76, question 14, paragraph 84, and question 20, paragraph 112.

993 Alphabet internal document, Document "Annex Q28", Doc ID [...], row […] submitted in Alphabet's reply to RFI 9 of 4 March 2026, Doc ID [...].

994 See Levin, M. (2026), 'March Pixel Drop: New personalization and AI tools', Google website, 3 March, https://blog.google/products-and-platforms/devices/pixel/march-2026-pixel-drop/, Doc ID [...], accessed on 17 April 2026, footnote 5.

995 See https://blog.google/products-and-platforms/devices/pixel/march-2026-pixel-drop/#footnote-5, Doc ID [...], accessed on 17 April 2026.

996 See Levin, M. (2026), 'March Pixel Drop: New personalization and AI tools', Google website, 3 March, https://blog.google/products-and-platforms/devices/pixel/march-2026-pixel-drop/, Doc ID [...], accessed on 17 April 2026, footnote 5.

997 Alphabet internal document, Presentation [Alphabet internal document], of August 2025, Doc ID [...], page 18.

998 Alphabet's reply to RFI PO 7 of 27 February 2026, Doc ID [...], question 9, paragraph 62.

(792) Internal documents show that Alphabet envisages [Alphabet forward-looking strategy].[999] Alphabet is also developing [Alphabet forward-looking strategy].[1000] Alphabet considers [Alphabet forward-looking strategy].[1001] Alphabet also has plans [Alphabet forward-looking strategy],[1002] [Alphabet forward-looking strategy],[1003] [Alphabet forward-looking strategy].[1004] An analysis of a beta version of the Dialer/Phone app also suggested a potential rebranding of Magic Cue to "Proactive Assistance", which would be presented as part of Gemini: [direct quote describing Alphabet's commercial and strategic discussions].[1005]

(793) In addition, Alphabet's first-party apps integrate with and benefit from the context-aware intelligence feature through proactive suggestions on devices from other OEMs, through their respective intelligence components. For example, one type of proactive suggestion in Samsung's Now Nudge lets end users open a suggested location in Google Maps.[1006] [Alphabet's internal system processes and components].[1007]

(794) Therefore, the Commission considers that the context-aware intelligence feature is available to and used by Alphabet's services within the meaning of Article 6(7) DMA, in particular as accessed through Android System Intelligence (ASI) and Pixel System Intelligence (PSI).

*9.3.6. Current access for third-party services and hardware*

(795) Third parties do not enjoy equally effective interoperability with the context-aware intelligence feature.

9.3.6.1. Access to inputs

(796) *First,* third-party apps, including user-downloaded apps, have no equal access to the user's physical context, i.e. ambient data. As explained further in Section 9.4.6, intelligence components have differentiated access to ambient data.[1008] Indeed, Alphabet imposes certain conditions that do not apply to the Alphabet's own services, which renders interoperability ineffective.[1009]

(797) *Second,* third-party apps, including user-downloaded apps, have no equal access to the user's digital context. Alphabet submits that downloaded apps can access "the same" screen content data through public APIs, for example the Media Projection API.[1010] However, certain inputs, such as ambient access to screen content (Section

999 Alphabet internal document, Document [Alphabet internal document], Doc ID [...], page 28.
1000 Alphabet internal document, Document [Alphabet internal document], Doc ID [...], pages 2 and 3.
1001 Alphabet internal document, Document [Alphabet internal document], Doc ID [...], page 1.
1002 Alphabet internal document, Document [Alphabet internal document], Doc ID [...], page 6. Alphabet's internal document, Presentation [Alphabet internal document], Doc ID [...], page 9.
1003 Alphabet internal document, Document [Alphabet internal document], Doc ID [...], pages 2 and 3.
1004 Alphabet internal document, Presentation [Alphabet internal document], Doc ID [...], page 25. Alphabet internal document, Document [Alphabet internal document], Doc ID [...], page 5, where Gemini is an agent.
1005 https://9to5google.com/2026/07/09/pixel-magic-cue-gemini/, Doc ID [...], accessed 13 July 2026.
1006 See https://www.samsung.com/lb/support/mobile-devices/how-to-use-the-now-nudge-feature-on-the-samsung-galaxy-s26-series/, accessed 22 April 2026, Doc ID [...].
1007 Alphabet internal document, Document [Alphabet internal document], Doc ID [...], page 1. Alphabet internal document, Presentation [Alphabet internal document], Doc ID [...], pages 16, 32.
1008 Alphabet internal document, Presentation [Alphabet internal document], Doc ID [...], page 36.
1009 Cf. paragraph (129).
1010 Technical meeting Alphabet/Commission of 19 February 2026, Doc ID [...], slide deck pages 23 and 24.

9.3.3.2), are exclusively available to an intelligence component.[1011] This also applies e.g. to speaker audio for voice call, which third parties cannot access.[1012]

(798) *Third,* third-party apps, including user-downloaded apps, do not have access to all ("global") data donated by other apps via AppSearch, as the required permission[1013] is only available to pre-installed intelligence components, which includes Alphabet's ASI. In addition, this permission being role-specific entails that it may be granted to and used only by the role holder at a given time.

(799) This limitation also includes a lack of access to data donated by Alphabet's first-party apps, which is accessed by Magic Cue. In ACE, access to apps' data would continue to be limited to pre-installed apps. In addition, data from Alphabet's first-party apps would be available to intelligence components, only if Gemini is set as the default assistant on the device, further limiting access for third parties to apps' data. [Alphabet internal document describing a third party's interactions with Alphabet regarding this limitation].[1014]

(800) Finally, third-party apps, including user-downloaded apps, have no effective access to discover and execute available actions inside apps, as developed in Section 10.2.

(801) Therefore, third-party apps do not have equal access to inputs.

9.3.6.2. Access to processing resources

(802) *First,* third-party apps, including user-downloaded apps, do not have equal and full access to on-device models, as described further in Section 11.2 and Section 11.3.

(803) *Second,* third-party apps, including user-downloaded apps, need equal access via communication channels to the cloud-based models used by that third party. Such access is important to provide as rich an offering as possible, as on-device models might be too limited in some cases.[1015] This includes the ability to process sensitive data such as ambient data in the cloud, as Alphabet does for Magic Cue in its Private AI Compute environment, which acts as a protected execution environment.[1016]

(804) *Third,* third-party apps, including user-downloaded apps, are unable to use isolated processes and protected execution environments to process data. This includes an inability to use the PCC framework, as it is only available to Alphabet's services.[1017] Alphabet submits that third parties could work with OEMs to create protected execution environments.[1018] However, this essentially amounts to requiring pre-installation, which does not provide effective interoperability to third parties (Section 6.3.1).

(805) Therefore, third-party apps do not have equal access to processing resources.

---

1011 Alphabet internal document, Document [Alphabet internal document], Doc ID [...], page 2.

1012 This requires the *CAPTURE_AUDIO_OUTPUT* permission, which is only available to privileged system apps. See https://developer.android.com/media/platform/sharing-audio-input, accessed on 15 April 2026, Doc ID [...], Section "Voice call + ordinary app". Email from [AI service provider], sent 23 March 2026, Doc ID [...], page 2. Agreed minutes of meeting with [AI service provider] of 30 March 2026, Doc ID [...], paragraphs 11, 12. Agreed minutes of meeting with [AI service provider] of 2 April 2026, Doc ID [...], paragraph 10.

1013 The *READ_GLOBAL_APP_SEARCH_DATA* permission.

1014 Alphabet internal document, Document [Alphabet internal document], Doc ID [...], page 3.

1015 Agreed minutes of meeting with [AI service provider] of 20 March 2026, Doc ID [...], paragraph 11.

1016 As required by the CDD, Section 9.8.6, [C-1-3], Android 16 Compatibility Definition, https://source.android.com/docs/compatibility/16/android-16-cdd, Doc ID [...].

1017 Technical meeting Alphabet/Commission of 19 February 2026, Doc ID [...], slide deck page 26. Alphabet's reply to RFI AI 6 of 23 February 2026, Doc ID [...], question 39, paragraph 183.

1018 Alphabet's reply to RFI AI 6 of 23 February 2026, Doc ID [...], question 39, paragraph 184.

9.3.6.3. Access to output channels and display surfaces

(806) *First,* third-party apps, including user-downloaded apps, do not have equally effective access to output channels and display surfaces as Alphabet.[1019] Third parties therefore also are unable to discover, select, and use all channels and surfaces that are available to Alphabet. For some surfaces inside Alphabet apps, OEM apps, and third-party apps, Magic Cue uses delegated UIs, which are not available to third parties.[1020]

(807) *Second,* third-party apps, including user-downloaded apps, cannot transmit data or trigger integrations and interactions with apps and the operating system from the proactive suggestion or intelligent experience in an equally effective manner as Alphabet. Next to a lack of access to structured on-device integration (Section 10.2), third parties are unable to provide intelligence components, as this requires system privileges, and can therefore not freely generate proactive suggestions or intelligent experiences that interact with or transmit data to any third-party service (e.g. invoking a third-party assistant with a proactively suggested query).

(808) Therefore, third-party apps do not have equal access to output channels and display surfaces.

9.3.6.4. Permissions

(809) Third-party apps, including user-downloaded apps, are prevented from being the relevant system intelligence role holder, through which access to functionalities could alternatively be obtained.

(810) Alphabet requires OEMs to grant either ASI or an OEM alternative the relevant system intelligence role(s).[1021] ASI is also considered non-replaceable, including through a signature check that the binary has not been replaced.[1022]

(811) At the same time, the CDD (Sections 9.8.6 and 9.8.15) explicitly prevents OEMs from allowing users to grant user-downloaded apps access to any of the ambient data regulated by those Sections of the CDD, including Content Capture and global AppSearch.[1023] Alphabet's policies thereby require OEMs to deny user-downloaded apps access to such ambient data.

(812) [Description of Alphabet's product development].[1024]

(813) Therefore, third-party apps do not have equal access to the relevant permissions.

*9.3.7. Alphabet's response to the Preliminary Findings*

(814) Alphabet considers that the draft measures are unlawful because Alphabet already complies with Article 6(7) DMA.[1025] Alphabet states that the context-aware intelligence feature is available to third parties under the "same" condition of OEM

---

1019 Alphabet's reply to RFI PO 15 of 27 March 2026, Doc ID [...], question 7, paragraphs 29, 30, 31, 32. Agreed minutes of meeting with [AI service provider] of 10 February 2026, Doc ID [...], paragraph 13.

1020 Alphabet's reply to RFI AI 7 of 27 February 2026, Doc ID [...], question 6.c/d, paragraph 38.

1021 [Confidential commercial agreement], Alphabet internal document, Document "Annex Q13.2 – […] - EC AI RFI #2 of 23 September 2025 (8 October 2025) (Confidential - Contains Business Secrets).pdf", Doc ID [...], page 270 submitted in Alphabet's reply to RFI AI 2 of September 2025.

1022 See Marchiori et al., "Private Compute Core Architecture", Google, 23 September 2022, https://arxiv.org/pdf/2209.10317v2, Doc ID [...], Section 2.4.2.

1023 CDD, Section 9.8.6 [C-2-1], Android 16 Compatibility Definition, https://source.android.com/docs/compatibility/16/android-16-cdd, Doc ID [...].

1024 Alphabet's reply to RFI AI 7 of 27 February 2026, Doc ID [...], question 19.i, paragraph 107.

1025 Alphabet's response to the PFs, Doc ID [...], paragraph 242.

preinstallation and permission pre-granting,[1026] and that the feature is not available to its services, emphasising Gemini.[1027] In any event, Alphabet considers that it already allows effective interoperability,[1028] and outlined how it sees the functionalities already being available, naming various APIs that Alphabet considers to provide certain functionalities "under equal conditions".[1029] Alphabet also raises that delegated UIs are proprietary to Pixel and not mediated by Google Android.[1030] Alphabet raises two points on cloud-based models and protected execution environments, which are clarified in recital (854).

(815) Alphabet states that any further interoperability gap is justified as a strictly necessary and proportionate integrity measure.[1031] According to Alphabet, the feature's functionalities are "too sensitive" to make available to any "first-party or third-party apps",[1032] Alphabet raises various risks, such as device instability, deceptive overlays, and fraud.[1033]

(816) Finally, Alphabet considers that the draft measures are disproportionate, as they would threaten device integrity, offer uncertain benefits, only resolve an interoperability gap that is at most very small, and deprive OEMs of control.[1034]

*9.3.8. The Commission's assessment*

9.3.8.1. Concerning the existing level of interoperability and integrity

(817) **Interoperability gap.** Alphabet's explanations of how the functionalities of the feature are already available are unsatisfactory.[1035] If anything, they confirm the interoperability gap described in Section 9.3.6. Where Alphabet identifies that functionalities are only available to OEM-trusted components, this limits beneficiaries and thus fails to offer effective interoperability (Section 5.4.1). Where Alphabet identifies alternative APIs, they are either tied to the default assistant role, which limits beneficiaries and precludes multi-homing (Section 5.2.4), or are not equally effective, e.g. in terms of user journey. Where Alphabet states a delegated UI is proprietary to Pixel devices,[1036] a parallel implementation of a functionality that is reserved only to the gatekeeper would seriously undermine the effectiveness of the interoperability solution made available to third parties (Section 4.5.3).

(818) Alphabet raises that none of the functionalities that depend on the system intelligence role "*are available to the Gemini app*".[1037] Section 9.3.5 already addresses how Alphabet's control of feature access in Android makes the feature available to its services, and how Alphabet's services, including Gemini, benefit from and therefore use the feature, even if through ASI and PSI and not via direct integration with Gemini.[1038]

---

1026 Alphabet's response to the PFs, Doc ID [...], paragraph 246.
1027 Alphabet's response to the PFs, Doc ID [...], paragraphs 242, 254.
1028 Alphabet's response to the PFs, Doc ID [...], paragraph 242.
1029 Alphabet's response to the PFs, Doc ID [...], paragraphs 247-249, 252.
1030 Alphabet's response to the PFs, Doc ID [...], paragraph 253.
1031 Alphabet's response to the PFs, Doc ID [...], paragraphs 242, 255.
1032 Alphabet's response to the PFs, Doc ID [...], paragraph 246. Also, Alphabet's response to the PFs, Doc ID [...], paragraph 225, second bullet.
1033 Alphabet's response to the PFs, Doc ID [...], paragraphs 255-257.
1034 Alphabet's response to the PFs, Doc ID [...], paragraphs 242, 259.
1035 Alphabet's response to the PFs, Doc ID [...], paragraphs 247-249, 252.
1036 Alphabet's response to the PFs, Doc ID [...], paragraph 253.
1037 Alphabet's response to the PFs, Doc ID [...], paragraph 254.
1038 The Commission notes that the interoperability obligation under Article 6(7) DMA addresses features available to *services*. This does not provide for a limitation to the Gemini 'app' but applies to the

(819) Alphabet mentions that "*potential future plans around use of a feature*" should not trigger the interoperability obligation.[1039] The feature and all of its functionalities already exist today, which Alphabet does not contest. Future plans are merely discussed to highlight that Alphabet is working on the further integration of this – already existing – feature into its own services, such as Gemini. In that regard, these future plans only reaffirm the availability of said feature, which is precisely what enables Alphabet to conceive of the further integration. Limiting feature access based on the current use by, or on direct integration with, Alphabet's services would deny third parties the same opportunities to use the feature for offering innovative proactive suggestions and intelligent experiences that go beyond what Alphabet offers.

(820) **Integrity.** Alphabet alleges that the interoperability gap is a strictly necessary and proportionate integrity measure.[1040] Alphabet invokes the sensitivity of the feature and the existing restrictions on components accessing the feature, being sandbox isolation and strict on-device processing rules, which would warrant limiting feature access to OEM-trusted components. Alphabet raises the same concern for global access to apps' data and actions stored on-device.[1041]

(821) The Commission notes that Alphabet presents a variety of risks, some of which relate rather to user security or privacy instead of integrity, without detailing how they are uniquely enabled by the context-aware intelligence feature, nor proposing countermeasures other than blunt access restriction through OEM preinstallation. While the Commission does not contest that such risks are genuine, they are not specific to the context-aware intelligence feature (e.g. the "overlay-based attacks"[1042] raised relate to any type of overlay, including that of the default assistant), and OEM preinstallation (i) does not enable third parties to access the feature under the "same" condition,[1043] as discussed in Section 5.4.1, and (ii) cannot be construed as a strictly necessary and proportionate integrity measure,[1044] as discussed in Section 6.3.1.

(822) Alphabet's security expert contends that the context-aware intelligence feature enables these various risks by breaking Android's app isolation security model, where apps cannot interact with each other.[1045] This is imprecise, as the entire purpose of this feature is to enable cross-app interaction, which is incompatible with strict isolation. Alphabet has already decided, through their control of the security and privacy measures that they apply, that this feature may deviate from Android's standard security model of isolation, to provide a benefit to Alphabet's services. This deviation cannot be used as an excuse to withhold that same benefit from third-party services, as this would undermine the obligation of effective interoperability. Under Article 6(7) DMA, the gatekeeper is free to implement integrity measures such as requiring that only authorised cross-app access is possible.

(823) The measures annexed to this Decision give Alphabet the additional flexibility to implement security and privacy measures on feature access by third parties in line

---

Gemini service at large (Section 4.2). The restriction to the Gemini app would be particularly artificial due to the distribution of the Gemini service over the Google app and Gemini app (Section 5.3).

1039 Alphabet's response to the PFs, Doc ID [...], paragraph 254.

1040 Alphabet's response to the PFs, Doc ID [...], paragraph 255-257.

1041 Alphabet's response to the PFs, Doc ID [...], paragraph 225, second bullet.

1042 [Third party's] contribution to the Commission's public consultation, Doc ID [...], paragraph 54.

1043 Alphabet's response to the PFs, Doc ID [...], paragraph 246.

1044 Alphabet's response to the PFs, Doc ID [...], paragraph 255-257.

1045 Expert report of [professor], Annex 1 of Alphabet's response to the PFs, Doc ID [...], paragraphs 37 and 38.

with principles of *inter alia* process (i.e. sandbox) isolation, which addresses Alphabet's concern that functionalities should only be available to components that adhere to such principles,[1046] and aligns with recommendations for such principles from some third parties who contributed to the public consultation.[1047] In this context, Alphabet raises that it would be impossible to enforce the PCC framework for third parties.[1048] Insofar as this framework is currently in part policy-based, the inability to (fully) enforce it technically would not be a valid reason to withhold a feature from third parties, as it would be a condition that is impossible to meet for any third party, therefore not allowing effective interoperability.[1049] In any event, work is already ongoing to make PCC a technical solution, as described in recital (780); additional engineering work may be required from a gatekeeper to make a feature interoperable (Section 4.5.1).

(824) Requiring on-device processing from third parties is not strictly necessary and would be contrary to what Alphabet does for its own services. Alphabet itself introduced Private AI Compute for processing user data in the cloud in line with e.g. its isolation principles, allowing Magic Cue to generate proactive suggestions using more advanced reasoning and computational power than is available on-device.[1050] A strict prohibition of off-device processing would have prevented even Alphabet from offering such innovative solutions for better proactive suggestions and intelligent experiences that maintain or improve e.g. privacy guarantees. The measures on feature access in line with principles of *inter alia* process (i.e. sandbox) isolation, may apply also to third-party off-device processing environments.

(825) In any event, due to the sensitivity of this feature, this Decision explicitly allows Alphabet to limit access to the context-aware intelligence feature only to third parties that meet certain eligibility criteria, which include criteria relating to integrity, security, and privacy.

9.3.8.2. Concerning proportionality

(826) Alphabet argued in its response to the PFs that the measures for context-aware intelligence are disproportionate. However, Alphabet does not substantiate why the measures would be more restrictive than necessary to achieve the objectives of Article 6(7) DMA. In particular, Alphabet fails to propose any less restrictive, yet equally effective means of complying with that provision. Nor does Alphabet explain and substantiate why the feature and/or a specific measure imposes a disproportionate burden on Alphabet. Instead, its arguments focus on questioning the scope of Article 6(7) and the actual need for interoperability – a line of reasoning which does not meet the legal standard for a valid proportionality claim, as explained in Section 13 below.

---

1046 Alphabet's response to the PFs, Doc ID [...], paragraph 246.

1047 [OEM's] submission to the public consultation, Doc ID [...], page 4. [AI service provider's] submission to the public consultation, Doc ID [...], page 3. [Third party's] submission, Doc ID [...], page 2.

1048 Alphabet's response to the PFs, Doc ID [...], paragraph 251. Alphabet comments on the term 'protected execution environment', which is taken directly from its own slide deck presented at the Technical meeting Alphabet/Commission of 19 February 2026, Doc ID [...], slide deck page 25. Alphabet further characterises (and then disputes) protected execution environments as a feature. The measures do not position them as such: instead, access to protected execution environments is a functionality of the context-aware intelligence feature, see Section 9.3.2.

1049 See Section 4.8.4 read together with footnote 232.

1050 See https://blog.google/innovation-and-ai/products/google-private-ai-compute/, Doc ID [...], accessed on 15 April 2026.

(827) The Commission nonetheless addresses each of Alphabet's arguments in this respect.[1051]

(828) *First*, Alphabet's contention that broader access is inconsistent with protecting the integrity of the operating system is already rebutted above and not further substantiated. Access to this feature is necessary for third-party AI services to offer comparably rich experiences as Alphabet's own services. To the extent that opening of this feature to third parties would raise any integrity risks, Alphabet is free to implement integrity measures under the framework outlined in Section 4.8, which is compounded by the fact that this Decision explicitly allows Alphabet to limit access to the context-aware intelligence feature only to third parties that meet certain eligibility criteria, which include criteria relating to integrity, security, and privacy.

(829) *Second*, Alphabet's contention that the benefits of broader access are uncertain is contradicted by the clear interest, both present and future, by Alphabet itself for integrating the context-aware intelligence feature with its own services, as outlined in Section 9.3.5, and by the interest shown by third parties (Section 9.3.1). The Commission notes that the purported lack of mention by developers may well be the result of a lack of transparency and thus awareness of the existence of the feature, exacerbated by the lack of third-party access (Section 9.3.6), leading to developers so far having been unable to properly explore their use of this feature.

(830) Importantly, as explained above, this feature is necessary for third-party AI services to offer rich experiences going forward. The Commission also notes that the DMA does not foresee any assessment of impact or effects,[1052] nor is the scope of Article 6(7) DMA limited to features of importance.

(831) *Third*, Alphabet's contention that there is a negligible interoperability gap is rebutted in Section 9.3.6. In any event, the DMA is not limited to features with a "large" interoperability gap, nor does it provide for such an assessment. The measures are proportionate in addressing the lack of access, or access under unequal conditions, that result in ineffective interoperability.

(832) *Fourth*, Alphabet's contention that feature access would result in a loss of control for OEMs is addressed in Section 6.1.2. OEMs remain free and able to differentiate their devices, including through offering their own on-device intelligence solutions.

(833) The Commission therefore concludes that there is an interoperability gap with respect to the context-aware intelligence feature of Google Android. As a result, third parties and in particular user-downloaded apps, cannot perform important functionalities that are expected of an AI assistant which places them sub-par in comparison with Google's own AI-powered services. As a result, the Commission considers that it should specify how Alphabet must fill this gap to comply with Article 6(7) DMA.

*9.3.9. Measures that Alphabet should implement*

(834) To meet the requirements of Article 6(7) DMA, Alphabet should implement the measures set out in Section 2.2 of the Annex to this Decision. The interoperability solution for the context-aware intelligence feature, as described in Section 9.3.2 of this Decision, must be equally effective as that available to any of Alphabet's own services.

---

1051 Alphabet's response to the PFs, Doc ID [...], paragraph 259.

1052 Recital 11 DMA.

(835) The purpose of the measures is to grant third parties access to, and effective interoperability with, the context-aware intelligence feature and its functionalities, to develop their own alternative components for generating proactive suggestions and intelligent experiences that can run in parallel to components offered by Alphabet or OEMs, and that have effective access to the same inputs, processing resources, and outputs. These components can then integrate with third-party services to enable those services to benefit from proactive suggestions and intelligent experiences. For example, a third-party downloaded app should be able to show a proactive suggestion on the same surface (e.g. keyboard) based on the same screen context (e.g. chat messages) as Alphabet's services; likewise, a third-party downloaded app should be able to show live captions on the same surfaces (including in other apps) based on the same system audio access.

(836) These alternative solutions for proactive suggestions and intelligent experiences should equally be considered as separate components that are reusable and can integrate with various third-party services, similar to the architecture of Android System Intelligence and Pixel System Intelligence (Section 9.3.5.1).[1053] This enables the benefits of proactive suggestions and intelligent experiences within and across the ecosystems of third parties. For example, a third-party email service may want to integrate with, benefit from, and thus use the context-aware intelligence feature through proactive suggestions. The service could theoretically develop and integrate its own intelligence component that uses the information in the end user's email to show proactive suggestions within other apps (relying on e.g. access to screen content). Alternatively, it may be more efficient for another third party to offer a generic intelligence solution, with which the email service could integrate, e.g. by contributing its data. As another example, a third-party assistant could create and integrate its own component to suggest queries for that assistant across apps (and possibly allows others to integrate with that component) or integrate with another existing component. Alphabet remains free to determine the exact implementation architecture, as long as it provides effective interoperability.

(837) Alphabet was given the opportunity to comment on the measures (Section 2.1). The Commission has updated and clarified the measures based on Alphabet's comments.

9.3.9.1. The scope of services

(838) Article 6(7) DMA requires that a gatekeeper allows effective interoperability with the same features as available to or used by the services of the gatekeeper. Throughout the proceedings, Alphabet has made various proposals to limit the scope of the services captured by Article 6(7) DMA. As a preliminary remark, Article 6(7) DMA does not provide for any limits on that scope of services (Section 4.7 and Section 12.4). In any event, as explained in Section 9.3.5, a variety of Alphabet services benefit from, and therefore use, the context-aware intelligence feature.

(839) While the Commission has clarified to Alphabet that the scope should be all of Alphabet's services, the Commission considers it useful to clarify why Alphabet's proposals cannot be taken on board. This is in particular because it is unclear what the intent of those proposals is, and because Alphabet's proposed carveouts may undermine the objective of the measures. If the proposal is to be read under the Commission's interpretation set out in this Section 9.3, it does not introduce any change to the measures. However, under an alternative interpretation, assuming that

[1053] In certain cases, the components may still be offered standalone and/or directly integrated with the service, in support of the provider's own service. For example, translation services could integrate an intelligence component that offers live translation using the context-aware intelligence feature.

the proposal's aim is to cause the measures to change, it would change the measures and reduce the feature's scope in such a significant way that the measures will not achieve their objective of enabling third parties to effectively offer proactive suggestions or intelligent experiences.

(a) Alphabet suggested that the scope should be limited to Gemini or similar Alphabet's AI services.[1054] Under the Commission's interpretation, Gemini already benefits from and uses the feature (Section 9.3.5.2), so the feature would not change. Alternatively, in an interpretation where only APIs directly called by Gemini would be in scope (which is flawed, see Section 4.2), several functionalities would be excluded or degraded, since Gemini does not directly call the relevant APIs (Section 9.3.5.1).

(b) Alphabet later accepted that the scope should be all of Alphabet's services, but sought a "clarification" that "*the scope of "Alphabet's services" excludes system intelligence components that are deeply integrated with the OS.*"[1055] Under the Commission's interpretation, Alphabet's intelligence components (ASI and PSI) are not themselves gatekeeper services in the meaning of Article 6(7) DMA. However, Alphabet's services still use the (Section 9.3.5.2). Alternatively, in an interpretation which excludes those functionalities that Alphabet lets only be accessed by system-privileged intelligence components (Section 9.3.3), third-party apps, including user-downloaded apps, would not be able to access those functionalities.

(840) For completeness, Alphabet also suggested early on to limit the beneficiaries to OEMs.[1056] Article 6(7) DMA does not provide for such a limit to beneficiaries (Section 4.7 and Section 12.4). In any event, OEMs can already access the feature by preinstalling their own intelligence component (Section 9.3.3.1), so this would not make the feature available to any additional beneficiaries (in particular not to third-party apps, including user-downloaded apps), stripping the measures of useful effect.

9.3.9.2. Access to apps

(841) The functionalities of the context-aware intelligence feature that relate to access to apps (i.e. their digital context, including on-screen content and on-device app data, or surfaces in/on apps), must be provided to components offering proactive suggestions and intelligent experiences across first-party apps, OEM apps, and third-party apps, on a non-discriminatory basis between Alphabet's own services and third-party services within the scope of the context-aware intelligence feature.

(842) The context-aware intelligence feature follows an *app caller agnostic* model. That is, the participating app cannot selectively decide that only specific components can access functionalities of the OS-mediated feature when they relate to that app. Within Magic Cue and Private Compute Core, the participating app is even unaware altogether that e.g. its screen context was accessed or that a suggestion was rendered on top of the app.[1057] The app caller agnostic model constitutes a good foundation for equally effective interoperability: if a third-party component is not able to integrate with certain apps, it cannot fully provide the end user with an attractive offering that

---

1054 Alphabet submission of 16 June "Draft measure – Priority Issues", Doc ID [...], row Context-aware intelligence – Scope of features / functionality subject to interoperability.

1055 Alphabet submission "mark-up of the context aware intelligence measure", Doc ID [...], footnote 1. Alphabet submission "Scope of Alphabet's services", Doc ID [...], row Context-aware intelligence.

1056 Alphabet submission "Alphabet Second Mark-up of the draft measure" of 29 May 2026, Doc ID [...], pages 15-17.

1057 Alphabet's reply to RFI PO 13 of 20 March 2026, Doc ID [...], paragraphs 6-8.

works across all their preferred services. Moreover, such selective integrations would likely require bilateral negotiations between services, where the gatekeeper would have the upper hand in convincing apps to integrate with its proactive suggestions or intelligent experiences and therefore interfering with the contestability of third parties (see also Section 5.4).[1058]

(843) This also resembles how default assistants can already access apps today. A user can invoke the default assistant while in any app. That assistant can then read the screen content and put an overlay on the screen of that app, similar to the functionalities of access to that app's digital context and surfacing on that app, respectively. Assistants also have access to apps' data stored on-device (Section 9.2). Meanwhile, the app is unaware of which assistant is invoked on top of them, and therefore unaware of which assistant receives the screen content or overlays content. From the app's perspective, it is interacting with the OS, not another app. Alphabet therefore already accepts app caller agnosticism for functionalities similar to those of the context-aware intelligence feature.[1059]

(844) However, apps should retain the ability to control whether they participate in proactive suggestions or intelligent experiences, aligning with concerns about a loss of control (i.e. forced participation) raised by Alphabet and some third parties that contributed to the public consultation.[1060] The measures explicitly allow all apps (including Alphabet's first-party apps) to opt out of the ability for components offering proactive suggestions and intelligent experiences to access that app's digital context and/or to appear in a surface in/on that app, either for the whole app or for specific parts of the app, where that opt-out applies to all such components on a non-discriminatory basis between Alphabet's own services and third-party services. For example, apps could mark certain screens (or parts thereof) as sensitive and prevent any component from reading the content of those screens (e.g. screens with financial data) or from overlaying any content on those screens (e.g. password entry screens). Android already provides such abilities today, letting apps hide certain areas from being accessed, by setting the *FLAG_SECURE* flag, and prohibit overlays, by setting the *HIDE_OVERLAY_WINDOWS* flag.[1061]

---

1058 Nevertheless, while participation by apps in the context-aware intelligence feature of Google Android, as described in Section 9.3.3, should follow the app caller agnostic model, nothing precludes a third party from having bilateral direct integrations with other apps for the same purpose. This already applies in similar contexts today: apps could adopt structured on-device integration to make an integration available to all other apps, mediated via the OS (Section 10.2) or allow for bilateral direct integration with a specific other app using e.g. the Model Context Protocol, without OS mediation.

1059 An assistant could therefore offer a very crude form of e.g. suggestions already today, by reading what is on screen (e.g. a chat conversation), combining it with other contextual data it has (e.g. AppSearch data), and processing it to generate a suggestion that is overlaid on screen (e.g. a suggested reply). However, this would not be proactive or autonomous in any way, as the user would have to manually trigger the assistant and later copy the suggestion back into the app.

1060 Alphabet submission of 16 June 2026, "Draft measure – Priority Issues", Doc ID [...], Context aware intelligence - Ability for apps to control who receives their data and who can surface suggestions over them. Alphabet submission "mark-up of the context aware intelligence measure", Doc ID [...], paragraph 34d/e/h. [Third party's] submission to the public consultation, Doc ID [...], page 1, on disintermediation. [Third party's] submission to the public consultation, Doc ID [...], page 4.

1061 Google for Developers, 'Secure sensitive activities | Fraud prevention', Android Developers website, 6 March 2026, https://developer.android.com/security/fraud-prevention/activities, Doc ID [...].

(845) In the same vein, the measures on data provided ad hoc or global access to apps' data and actions stored on-device do not constitute forcing apps to donate any data, contrary to what Alphabet claims.[1062]

(a) For data provided ad hoc (e.g. metadata, relevant for intelligent experiences, that is not visible on screen, such as the current phone number called), apps must actively choose on the fly what to donate; no data would be available in a way that would remove the participating app's control over this data.

(b) For apps' data stored on-device, this only covers data that apps choose to donate voluntarily through an on-device database or data sharing platforms, where it is explicitly known to the developer that they are making this data available to intelligence components. As explained in Section 9.2.3 and 9.3.3.2, this currently corresponds to sharing data via AppSearch platform storage, for which the developer documentation explains that "*data that is indexed* [i.e. shared] *can be displayed on System UI surfaces*" subject to an opt-out.[1063] To be more exact, the data is accessible to the component that has the system intelligence role (Section 9.3.3.1). As such, Android System Intelligence receives the relevant permission,[1064] and thus can access apps' data stored on-device for any proactive suggestion or intelligent experience,[1065] to the potential benefit of Alphabet's services. Third parties must therefore be provided with such global access to apps' data stored on-device, as voluntarily donated by developers for use by intelligence components, given that this access to data is used by or available to Alphabet's services via ASI (Section 9.3.5). However, Alphabet is free to adapt the concrete implementation, as long as it results in access that is equally effective as available to Alphabet's services, and does not unfairly bias developers' choices towards Alphabet's ecosystem. For example, such a bias could be to require apps to opt in to specific intelligence components, as there is a risk that third parties would then (only) integrate with Alphabet's component, given that it already exists and may more easily enjoy e.g. preinstallation on OEM devices (Section 9.3.3.1).

(846) As such, this voluntary choice to donate data also extends to Alphabet's first-party apps. They are not required to make any specific data available for use by intelligence components. However, [details of how Alphabet's first party apps share data on device to Alphabet's intelligence components]. The measures specify that Alphabet must enable access for third parties to such apps' data stored on-device by Alphabet's first-party apps. Indeed, that data is available to Alphabet's services, given that it is accessed by Alphabet's intelligence components (per Section 9.3.5.1). Even if Alphabet stores that data in a way that currently makes it available to (and

---

1062 Alphabet submission of 16 June 2026, "Draft measure – Priority Issues", Doc ID [...], row Context aware intelligence - Ability for apps to control who receives their data and who can surface suggestions over them.

1063 See https://developer.android.com/develop/ui/views/search/appsearch, Doc ID [...], accessed on 10 March 2026.

1064 See Marchiori et al., "Private Compute Core Architecture", Google, 23 September 2022, https://arxiv.org/pdf/2209.10317v2, Doc ID [...], Table 3.

1065 Currently, on Pixel devices, proactive suggestions, which are a variety of intelligent experiences, are implemented via PSI, which does not have the *READ_GLOBAL_APP_SEARCH_DATA* permission. Alphabet's reply to RFI PO 18 of 10 April 2026, Doc ID [...], question 7, paragraph 30. [Alphabet's internal system processes and components]. ASI, which does hold the permission, could equally well offer proactive suggestions, given its access to the context-aware intelligence feature, thus making the global access to apps' data stored on-device functionality available to Alphabet's services via ASI, on top of Alphabet's services using the functionality via PSI (supported by data donated by first-party apps, as explained in recital (846)).

that is used by) its own intelligence components (through an opt-out from sharing with all),[1066] these measures remain justified, as explained in Section 4.5.3.

(847) Where the measures address access to functionalities of the context-aware intelligence feature that relate to apps, first-party apps must therefore allow that their digital context (including on-screen content), data provided ad hoc, data and actions stored on-device, and/or surfaces in/on those apps are accessed or used by a third party for proactive suggestions and intelligent experiences, in an equally effective manner as available to or used by Alphabet's services, in a non-discriminatory way without undue restrictions or conditions. As examples:

(a) Alphabet should not prevent that a third-party component generates a suggestion based on first-party app data and surfaces it in a third-party app.

(b) Alphabet should not prevent that a third-party component generates a suggestion based on third-party app data and surfaces it in a first-party app.

(c) Alphabet should not condition the availability of first-party app data to third parties for proactive suggestions and intelligent experiences on the selection of Gemini as the default assistant.

9.3.9.3. Equal effectiveness

(848) The access to device sensors, outputs, and data must be available with a level of continuous access that enables offering experiences of a proactive nature, i.e. not requiring user interaction and with the ability to appear at any time. This includes sufficient capabilities for an intelligence component to run in the background (see also Section 11.4). However, this is not the same as requiring that the intelligence component is always running or has fully continuous access. There can be flexibility in the implementation, for example only running the intelligence component after specific triggering events, such as a change of the content visible on screen, insofar as this grants effective interoperability.[1067] Moreover, for third parties to provide proactive suggestions and intelligent experiences with the same effectiveness as for Alphabet's services, the access to device sensors, outputs, and data must be provided with the same degree of granularity, frequency, and recency as are used by or available to Alphabet's services. For example, Magic Cue captures and processes screen content continuously over the last five minutes,[1068] and has access to app usage statistics for the last 30 days.[1069]

(849) Third parties offering proactive suggestions and intelligent experiences should be able to tailor the surfacing experience, for instance in terms of including the branding, icon, or link for the apps the surfaced information has been sourced from (including as provided by those latter apps),[1070] or in terms of contributing to surfacing decisions such as which surface to use. Third parties should have the necessary information available to them to do such tailoring or make such tailoring decisions, such as being aware of available surfaces to decide on the most appropriate surface for the suggestion or experience, or *vice versa* to decide on the appropriate type of suggestion or experience for the surface. For example, a short

---

1066 Alphabet's reply to RFI PO 18 of 10 April 2026, Doc ID [...], question 7a and 7b, paragraph 31, 32.

1067 Magic Cue uses background execution, e.g. for accessing screen content, but is only surfaced in response to defined triggering events. Alphabet's reply to RFI PO 7 of 27 February 2026, Doc ID [...], question 14, paragraph 84.

1068 Alphabet's reply to RFI PO 7 of 27 February 2026, Doc ID [...], question 8, paragraph 55, question 11.c, paragraph 78.

1069 Alphabet's reply to RFI PO 7 of 27 February 2026, Doc ID [...], question 11.a, paragraph 74.b.

1070 See also [third party's] submission to the public consultation, Doc ID [...], page 1, on disintermediation.

text snippet may be more appropriate for a keyboard "chip" than when the experience has longer text or images.

(850) Alphabet raised that OEMs also make design decisions about such surfaces.[1071] The Commission notes that currently, there is already a lot of freedom in the layout of the overlay that the default assistant presents, and that delegated UI also grants some control over positioning of proactive suggestions to the surfacing app. Nevertheless, the Commission considers it appropriate that OEMs retain an ability to adapt e.g. the visual design or location of proactive suggestions and intelligent experiences (e.g. setting a uniform location, shape and font)[1072] and has therefore adapted the measures to clarify that third parties do not need to be able to unilaterally design suggestions or experiences.

(851) In addition, there is no requirement that intelligence components can use arbitrary surfaces. The measures only require access to those surfaces used by or available to Alphabet's services, leaving a role to the operating system (including the OEM) to determine where and how such surfaces are offered. These surfaces could be designed in a way to let the user clearly notice that the overlay results from a proactive suggestion or intelligent experience. For example, the surfaces can be assigned to a consistent location (e.g. the bottom of the screen), they can have a particular layout or design (e.g. a glowing effect), and they can clearly attribute the component that generated the suggestion or experience and its data sources. These design elements should apply equally and non-discriminatorily to Alphabet's components. This address concerns raised by Alphabet and some third-party contributors to the public consultation about malicious overlays (via e.g. limited locations),[1073] and about low-quality suggestions rendered in apps (via attribution).[1074]

(852) It should be possible to use the context-aware intelligence feature concurrently. For example, it should be possible to display multiple suggestions from multiple intelligence components to the end user simultaneously (e.g. next to each other on the keyboard), for which it may be necessary that multiple components can read the screen content in parallel (to understand the context for which a suggestion should be provided). Alphabet itself considers this feasible: an internal documents shows that it developed [Alphabet's internal system processes and components].[1075] Alphabet has argued that concurrency may lead to degraded system health and performance.[1076] The measures allow to take technical capabilities into account when enabling concurrency, including to address system health. However, an arbitrary prohibition of concurrency, which fails to acknowledge that the system may be sufficiently

---

1071 Alphabet submission "First Mark-up of the draft measure" of 20 May 2026, Doc ID [...], page 10 (line on "Paragraph 34(i), 34(k)" in box).

1072 Alphabet internally also raised that OEMs may want to have proactive suggestions [direct quote describing Alphabet's commercial and strategic discussions]. Alphabet internal document, [Alphabet internal document], Doc ID [...], page 12.

1073 Alphabet's response to the PFs, Doc ID [...], paragraph 256, third bullet. [Third party's] submission to the public consultation, Doc ID [...], page 2. [AI service provider's] submission to the public consultation, Doc ID [...], pages 14-16. [Third party's] submission to the public consultation, Doc ID [...], page 2.

1074 Alphabet submission of 16 June 2026, "Draft measure – Priority Issues", Doc ID [...], Context aware intelligence - Ability for apps to control who receives their data and who can surface suggestions over them. [OEM's] submission to the public consultation, Doc ID [...], pages 1-2.

1075 Alphabet's internal document, Presentation [Alphabet internal document], Doc ID [...], page 31.

1076 Alphabet submission "First Mark-up of the draft measure" of 20 May 2026, Doc ID [...], page 14 (line on "Paragraph 44" in box). See also [third party's] contribution to the public consultation, Doc ID [...], page 4.

capable to enable multiple components at once (just like it can run multiple apps concurrently), would render interoperability ineffective, as it prevents multi-homing.[1077] Nevertheless, there may be circumstances where it is not possible to enable concurrency, or concurrency would be limited. For example, certain sensors may only be able to be accessed by one component at a time, or there may be certain outputs where only one or a few experiences can be displayed, e.g. due to space limitation. In such circumstances, the context-aware intelligence feature should be accessible on a non-discriminatory basis between Alphabet's own services and third-party services.[1078]

(853) The measures have been adapted to clarify that outright and wholesale replacement of existing intelligence components, such as Android System Intelligence, is not required, in particular as certain system experiences rely on such components to offer utility functions, such as Screen Attention. These experiences do not warrant replaceability, nor do they relate to an Alphabet service, such that it is not necessary to make their "surfaces" (e.g. screen rotation) accessible to third parties. Third-party components should also not be required to offer all types of proactive suggestions and/or intelligent experiences that exist already in e.g. Android System Intelligence.

(854) The measures have been adapted to clarify that Alphabet is not required to let third parties use Alphabet's cloud-based models, including cloud-based Gemini models or Alphabet's Private AI Compute environment, for their intelligence components, addressing such a concern from Alphabet.[1079] Instead, third parties should be able to access and use their own cloud-based models, including cloud-based protected execution environments.

(855) As explained in Section 12.4.5, the context-aware intelligence feature meets the exceptionality conditions that warrant limiting access to third parties that meet certain eligibility criteria. In specifying conditions specific to the context-aware intelligence feature, Alphabet should account for the proactive nature of the feature. For example, while user intent and agency can and should be incorporated in the ability for a user to consent to a component's access to the feature, Alphabet should not require that the user explicitly requests to receive the proactive suggestion or intelligent experience, as that experience would then no longer be proactive, negating the "magic" it is meant to provide in the first place (Section 9.3.1). Current implementations also do not require such explicit intent. This does not preclude that the triggering of a suggestion or experience may be tied to a user action, such as navigating to another screen or receiving a chat message.

(856) For the purpose of ensuring that effective interoperability continues in the future, Alphabet should also provide effective interoperability with any future updates, including new functionalities, of the context-aware intelligence feature insofar as they are available to Alphabet's own services or hardware. To this end, the general measures in Section 12 of this Decision apply, including in particular the measures concerning future updates and new functionalities set out in Section 12.9 in this Decision.

---

1077 See Section 12.5.

1078 The need for non-discriminatory access was raised in [AI service provider's] submission to the public consultation, Doc ID [...], page 4, and [third party's] submission to the public consultation, Doc ID [...], page 2.

1079 Alphabet's response to the PFs, Doc ID [...], paragraphs 250-251.

*9.3.10. Timing for implementation of the measures*

(857) To meet the requirements of Article 6(7) DMA, and taking into account Alphabet's proposal and the available evidence,[1080] Alphabet should implement the measures in the Annex (Section 2.2) according to the timelines indicated therein.

## 9.4. Measures for access to ambient data

*9.4.1. Introduction*

(858) Access to ambient data refers to the ability to access the stream of information produced or collected through a device's core sensors; such as the screen, microphone, speaker, and camera, whether generated by user input or system output. Such access enables AI services and assistants to maintain contextual awareness of the user's interaction with the device and surrounding environment.

(859) Ambient data constitutes an important foundation for current and possible future AI-driven services to operate seamlessly in a personalised manner that is based on information that cannot be derived from application data alone. AI services and assistants can use ambient data to provide adaptive and immediate responses. For instance, microphone input may be used to detect environmental sounds (e.g., a smoke alarm) and screen output might be used to provide suggestions. Ambient data is also tightly linked with features relating to invocation, including contextual invocation (Section 8.2) and always-on hotword detection (Section 8.3), as the service receive access to certain ambient data as part of the context provided upon invocation.

(860) The full value of access to ambient data lies in its ability to power intelligent experiences that feel intuitive and integrated into a user's daily interactions with physical and digital surroundings, A concrete example of such use is described in Section 9.3, which outlines the proactive, context-aware experiences such as Magic Cue, which are enabled by access to ambient data. By analysing real-time inputs and outputs, such as visual context, AI services can anticipate needs, enhance accessibility through live translation of conversations,[1081] or provide context for visual input.[1082] Ambient data can ultimately enable developers to craft a user experience that provides dynamic and responsive user interactions.

(861) Alphabet sees access to ambient user data as crucial for powerful AI services and assistants that ultimately provides insights beyond what can be gleaned from app data.[1083] Talking about AI opportunities, the head of the European office of Alphabet's venture capital arm said, "*[he] would be stunned if in 10 years we're not recording most of our lives all of the time because I think it would have*

---

1080 Technical meeting Alphabet/Commission of 12 June 2026, Doc ID [...], slide deck page 13. Alphabet's reply to RFI 21 of 8 June 2026, Doc ID [...], question 6, paragraph 17.

1081 For example, Gemini relies on audio input through the microphone to leverage its live translation capabilities: See https://blog.google/products-and-platforms/products/search/gemini-capabilities-translation-upgrades/, Doc ID [...], accessed on 2 March 2026.

1082 For example, Gemini relies on visual input through the camera to leverage its Gemini Live feature See https://store.google.com/intl/en/ideas/articles/gemini-camera-updates/, Doc ID [...], accessed on 2 March 2026.

1083 [Direct quote describing Alphabet's commercial and strategic discussions], Alphabet internal document, [Alphabet internal document] email of 24 June 2024, Doc ID [...], page 12.

*extraordinary utility*".[1084] Simply put, [direct quote describing Alphabet's commercial and strategic discussions].[1085]

(862) To be leveraged effectively, ambient data must be accessed persistently and processed without adverse effect on device capabilities such as battery life.[1086] For example, a feature detecting certain sounds through the microphone which informs users (e.g., alarm detection), relies on uninterrupted microphone access. Similarly, live on-screen user guidance from an AI service can only be enabled by an uninterrupted access to the screen while that session is active, including as the user navigates between apps.

(863) In addition to supporting proactive suggestions and intelligent experiences as explained in Section 9.3.2 above, the ability to access ambient data is leveraged to provide user experiences, like Gemini Live, where users "talk to" Gemini.[1087] Here, Gemini is processing camera, microphone, and screen data in a seamless manner, to provide helpful and actionable responses based on the user's sensor input and ambient context, this access relies on the publicly available APIs.

(864) Currently access to ambient data is available through either public APIs or system APIs.[1088] Direct and continuous access to raw ambient data via system APIs is reserved for services that are pre-installed and have system privileges granted by OEMs and Alphabet itself.[1089] Access through the available public APIs to ambient data is not similarly uninterrupted, nor does it benefit from a similar depth of data access or a similar streamlined user consent flow as explained in further detail below.[1090] This affects the utility of the access for AI services and assistants relying on public APIs and hampers the ability of such services to provide a fully personalised experience.[1091]

(865) The distinction between access available through public APIs and through system APIs, including the modalities under which such access is provided is determined by Alphabet. As detailed above in Section 9.3.5.2, Alphabet has exercised this discretion to provide the ability to access ambient data on a continuous basis for its own services and has actively contemplated changing other modalities of access on

---

1084 See "Why Alphabet's VC arm has doubled down on Europe", *Financial Times,* 18 March 2026, https://www.ft.com/content/f6dfc90d-7a6a-4a5f-ad56-cd9d3e1ef62a, Doc ID [...], accessed on 27 April 2026.

1085 Alphabet internal document, Presentation [Alphabet internal document] of 7 August 2025, Doc ID [...], page 5.

1086 Agreed minutes of meeting with [AI service provider] of 6 March 2026, Doc ID [...], paragraph 24-26.

1087 See https://store.google.com/intl/en/ideas/articles/gemini-camera-updates/, Doc ID [...], accessed on 2 March 2026.

1088 Technical meeting Alphabet/Commission of 19 February 2026, Doc ID [...], slide deck page 23 and Alphabet's reply to RFI 9 of 04 March 2026, Doc ID [...], question 26. Technical meeting Alphabet/Commission of 19 February 2026, Doc ID [...], slide deck page 23, specifically details the two avenues to access ambient data, namely [Alphabet's internal system processes and components].

1089 Technical meeting Alphabet/Commission of 19 February 2026, Doc ID [...], slide deck page 23 and Alphabet's reply to RFI PO 9 of 04 March 2026, Doc ID [...], question 26.

1090 For example, for access through public APIs, developers may rely on the MediaProjection APIs ( See Media projection | Android media | Android Developers, Doc ID [...], (accessed on 27 February 2026) which explicitly requires "*user consent before each media projection session*". Similarly, an internal document details an enhancement of media projection capabilities for AI agents which describes the consent flow when using MediaProjection APIs in the following way: [direct quote from Alphabet internal document describing Alphabet internal processes and components], Alphabet internal document, [Alphabet internal document], Doc ID […], page 2.

1091 [AI service provider's] reply to RFI of 20 February 2026, Doc ID [...], table 2 and agreed minutes of meeting with [AI service provider] of 20 March, Doc ID [...].

other instances. Indeed, internal documents show [Alphabet's future strategic plans for ambient data][1092] [Alphabet's future strategic plans for ambient data].[1093]

(866) On the basis of the access to ambient data, namely data produced or collected via a device's core sensors and data presented via the device's main output channels, AI-powered services including those of Alphabet itself could provide a range of personalised interactions. Access that is necessary to provide intelligent services, such as those detailed in Section 9.3, rely on access to ambient data at a low level. Conversely, the limited access to ambient data at a higher or more restrictive level that public APIs provide, would not necessarily be suitable for providing solutions comparable to Magic Cue nor as a basis to innovate and develop new services in the future.[1094]

(867) Given the considerable and tangible benefits that ambient data provides and could provide for AI services, several third-party AI service providers identified access to ambient data as particularly relevant for innovation.[1095] Indeed, certain third-party AI service providers submitted that the current access to ambient data, has kept them from developing features relying on this data.[1096]

*9.4.2. Description of the feature*

(868) Access to ambient data consists of the ability to access data produced or collected via a device's core sensors and data presented via the device's main output channels. This includes, but is not limited to, the microphone input, speaker output (system audio), camera, screen contents, location data, and other environmental sensors such as the accelerometer or proximity sensor.[1097]

(869) The access to ambient data can be leveraged to offer a wide range of functionalities that facilitates a user's interactions with their surroundings, including through intelligent solutions as outlined in Section 9.3.5.2, and to provide personal assistance to the user. Non-exhaustive examples of such use cases are:

---

1092 Alphabet internal document, [Alphabet internal document], Doc ID [...].

1093 Alphabet internal document, presentation [Alphabet internal document], Doc ID [...]. https://www.androidpolice.com/google-cosmo-makes-its-unofficial-debut/ and screenshots (retrieved on 24 June 2026), Doc ID [...], Doc ID [...], Doc ID [...].

1094 As further outlined above in Section 9.3.5.2, and further to this point agreed minutes of meeting with [AI service provider] of 20 March 2026, Doc ID [...] *"[AI service provider] stressed that the most innovation happens with access to raw data and being able to innovate on top of it. Innovation comes from the things nobody has seen before, which is only possible with low-level access.*"

1095 [AI service provider's] reply to RFI of 20 February 2026, Doc ID [...], table 2 outlines that "*[AI service] does not have access to Ambient data.*" as "*system-level permissions prevent background or always-on screen access*", in this same context, it is detailed that *"Obtaining access here and in other data categories would improve the competitiveness of rival assistants to Google's system-level assistant"* Similarly [AI service provider] minutes of 20 March 2026, Doc ID [...].

1096 [AI service provider's] reply to RFI of 10 March 2026, Doc ID [...] p.5 *"Access to such [ambient] data would enable [AI service provider] to provide new and/or enhanced services to end users."* and [AI service provider's] reply to RFI of 20 February 2026, Doc ID [...], table 2 as well as [AI service provider's] reply to RFI 1 of 13 March 2026, Doc ID [...], question 6 detailing that *"If more broadly available, certain types of ambient data would be valuable for many assistive use cases aimed at providing users both realtime and longitudinal views of their digital lives."*

1097 Technical meeting Alphabet/Commission of 19 February 2026, Doc ID [...], slide deck pages 5, 9, 10, 22. Alphabet internal document, Document [Alphabet internal document], Doc ID [...], page 3. See Marchiori et al., "Private Compute Core Architecture", Google, 23 September 2022, https://arxiv.org/pdf/2209.10317v2, Doc ID [...], Table 1. "Private Compute Core Architecture", Google, 23 September 2022, https://arxiv.org/pdf/2209.10317v2, Doc ID [...], Table 1. Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID [...], Annex Q28, Doc ID [...], second column. Alphabet internal document, presentation [Alphabet internal document], Doc ID [...], page 15.

(a) The ability to access microphone input can be leveraged for sound detection monitoring to alert end users.

(b) The ability to access screen output can be leveraged to summarise, explain, provide guidance, and organise information displayed through the device screen.

(c) The ability to access camera input can help the user understand physical surroundings, and context.

(d) The ability to access audio and screen output can enable proactive assistance which could trigger contextually relevant suggestions or actions based on cues on screen and audio such as messages and songs.[1098]

(870) Access to ambient data is central for innovation in these areas and cannot be provided without it. An AI service cannot respond to users or interact with input from the microphone unless it has access to it.[1099] Similarly, personalised explanations or indeed suggestions of on-screen content cannot be provided without access to the ambient data of the screen. Practical examples of a dependence on ambient data include access to screen content or camera for the AI service to read aloud text, or to the microphone or system audio to provide live transcription.[1100]

(871) Additionally, access to ambient data may be particularly useful for accessibility services, specifically for end users that are unable to rely on sound or visual input, Alphabet itself offers a number of solutions to this end.[1101]

*9.4.3. Technical implementation of access to ambient data in Google Android*

(872) At the base level, ambient data is gathered in Google Android from system-level sensors. The ability to access this ambient data in its raw form and on a continuous basis is subject to system-privileged permissions and is not available to third-party downloaded apps.[1102] As explained in Section 9.3.3.3, system-level access to ambient data is also subject to specific CDD requirements.[1103] Full and continuous access to ambient data in the background is enabled via a separate set of system-level APIs. These include the Content Capture API for screen context, AudioPlaybackCapture API for speaker audio, and SoundTrigger API and AmbientContext API for the microphone.[1104] OEMs may grant permission to (the APIs that provide the ability to access) that data through pre-installation.[1105]

---

1098 Agreed minutes of meeting with [third party] on 21 November 2025, Doc ID [...], paragraph 2, 4 and 9.

1099 Agreed minutes of meeting with [AI service provider] on 6 March 2026, Doc ID [...], paragraph 24 describing that *"Accessing ambient data is important for agentic experiences. However, there are obstacles to such access."*.

1100 Alphabet itself offers APIs for integrating to Gemini to offer transcription of speech to text and analysing audio input to generate text responses, see https://ai.google.dev/gemini-api/docs/audio, Doc ID [...], accessed on 13 April 2026.

1101 See for example https://www.android.com/accessibility/live-transcribe/, Doc ID [...], accessed on 13 April 2026

1102 Alphabet's response to the PFs, Doc ID [...], paragraph 267 and 268, see also [AI service provider's] reply to RFI 1 of 13 March 2026, Doc ID [...], question 6, detailing that for ambient data there is: *"No meaningful access (aside from standard notification access via NotificationListenerService when users grant the appropriate permissions)."*

1103 Alphabet internal document, Document [Alphabet internal document] Document of January 2026, Doc ID [...], Sections 9.8.6, 9.8.16. See also Technical meeting Alphabet/Commission of 19 February 2026, Doc ID [...], slide deck page 23 and 17 March 2026 Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID [...], question 26.

1104 Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID [...], Annex Q28, Doc ID [...], second column.

1105 [AI service provider's] reply to RFI of 20 February 2026, Doc ID [...], table 1.

(873) Mediated access for third-party downloaded apps is provided for certain ambient data points, through public APIs which have certain restrictions outlined further below.[1106] Alphabet explains that these public API access points include:[1107]

(a) AudioRecord API for device microphone data;

(b) Camera APIs for device camera data;

(c) MediaProjection APIs for screen content data as well as speaker data;

(d) Assist APIs for a general access to screen content data, which is however limited to the default assistant role holder and thus only available to one AI service at a time (cf. Section 5.2.4).

(874) Third parties have indicated that accessing ambient data through these public access points can be battery intensive and deficient.[1108] All of these public access points also require user permission through a specific consent process, and a so-called privacy indicator, which is usually a small symbol at the top right of the screen that tells the user that ambient data is being accessed.[1109] Conversely, system-privileged access to ambient data does not require user consent.[1110] A privacy indicator is also not shown, as Alphabet considers that "*displaying a privacy indicator continuously is inappropriate (as it would devalue the meaning of this protection for the user)*".[1111]

(875) Accordingly, access through public versus system-level interfaces differs not merely in technical implementation but also in the practical conditions and modalities under which ambient data may be collected and used.

(876) The practical utility of ambient data also depends on the availability of relevant system-level processing capabilities that allow such information to be analysed continuously with low latency and limited battery consumption. To this end, Android provides Context Hub Runtime Environment (CHRE), an environment where ambient data can be processed with a low-power processor, such as a digital signal processing (DSP) chip (see also Section 8.3). System apps can install small "nanoapps" in CHRE, which use a standardised API that is compatible across all devices that support CHRE, meaning there is no per-device work needed to create a

---

1106 9 March 2026 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID [...], question 25.i and [AI service provider's] reply to RFI 1 of 13 March 2026, Doc ID [...], question 4 detailing that: access to screen content is available *"through VoiceInteraction API"* although this access is dependent on *"user selecting [AI service] as the default assistant"* and similarly agreed minutes of meeting with [AI service provider] on 6 March 2026, Doc ID [...], paragraph 24 and 26.

1107 Technical meeting Alphabet/Commission of 19 February 2026, Doc ID [...], slide deck page 23 and 17 March 2026 and Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID [...], question 26 and 27 including footnote 8.

1108 [AI service provider's] reply to RFI of 20 February 2026, Doc ID [...], table 2 outlines that "*[AI service] does not have access to Ambient data.*" as "*system-level permissions prevent background or always-on screen access*" similarly and further detailed agreed minutes of meeting with [AI service provider] of 11 March 2026, Doc ID [...] paragraphs 21, 28 and 29 similarly, agreed minutes of meeting with [AI service provider] of 6 March 2026, Doc ID [...], paragraph 24 describing that *"These factors make screen access in particular subject to a cumbersome and technically strenuous access, firstly due to the consent process which is accompanied by a recording indicator at all times for the end users and secondly because the access is battery intensive."*

1109 Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID [...], question 26 and 27 including footnote 8 which describes: *"these routes generally require user permission and a privacy indicator save for AppAssist (which is available to the default assistant role-holder).*

1110 Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID [...], question 26, paragraph 60.

1111 Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID [...], question 26, paragraphs 60 and 61a.

nanoapp that processes ambient data efficiently.[1112] Android System Intelligence (Section 9.3.3.1) relies on CHRE for intelligent experiences such as cough and snore detection.[1113]

(877) The CHRE is supported on a variety of devices from five of the largest OEMs, including Alphabet itself.[1114] Where sensitive ambient data is processed by Alphabet's own intelligence components, those processing capabilities are complemented by the Private Compute Core framework, which implements the CDD requirements through dedicated isolation and privacy mechanisms and is not available to downloaded third-party applications, as described in Section 9.3.3.3.

*9.4.4. Feature falls within the scope of Article 6(7) DMA*

(878) The Commission considers that the access to ambient data feature as described in Section 9.4.2 – together with its functionalities – is controlled via Google Android and falls within the scope of Article 6(7) DMA.

*9.4.5. Current access for Alphabet's own services*

(879) The access to ambient data is available to and used by Alphabet's services.

(880) Alphabet's first-party services use the access to ambient data feature, as defined in Section 9.4.2, which consists of the ability to access data produced or collected via a device's core sensors under some modality of access. For example, for Gemini, end users can enable sharing of ambient data simply by tapping to share the screen, audio or camera, as demonstrated by Gemini Live.[1115] The ambient data accessed for the purposes of Gemini Live is subject to the user prompting the service and a privacy indicator.

(881) Moreover, Alphabet's first-party services can and have benefitted from access to ambient data via system-privileged APIs. As described above, this level of integration can enable Alphabet's own services to work in a seamless and concerted manner and provide personalised and intelligent user experiences, whereas third-party services are effectively restricted to rely on the public APIs described above, as provided and defined by Alphabet. The system-privileged APIs provide a more direct access to the ambient data, which in turn may enable improved properties for use such as quality, frequency and reduced consent or privacy safeguards for the access, which form an important dimension of effective interoperability.

(882) While Gemini currently only directly integrates with these publicly provided APIs, its access to ambient data could benefit from improved access at Alphabet's discretion. Indeed, Alphabet's services, including Gemini, already benefit from and use access to ambient data, in particular via intelligence components and their access to system-privileged permissions, as described in Section 9.3.5.1. For example, Alphabet's services use access to ambient data to provide personalised suggestions through Magic Cue's continuous access to screen content, with Alphabet planning even deeper integration of Gemini with those capabilities, as described in Section 9.3.5.2.[1116] This demonstrates that Alphabet's own services benefit from a broader set

---

1112 See https://source.android.com/docs/core/interaction/contexthub, Doc ID [...], accessed on 15 April 2026.

1113 Alphabet's reply to RFI PO 16 of 01 April 2026, Doc ID [...], question 11, paragraph 25.

1114 Alphabet's reply to RFI PO 16 of 01 April 2026, Doc ID [...], question 7, paragraph 21.

1115 See https://support.google.com/gemini/answer/15274899?hl=en&co=GENIE.Platform%3DAndroid Doc ID [...], accessed on 27 March 2026.

1116 For further explanation, see also for example [AI service provider's] reply to RFI of 20 February 2026, Doc ID [...], paragraph 3.4 and table 1.

of modalities for accessing and using ambient data than those generally available to third-party services.

(883) Internal documents specifically point to Alphabet's [Alphabet's commercial and strategic discussions].[1117] These internal documents show [Alphabet's internal system processes and components].

(884) Therefore, the Commission considers the access to ambient data is available and used by Alphabet within the meaning of Article 6(7) DMA.

*9.4.6. Current access for third-party services and hardware*

(885) Third parties do not enjoy equally effective interoperability with the access to ambient data feature.

(886) Third-party services, in particular AI services, have an interest in accessing ambient data.[1118] For example, the Perplexity assistant recommends that users "ask about screen context".[1119]

(887) Unless granted access to system-privileged APIs and in certain cases the default assistant role (Section 5.2.4), third-party services are only able to access ambient data through:

(a) The public APIs created by Alphabet as detailed above. This access is always subject to a runtime user consent activated each time access is prompted and must be accompanied by a privacy indicator and by its nature restricts continuous access.[1120]

(b) So-called accessibility APIs created by Alphabet that are specifically intended for aiding users with accessibility issues and unsuited for general use by AI services. Indeed, on Google Play, usage of the AccessibilityService API is subject to specific permission from Alphabet and restricted to *"services that are designed to help people with disabilities access their device or otherwise overcome challenges stemming from their disabilities are eligible to declare that they are accessibility tools.*"[1121]

(888) As such, only services with system privileges can access ambient data on a continuous basis without runtime user consent or privacy indicators.[1122] This differentiated access to ambient data is therefore not available to third-party user-downloaded apps and effectively limited to Alphabet and OEM services as detailed above in Section 5.4. Certain types of ambient data, such as speaker audio for voice calls, are wholly unavailable to third-party user-downloaded apps.[1123]

---

1117 Alphabet internal document, [Alphabet internal document] of 11 June 2025, Doc ID [...], page 2

1118 Agreed minutes of meeting with [AI service provider] on 12 March 2026, Doc ID [...], paragraph 24 as well as [AI service provider's] reply to RFI of 20 February 2026, Doc ID [...], table 2 describing that: *"Obtaining access here and in other data categories would improve the competitiveness of rival assistants to Google's system-level assistant"*.

1119 See screenshot of the Perplexity app setup experience, https://static0.anpoimages.com/wordpress/wp-content/uploads/2025/01/screenshot_20250114-122951.png, Doc ID [...], from https://www.androidpolice.com/perplexity-ai-new-android-voice-assistant-is-here/, Doc ID [...].

1120 Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID [...], question 26, paragraph 60. [AI service provider's] reply to RFI PO of 10 March 2026, Doc ID [...].

1121 See https://support.google.com/googleplay/android-developer/answer/10964491?hl=en, Doc ID [...], accessed on 27 April 2026.

1122 Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID [...], question 26, paragraph 60.

1123 This requires the *CAPTURE_AUDIO_OUTPUT* permission, which is only available to privileged system apps. See https://developer.android.com/media/platform/sharing-audio-input, accessed on 15 April 2026, Doc ID [...], Section "Voice call + ordinary app". Email from [AI service provider], sent 23

(889) More generally, third parties submit that the current access to ambient data for third-party user-downloaded apps relies on solutions that are resource-intensive, and less efficient than those offered to system-privileged apps.[1124] For example, third-party services state to have higher latency (i.e. delay) when accessing ambient data and may consume more battery while doing so.[1125]

(890) Third-party services cannot access runtime environments to efficiently process ambient data using a low-power processor. CHRE is only available to system-trusted apps; it is not open for use by third-party downloaded apps.[1126] This makes that interoperability for third-party services is not equally effective, as this also depends on the availability of power-efficient runtime environments for processing ambient data.

(891) As a result, third-party services relying on public APIs cannot process ambient data under conditions that are equally effective to those enjoyed by Alphabet's services, even where access to the underlying ambient data inputs may be available via public APIs. Third parties that rely on the currently available public APIs, are inherently limited by the specific framework created by Alphabet itself for these public APIs, namely that access is subject to runtime user consent, privacy indicators, limitations on continuous background access, restrictions on the level of data that may be accessed, and the absence of efficient processing capabilities.

(892) Certain third parties even choose not to rely on ambient data in their AI services, hindering innovation, due to restrictions currently imposed such as the repeated consent requirement or need to be the default assistant role holder for certain access to ambient data.[1127] Indeed, a submission to the public consultation for example explains that *"A third-party assistant operating under standard applications constraints cannot maintain the same quality of ambient awareness, regardless of the sophistication of the model. The result is a class of use cases – live translation, real-time object recognition, continuous health monitoring, proactive location-based suggestions – that are structurally inaccessible to third-party assistants at equivalent quality levels under the current framework*".[1128]

*9.4.7. Alphabet's response to the Preliminary Findings*

(893) Alphabet considers that the draft measures are unlawful.[1129] Alphabet contends that it already complies with Article 6(7) DMA, as first-party and third-party apps access sensor data under equal consent-based conditions,[1130] and that any differences between downloaded and preinstalled apps are justified by integrity concerns.[1131] Alphabet argues that broader access would introduce unacceptable risks to Google

---

March 2026, Doc ID [...], page 2. Agreed minutes of meeting with [AI service provider] of 2 April 2026, Doc ID [...].

1124 Agreed minutes of meeting with [AI service provider] of 11 March 2026, Doc ID [...].

1125 Agreed minutes of meeting with [AI service provider] of 11 March 2026, Doc ID [...].

1126 See https://source.android.com/docs/core/interaction/contexthub, Doc ID [...], accessed on 15 April 2026. Alphabet's reply to RFI PO 16 of 01 April 2026, Doc ID [...], question 12, paragraph 27.

1127 Agreed minutes of meeting with [AI service provider] of 11 March 2026, Doc ID [...]. [AI service provider's] reply to RFI of 20 February 2026, Doc ID [...]. [AI service provider's] reply to RFI 1 of 13 March 2026, Doc ID [...]. [AI service provider's] reply to RFI 1 of 10 March 2026, Doc ID [...].

1128 [AI service provider's] submission in response to the Commission's public consultation, 13 May 2026, Doc ID [...], page 23.

1129 Alphabet's response to the PFs, Doc ID [...], paragraph 261, 263-268

1130 Alphabet's response to the PFs, Doc ID [...], paragraphs 263, 264

1131 Alphabet's response to the PFs, Doc ID [...], paragraphs 267, 268

Android system security and stability which justifies restricting access to trusted, system-level components.[1132]

(894) Alphabet further argues that ambient data is mis-defined in the PFs,[1133] and that its services such as Gemini do not access ambient data, and instead rely on the same user-visible, consent-based APIs available to third parties.[1134] Alphabet only accesses ambient data when acting in a system-level capacity, under the same safeguards as applied to any third party.[1135] In this regard, Alphabet explains that the CDD explicitly enables OEMs to access and secure sensitive data on-device, and they remain free to engage alternative service providers.[1136]

(895) In its response to the PFs, Alphabet considered that the draft measures concerning ambient data were disproportionate.[1137] as they would undermine OEM control over sensitive device features which would compromise system health, user security, and privacy.[1138] Alphabet also contended that the measures would only have "speculative benefits" since downloaded apps have alternative ways of accessing the same data.[1139]

*9.4.8. The Commission's assessment*

9.4.8.1. Concerning the existing level of interoperability and integrity

(896) As outlined in Section 9.4.2, access to ambient data comprises the ability to access and process information produced or collected through a device's core sensors and output channels. Alphabet's argument that the PFs conflate ambient data and sensor data does not make a difference, in light of this description of "ambient data" for the purpose of this Decision.[1140] The Commission's assessment relates to access to device context derived from sensors, and the modalities and conditions under which such access is provided. Whether Alphabet classifies some elements of this as "ambient data" or as "sensor data" in internal policies does not alter the fact that different modalities of access exist and are subject to conditions determined by Alphabet.

(897) Access to ambient data depends on specific conditions established within Google Android, including system-level privileges, roles, and pre-installation, which Alphabet acknowledges.[1141] In turn, these conditions affect the depth of access available, the applicable user consent mechanisms and the ability to access such data continuously or in the background. Alphabet can unilaterally decide to change these modalities of access to ambient data, including the quality, continuity, latency, consent mechanisms and other characteristics that determine the practical utility of such access, to the benefit of its own services or detriment of third-party services (see Section 4.4).

---

1132 Alphabet's response to the PFs, Doc ID [...], paragraph 261
1133 Alphabet's response to the PFs, Doc ID [...], paragraph 261 and 263
1134 Alphabet's response to the PFs, Doc ID [...], paragraph 263, 264, 266
1135 Alphabet's response to the PFs, Doc ID [...], paragraph 261
1136 Alphabet's response to the PFs, Doc ID [...], paragraph 261
1137 Alphabet's response to the PFs, Doc ID [...], paragraph 261 and 269
1138 Alphabet's response to the PFs, Doc ID [...], paragraph 261 and 269
1139 Alphabet's response to the PFs, Doc ID [...], paragraph 261 and 269
1140 Alphabet's response to the PFs, Doc ID [...], paragraph 263.
1141 Alphabet's response to the PFs, Doc ID [...], paragraph 264

(898) Alphabet occupies a dual role in relation to access to ambient data, which is underlined by its own claims.[1142] As a result, Alphabet is simultaneously responsible for determining the conditions governing access to ambient data and a potential beneficiary of that access. As a point of illustration of this dual role, Alphabet itself explained that when considering possible approaches for enhanced access to ambient data, namely the screen context, *"Android refused Gemini's requests for changes*".[1143] Contrary to Alphabet's claims,[1144] the risk of preferential access for Alphabet's services is therefore not theoretical or speculative: business needs have already driven the Gemini team to request changes to Android, and those needs may well drive Alphabet to accommodate such requests in future.

(899) Where Alphabet considers that third parties can access ambient data under equal conditions through OEM preinstallation and holding system intelligence roles,[1145] this is not generally available to third-party services, and does not enable third parties to access the feature under the "same" condition, as discussed in Section 5.4.1. Indeed, internal Alphabet documents confirm that conditioning access to ambient data on OEM approval creates barriers for providers competing with Alphabet. [Alphabet's internal system processes and components].[1146]

(900) The Commission does not contest Alphabet's claim that the Gemini app does not directly access ambient data differently than third-party services.[1147] However, direct integration is not determinative for the assessment of whether the feature is available to Alphabet's own services, as explained in Sections 5.3 and 9.3.5.1. In any event, as outlined in Section 9.3.5, Gemini already uses ambient data in a way that is unavailable to third parties. Such preferential access may then lead to improved conditions such as latency or power consumption, unavailable to third parties.[1148] [Cosmo], an experimental AI assistant app developed by Alphabet, which accessed ambient data that is unavailable to third parties,[1149] further illustrates how Alphabet can contemplate, develop, or test attractive AI experiences due to the availability of access to ambient data to its services.

(901) Alphabet argues that access to ambient data raises particular user privacy (which is not an integrity concern, see Section 4.8.1), and system-performance concerns, including risks caused by continuous sensor access such as increased battery consumption, degraded device performance and the need to preserve appropriate isolation and sandboxing mechanisms,[1150] which is echoed in some third-party responses to the public consultation.[1151] Alphabet is free to apply relevant safeguards

---

1142 In Alphabet's response to the PFs, Doc ID [...], paragraph 261, Alphabet describes that *"where Google receives access to ambient data, it is only where it is playing such a system-level role, either as an OEM (in the case of Pixel) or as a service provider for Android OEMs."*

1143 Alphabet's response to the PFs, Doc ID [...], paragraph 264

1144 Alphabet's response to the PFs, Doc ID [...], paragraph 266, mentioning that *"The PFs speculate that the Android OS could be altered to give Gemini ambient data access"*; paragraph 269.

1145 Alphabet's response to the PFs, Doc ID [...], paragraph 264, first bullet.

1146 Alphabet internal document, Document [Alphabet internal document], Doc ID [...], page 9

1147 Alphabet's response to the PFs, Doc ID [...], paragraphs 261, 264 and 266.

1148 Alphabet's response to the PFs, Doc ID [...], paragraphs 265.

1149 The types of ambient data covered include "screen context", "screen interactions", "ambient audio" and "media audio". https://www.androidpolice.com/google-cosmo-makes-its-unofficial-debut/ and screenshots (retrieved on 24 June 2026), Doc ID [...], Doc ID [...], Doc ID [...].

1150 Alphabet's response to the PFs, Doc ID [...], paragraph 264-267

1151 See for example, [third party] submission in response to the Commission's public consultation, Doc ID [...], p. 3. [OEM] submission in response to the Commission's public consultation, Doc ID [...], p. 1. [AI service provider's] contribution to the Commission's public consultation, Doc ID [...], paragraph 43.

in a non-discriminatory manner. Importantly, the final measures in this Decision clarify that Alphabet needs to provide a deeper modality of access to ambient data (e.g. without privacy indicator) only if that modality is made available to Alphabet's services, in line with the requirement to allow effective interoperability under equal conditions.

(902) However, the existence of such risks does not in itself establish that access must be conditioned on OEMs pre-granting, pre-installing, allocating system intelligence roles or privileges.[1152] In any event, the requirement of OEM preinstallation cannot be construed as a strictly necessary and proportionate integrity measure,[1153] as discussed in Section 6.3.1. While Alphabet explains why ambient data should be subject to safeguards, it does not demonstrate why this access must depend on OEM discretion, or why equivalent mitigation could not be achieved through other means. The measures annexed to this Decision give Alphabet the additional flexibility to implement security and privacy measures on feature access by third parties in line with principles of isolation and sandboxing. Alphabet's claim that this cannot be guaranteed for downloaded applications and therefore requires cooperation with OEMs is addressed in recitals (823)-(824).

9.4.8.2. Concerning proportionality

(903) Alphabet argued in its response to the PFs that the measures for ambient data are disproportionate. However, Alphabet does not substantiate why the measures would be more restrictive than necessary to achieve the objectives of Article 6(7) DMA. In particular, Alphabet fails to propose any less restrictive, yet equally effective means of complying with that provision. Nor does Alphabet explain and substantiate why the feature and/or a specific measure imposes a disproportionate burden on Alphabet. Instead, its arguments focus on questioning the scope of Article 6(7) and the actual need for interoperability – a line of reasoning which does not meet the legal standard for a valid proportionality claim, as explained in Section 13 below.

(904) The Commission nonetheless addresses each of Alphabet's arguments in this respect.[1154]

(905) Alphabet's alleged harm to end users due to privacy and security risks[1155] is already rebutted in Section 9.4.8.1 above and not further substantiated.

(906) Alphabet's contention that feature access would result in a loss of control for OEMs is addressed in Section 6.1.2. Alphabet quotes concerns from several OEMs (and a certification organisation) over device integrity, privacy, and security.[1156] The measures do not prevent appropriate safeguards for integrity, privacy, and security.

(907) Alphabet claims that the benefits of the measures are speculative and that "*it is not clear what problem the Draft Measures are trying to solve*".[1157] Section 9.4.6 shows that an interoperability gap does exist. More broadly, access to ambient data enables AI-powered services to understand user's surrounding context with the potential to provide personalised, adaptive and intelligent services, enabling a unique set of

1152 Alphabet's response to the PFs, Doc ID [...], paragraph 264-268.
1153 Alphabet's response to the PFs, Doc ID [...], paragraph 267.
1154 Alphabet's response to the PFs, Doc ID [...], paragraph 269.
1155 Alphabet's response to the PFs, Doc ID [...], paragraph 269, first bullet.
1156 Alphabet's response to the PFs, Doc ID [...], paragraph 269, second bullet. [Third party's] submission in response to the Commission's public consultation, Doc ID [...]. [OEM's] submission in response to the Commission's public consultation, Doc ID [...]. [OEM's] contribution to the Commission's public consultation, Doc ID [...].
1157 Alphabet's response to the PFs, Doc ID [...], paragraph 269, third bullet.

services and innovation. Third parties confirm this importance of ambient data.[1158] They expressed how access to ambient data may enable "*new and/or enhanced services to end users*",[1159] and that "*Accessing ambient data is important for agentic experiences".*[1160] Third-party submissions to the public consultation similarly confirm this view: *"The competitive significance of ambient data is also structural"*,[1161] "*Without this access, a third-party assistant is limited to responding to what the user consciously chooses to share; with it, the assistant can perceive and respond to the user's environment in the way that makes AI assistance genuinely transformative*",[1162] and in relation to device context: *"Third-party AI services should have access to the same structured integrations and APIs used by Google's own AI services, rather than relying on indirect methods such as screen automation."*[1163] Further, internal documents [Alphabet internal document discussing the relevance of ambient data for AI experiences].[1164] Access to this feature is therefore necessary for third-party services to offer comparably rich experiences as Alphabet's own services.

(908) The Commission therefore concludes that there is an interoperability gap with respect to the access to ambient data feature of Google Android. As a result, third parties and in particular user-downloaded apps, cannot perform important functionalities that are expected of an AI assistant which places them sub-par in comparison with Alphabet's own AI-powered services. As a result, the Commission considers that it should specify how Alphabet must fill this gap to comply with Article 6(7) DMA.

*9.4.9. Measures that Alphabet should implement*

(909) To meet the requirements of Article 6(7) DMA, Alphabet should implement the measures set out in Section 2.3 of the Annex to this Decision. The interoperability solution for the ambient data feature, as described in Section 9.4.2 of this Decision, must be equally effective as that available to any of Alphabet's own services.

(910) The measures are limited to what is necessary to ensure effective interoperability for the access to ambient data. They do not require Alphabet to introduce new forms of access to ambient data or to expand such access beyond what is available to Alphabet's AI-powered services. The measures do not require unrestricted access to ambient data where such access is not provided for Alphabet's own complementary services. The measures expressly leave discretion for Alphabet to apply necessary safeguards to protect user privacy, security, and the integrity of Google Android, including process isolation, encryption and user control mechanisms, provided they

---

1158 Agreed minutes of meeting with [AI service provider] of 12 March 2026, Doc ID [...], paragraph 24 as well as [AI service provider's] reply to RFI of 20 February 2026, Doc ID [...], table 2 describing that: *"Obtaining access here and in other data categories would improve the competitiveness of rival assistants to Google's system-level assistant"*. [AI service provider] submission in response to the Commission's public consultation, 13 May 2026, Doc ID [...], page 15 describes that *"Ambient data access enables a qualitatively distinct category of AI assistant functionality – one that is proactive, continuous and grounded in the user's physical environment rather than dependent on explicit queries."* and mentions *"The competitive significance of ambient data access is also structural."*

1159 [AI service provider's] reply to RFI of 10 March 2026, Doc ID [...] page5.

1160 Agreed minutes of meeting with [AI service provider] of 6 March 2026, Doc ID [...], paragraph 24.

1161 [AI service provider] submission in response to the Commission's public consultation, Doc ID [...], page 23.

1162 [AI service provider] submission in response to the Commission's public consultation, Doc ID [...], pages 23-24.

1163 [AI service provider] submission in response to the Commission's public consultation, 13 May 2026, Doc ID [...], page 3.

1164 Alphabet internal document, Presentation [Alphabet internal document], Doc ID [...], page 5.

allow for equally effective interoperability. Indeed, the measures takes the sensitive nature of ambient data into account by requiring equality in the modalities of access to ambient data, including with regard to user informed consent mechanisms, privacy-preserving methods and general integrity safeguards.

(911) The measures address the structural risk that, absent specification, Alphabet may, now or in future, offer enhanced modalities of access to ambient data to its own AI-powered services while third-party providers remain limited to routes that provide less effective access, including through differences in continuity of access, background availability, latency and consent-related interruptions, including the frequency and nature of user prompts and indicators, or prioritisation of access. The measures require equivalent access, including for concrete modalities such as privacy indicators and user consent. This equivalence should be established by the practical conditions governing the collection, processing and use of ambient data. For example, processing in a protected execution environment may obviate the need for privacy indicators, while processing in a way where Alphabet or a third party could access the data may require user consent. As described in Section 12.4.5, depending on the final implementation and in particular on future changes to modalities of access, access to ambient data could meet the requirements of a restricted feature within the meaning of this Decision.

(912) For the reasons described in Section 4.5.3, the Commission further considers it necessary to specify a measure to address the ambient data generated by Alphabet's own applications. Specifically, Alphabet should allow third parties to access ambient data related to Alphabet's applications (such as screen content), whenever such functionality is enabled for Alphabet's AI-powered services. Given the importance of Alphabet's application ecosystem within Google Android, excluding ambient data generated through those applications would materially reduce the practical effectiveness of interoperability and grant an unfair competitive advantage to Alphabet.

(913) The measures foresee that, where Alphabet's services rely on dedicated "nanoapps" installed on low-power runtime environments to achieve efficient processing of ambient data, third parties can access those dedicated apps via APIs exposed by those apps. This ensures that effective interoperability is not undermined by differences in performance, latency and energy efficiency due to preferential access to those environments. This measure is adapted from a draft measure annexed to the PFs, where it was originally part of the context-aware intelligence feature. The measure was moved to the ambient data feature without material impact, as the original draft measure was already related to the processing of ambient data, and reduced in scope to address Alphabet's concerns on the matter,[1165] now requiring only integration with existing "nanoapps" and not the ability for a third party to install new "nanoapps" themselves.

(914) For the purpose of ensuring that effective interoperability continues in the future, Alphabet should also provide effective interoperability with any future updates, including new functionalities, of the ambient data feature insofar as they are available to Alphabet's own services or hardware. To this end, the general measures in Section 12 of this Decision apply, including in particular the measures concerning future updates and new functionalities set out in Section 12.9 in this Decision.

---

1165 Alphabet's response to the PFs, Doc ID [...], paragraph 256, second bullet.

*9.4.10. Timing for implementation of the measures*

(915) To meet the requirements of Article 6(7) DMA, and taking into account Alphabet's proposal and the available evidence,[1166] Alphabet should implement the measures in the Annex (Section 2.3) according to the timelines indicated therein.

## 10. MEASURES ON FEATURES FOR ACTIONS ON APPS AND THE OS

### 10.1. Introduction

(916) Modern AI services, such as AI assistants and AI agents, integrate deeply with apps, services and the operating system. These AI services are expected to handle tasks on behalf of the user and perform actions across different apps, websites and services.[1167] These AI services use the user's prompts and context to deduce which actions need to be taken and performed on that user's device. Alphabet expects users to adopt AI tools and shift their habits on mobile devices away from "*opening apps to do tasks step-by-step"* and towards "*asking AI to do the heavy lifting for them."*[1168] An AI service's ability to perform tasks and integrate with other apps and the operating system therefore represents vital capabilities to attract and retain users. Importantly, it relies on the presence of accessible integration channels.

(917) On Google Android devices, AI services can currently interact with, and execute actions within, other apps in a number of ways.

(918) The first type is *structured integration*, where apps or the OS formally declare which specific actions they expose, and which parameters those actions take. For example, a music app might declare "play song" as an action, with a song title as parameter. In practice, such integration involves interactions between (at least) two apps, or an app and the OS. It requires the participation of (i) "*exposing apps*" (or the OS) to expose potential integrations, e.g. by making actions, data and metadata available to other apps, including AI services, and (ii) "*caller apps*" (or the OS) to call the integration, i.e. calling upon the functionality that the exposing app has made available, and execute actions.

(919) Alphabet splits these structured integration channels into two broad categories:

(a) "OS mechanisms", or *on-device integration*, which consist of the Android intent system, App Actions, and App Functions; and

(b) "App-level integration", or *service-level integration*, such as cloud-based APIs, other proprietary solutions and non-OS open-source solutions.[1169]

(920) While these are complementary integration channels, on-device integration offers certain advantages over service-level integration. For instance, on-device execution can access on-device data and reuse other apps' existing authentication, has enhanced privacy and latency, and crucially, can function offline for some tasks.[1170]

---

1166 Technical meeting Alphabet/Commission of 12 June 2026, Doc ID [...], slide deck page 13. Alphabet's reply to RFI 21 of 8 June 2026, Doc ID [...], question 6, paragraph 17.

1167 See https://android-developers.googleblog.com/2026/02/the-intelligent-os-making-ai-agents.html, Doc ID [...], accessed on 20 April 2026.

1168 See https://android-developers.googleblog.com/2026/02/the-intelligent-os-making-ai-agents.html, Doc ID [...], accessed on 20 April 2026.

1169 Technical meeting Alphabet/Commission of 26 February 2026, Doc ID [...], slide deck page 6.

1170 Alphabet internal document, Document [Alphabet internal document], Doc ID [...].

These factors contribute to a more seamless integration with apps installed on the device, which matches users' expectations.[1171]

(921) The second type is *automated integration*. Here, the app executing the integration ("controlling app") takes control over the exposing app ("controlled app"), and imitates user behaviour through button taps, swipe gestures, and keyboard input. This form of agentic control is also termed "UI Automation": the controlling app is automating the existing app's user interface by behaving like a user. The major advantage is that the controlled app does not need to declare or support any action: its user interface suffices to let the controlling app do its work.

(922) When handling tasks on the user's behalf, AI services can chain together actions and interaction paradigms, within and across apps. For example, to plan a trip for the user, the AI service may use a structured on-device integration to consult the user's calendar, service-level integration with an airline to book a flight, and automated integration on a hotel booking app, all in one flow based on a single user prompt. To support this, AI services should be able to reliably discover which integrations are available on the device.

(923) Equivalent access to integrations with apps and the OS is crucial to enabling third-party services to deliver at least as rich a set of tasks as available to Alphabet's first-party services. If the first-party service has exclusive access to integration channels with apps or the OS, third-party services are severely impaired in their ability to be useful to the user. In turn, this severely impacts the competitive process as users are more likely to choose the incumbent gatekeeper service over exploring and selecting a third-party alternative. Therefore, third parties need to enjoy the same degree of on-device integration as Alphabet's services. In addition, in terms of scope, to ensure a level playing field and deliver equivalent access, integration must touch all of the end user's apps and on-device services, including those of the gatekeeper, as well as the operating system itself.

(924) Based on the assessment developed hereafter, the Commission considers that there is currently an interoperability gap when it comes to features for actions on apps and the OS for third-party services on Google Android mobile devices.

(925) The Commission therefore specifies how Alphabet must fill this gap, to comply with Article 6(7) DMA. The measures address these various, complementary means of integrations with apps and the OS, namely structured on-device integration, in particular via App Actions and App Functions (Section 10.2), screen automation (Section 10.3), and system integration with the OS (Section 10.4). All these integrations contribute to effectively enabling interoperability within the meaning of Article 6(7) DMA for third-party AI services.

## 10.2. Measures for structured on-device integration

### *10.2.1. Introduction*

(926) Structured on-device integration enables services to take actions inside other apps in response to a user's request. For example, an AI assistant could, following a user query, call upon an email app on the device to send an email that the AI assistant drafted. Structured on-device integration can also be used to share data between

[1171] Agreed minutes of meeting with [AI service provider] of 28 October 2025, Doc ID [...], paragraph 14.

apps, in particular for data that changes frequently (e.g., stock prices), in complement to other app data sharing mechanisms (Section 9.2).[1172]

(927) Structured on-device integration lets services offer functionality beyond what is embedded within that service itself, by leveraging the other apps that the user has installed. Just like apps and websites from third-party developers make a mobile ecosystem highly attractive and useful, integrating with a wide variety of services makes AI services such as assistants more powerful.

(928) Relative to cloud-based, or app-level, integration, structured on-device integration provides several significant benefits to developers and users. It can offer operational benefits, such as improved reliability and offline functionality, as well as exclusive access to device-specific features like accessing encrypted messages and setting alarms. Services using structured on-device integration can also offer higher privacy and security guarantees, as data is kept on the device. Finally, structured on-device integration can solve the issue of discovering the services available for integration: apps declare that they offer an integration and the OS can keep a central index of integrations that other services can consult to find actions available to take.

(929) Currently, the two primary channels through which Alphabet's AI services engage in structured on-device integration, namely App Actions and App Functions, are not as seamlessly, or in many instances at all, accessible to third parties. As a result, Google Android grants a significant advantage to Alphabet's services by effectively keeping these integration channels largely exclusive to Google Assistant and now also Gemini.

*10.2.2. Description of the feature*

(930) The structured on-device integration feature allows services to interact with other apps installed on the user's device and perform tasks within those apps on the user's behalf. The integration is structured: exposing apps formally declare which specific actions other services can execute, and which parameters those actions take.

(931) The structured on-device integration feature includes, amongst others, the following functionalities:

(a) The ability for apps to discover all available structured on-device integrations, including metadata necessary to determine which actions to trigger;

(b) The ability for apps to execute structured on-device integrations;

(c) The ability to let the user be aware of and observe that a structured on-device integration is being executed; and

(d) The ability to execute the structured on-device integration within the caller app, i.e. without app switching (unless that is the intent of the structured on-device integration).

*10.2.3. Technical implementation of structured on-device integration in Google Android*

(932) There are currently two primary channels through which structured on-device integration takes place on Google Android mobile devices: App Actions and App Functions.

[1172] Alphabet internal document, internal email of 4 April 2025 at 12:20:44 with subject [Alphabet internal document], Doc ID [...], page 2.

10.2.3.1. App Actions

(933) App Actions are built on top of the Android intent system, which is the core Android messaging framework that lets apps establish a communication channel with other apps and request that the operating system or another app perform a specific action. App Actions provide a structured manner for developers to implement and expose these integrations which, once triggered by a caller app, hand users over to the app exposing the integration.[1173] App Actions have been supported since Android 5 and can be executed by a third-party downloaded app if it is aware of an App Action.[1174]

(934) Alphabet confirmed that App Actions are specific to Google Assistant[1175] and dozens of apps were described by Alphabet as having a high usage of App Actions.[1176] However, the established discovery mechanism for App Actions is exclusively available to Google Assistant. To implement App Actions and make them discoverable, developers are instructed to take the following steps:

(a) Developers identify the action to expose, for which they must use an intent. They can reuse an intent predefined by Alphabet, called built-in intents (BIIs), if it matches the type of action (one example is "starting a workout"),[1177] or, if none of the BIIs match, the developer defines their own, fully custom intent;[1178]

(b) Developers declare these intents in a file inside their app[1179] which, once discovered, confirms which App Actions are supported and explains where to find the definition of the intents for the supported actions;[1180]

(c) Developers then upload their app to the Google Play Console for publication. Once the App Actions are reviewed and approved by Alphabet,[1181] Alphabet registers the declared intents, which then become discoverable to Google Assistant through a server-side integration with Google Play;[1182] and

(d) Developers must also publish their app to the Google Play Store, as App Actions are only available for apps published there.[1183]

(935) Alphabet has therefore developed an implementation process which leads developers to donate their actions through Alphabet's proprietary mechanisms, which are only made available to Google Assistant.[1184] The use of this proprietary discovery mechanism ultimately deprives the operating system or third parties from discovering them and prevents third parties from effectively using this feature and

---

1173 Alphabet's reply to RFI PO 2 of 23 September 2025, Doc ID [...], paragraph 130.
1174 Alphabet's reply to RFI PO 12 of 13 March 2026, Doc ID [...], questions 10.
1175 Alphabet's reply to RFI PO 12 of 13 March 2026, Doc ID [...], question 10.
1176 Alphabet's reply to RFI AI 2 of 23 September 2025, Doc ID [...], Annex Q54, Doc ID [...].
1177 See https://developer.android.com/develop/devices/assistant/overview#build_app_actions, Doc ID [...], accessed on 20 April 2026.
1178 See https://developer.android.com/develop/devices/assistant/custom-intents, Doc ID [...], accessed on 20 April 2026.
1179 The shortcuts.xml file.
1180 See https://developer.android.com/develop/devices/assistant/action-schema, Doc ID [...], accessed on 20 April 2026.
1181 Alphabet internal document, [Alphabet internal document], Doc ID [...], page 3.
1182 Alphabet's reply to RFI PO 12 of 13 March 2026, Doc ID [...], question 9.
1183 See https://developer.android.com/develop/devices/assistant/get-started, Doc ID [...], accessed on 20 April 2026.
1184 See https://developer.android.com/develop/devices/assistant/overview, Doc ID [...], accessed on 20 April 2026.

accessing its functionalities, despite App Actions being assigned a "public" Android permission level (Section 5.2.3) which should grant full access to its functionalities.

10.2.3.2. App Functions

(936) App Functions was released in Android 16 and aims to automate tasks and streamline user interactions.[1185] In its internal documents, Alphabet described App Functions as [direct quote describing Alphabet's commercial and strategic discussions].[1186] App Functions was therefore designed to extend AI assistants' capabilities beyond generating content to allowing assistants to interact with other apps on users' behalf and fulfil users requests directly within those apps.[1187] Gemini integrates with App Functions, which offer additional benefits over earlier generation integration models, including App Actions,[1188] such as:

(a) Ease of integration: App Functions enable a caller app to dynamically discover and call on any available App Function; and

(b) User experience: App Functions can be executed within the caller app, rather than handing users over to the app exposing the action. App Functions also allow the chaining of multiple actions without requiring user interaction at each stage.[1189]

(937) Apps can expose far-reaching functionalities through App Functions, as the feature is open-ended in the sense that it could allow the caller app to execute any task that users could execute within the apps.[1190] Alphabet published a guide which illustrates some of App Functions' capabilities through example use cases, ranging from task management, such as setting a reminder, to cross-app workflows, which could involve the use of functions across multiple apps by collecting information from one app to inform an action in another app.[1191]

(938) Many preinstalled OEM apps (such as Samsung Clock, Calendar and Photos) currently integrate with App Functions. Alphabet has also identified [Alphabet's internal strategy][1192] and has engaged in onboarding discussions with popular third-party apps, such as [third party app][1193] and [third party app].[1194] Alphabet also intends to integrate several first-party apps (namely Gmail, Calendar, Docs, Drive, Messages, Dialer, YouTube and Maps) through structured on-device integration channels, including App Functions.[1195]

(939) The typical flow of how App Functions are exposed by exposing apps, and subsequently executed by a caller app, is illustrated in the diagram below. To expose an App Function, the following steps happen:[1196]

(a) **App Function declaration:** The Android app exposes its App Functions, such as "Create Note" or "Send Message", by describing and donating the functionalities. The creation of an App Function is straightforward as

---

1185 See https://developer.android.com/ai/appfunctions, Doc ID [...], accessed on 20 April 2026.
1186 Alphabet internal document, Document [Alphabet internal document], Doc ID [...], page 3.
1187 Alphabet internal document, Document [Alphabet internal document], Doc ID [...], page 2.
1188 Alphabet's reply to RFI AI 2 of 23 September 2025, Doc ID [...], paragraph 120.
1189 See https://developer.android.com/ai/appfunctions, Doc ID [...], accessed on 20 April 2026.
1190 Alphabet's reply to RFI AI 2 of 23 September 2025, Doc ID [...], paragraph 120.
1191 See https://developer.android.com/ai/appfunctions, Doc ID [...], accessed on 20 April 2026.
1192 Alphabet internal document, Document [Alphabet internal document], Doc ID [...], page 4.
1193 Alphabet internal document, Document [Alphabet internal document], Doc ID [...].
1194 Alphabet's reply to RFI PO 1 of 06 February 2026, Doc ID [...], Annex Q13.19, Doc ID [...].
1195 Alphabet's submission of 18 June 2026, [Alphabet submission], Doc ID [...].
1196 See https://developer.android.com/ai/appfunctions, Doc ID [...], accessed on 20 April 2026.

developers already write and annotate the necessary code for their apps today; and

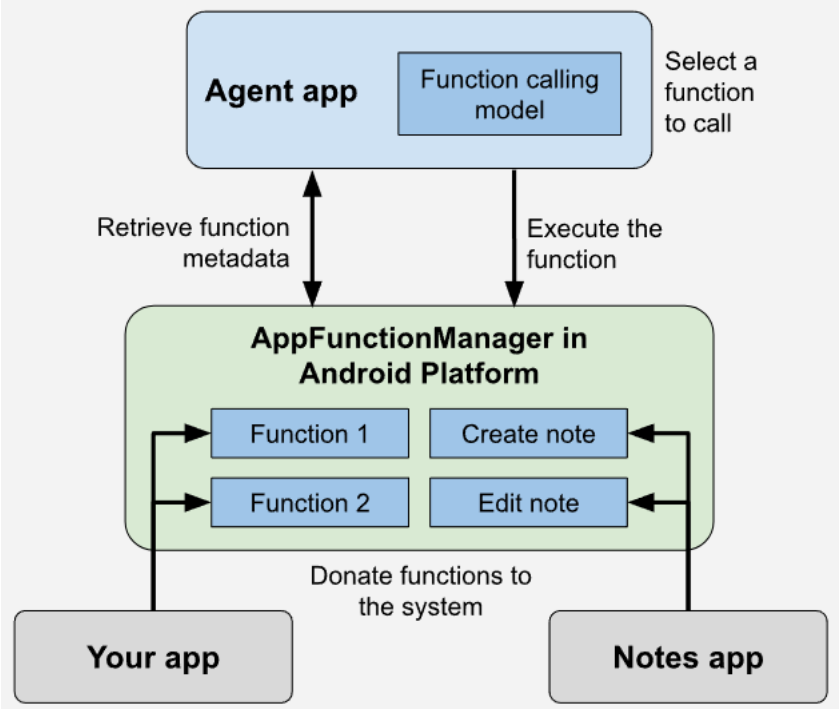
(b) **Schema generation:** The App Functions compiler generates a file that lists all the declared App Functions and stores it in the app. On installation of the app, this file is read by the Android OS to discover and index App Functions. This index is currently made centrally available via the App Function Manager, which stores the index in the AppSearch database.[1197]

(940) For an app to execute an App Function, it must perform the following steps to retrieve the App Function and then subsequentially execute it:

(a) **App Function retrieval:** The app that wants to execute App Functions can search for and retrieve metadata of available App Functions, either all or a filtered subset,[1198] by invoking the *searchAppFunctions* API of the App Function Manager.[1199]

(b) **App Function selection and execution:** Based on user prompts, the app will select which function is appropriate for a given user request. Once a function has been selected, the app executes it by calling the *executeAppFunction* API of the App Function Manager with the relevant function identified.[1200]

**Figure 4. The typical flow of how App Functions are exposed and subsequently executed by an agent.**




*Source: https://developer.android.com/ai/appfunctions, Doc ID […]*

(941) App Functions are discovered "dynamically" because the caller app obtains the set of App Functions currently available from exposing apps on the device at runtime,

1197 Alphabet internal document, Document [Alphabet internal document] of 14 August 2025, Doc ID [...], page 11.

1198 See https://developer.android.com/reference/android/app/appfunctions/AppFunctionSearchSpec.Builder, Doc ID [...], accessed on 20 April 2026.

1199 See https://developer.android.com/reference/android/app/appfunctions/AppFunctionManager, Doc ID [...], accessed on 20 April 2026.

1200 Alphabet's reply to RFI PO 9 of 04 March 2026, Doc ID [...], question 4.

rather than relying on a fixed or pre-configured integration with specific apps. This differs from integration channels such as App Actions, which are based on pre-defined schemas (i.e. the BII described in Section 10.2.3.1) as well as rule-based matching, which require a one-on-one mapping between specific user queries and actions, instead of being adaptable to new queries and actions.

(942) Under Alphabet's current implementation of App Functions, the ability to execute App Functions is subject to OEM pre-installation and the granting of a privileged permission. OEMs must grant apps the *EXECUTE_APP_FUNCTIONS* permission to execute an App Function in another app. As a result, user-downloaded apps cannot currently access this feature under any circumstances (see Section 5.2).

(943) Finally, App Functions follow an *app caller agnostic* model: a developer that chooses to expose an App Function automatically allows every app with the *EXECUTE_APP_FUNCTIONS* permission to invoke it. The App Functions framework does not provide an opt-out mechanism, meaning that developers may not choose which apps are allowed to invoke their App Functions.

*10.2.4. Feature falls within the scope of Article 6(7) DMA*

(944) The Commission considers that the structured on-device integration feature as described in Section 10.2.2 – together with its functionalities – is controlled via Google Android and falls within the scope of Article 6(7) DMA.

*10.2.5. Current access for Alphabet's own services and hardware*

(945) The structured on-device integration feature, as currently implemented via App Actions and App Functions, is available and used by Alphabet, as described below.

10.2.5.1. App Actions

(946) App Actions are marketed and implemented by Alphabet as the means through which users can launch apps, perform tasks, access content on other apps via Google Assistant.[1201] As described in Section 10.2.3.1, Alphabet instructs developers to expose their apps directly to the Google Assistant ecosystem. App Actions is therefore currently available to and used by Alphabet.

10.2.5.2. App Functions

(947) App Functions can only be accessed by apps that have been granted the requisite privileged permissions. As described in Section 5.3, Gemini has access to this feature on all major OEM devices that run Android 16. As the provider of Google Android, Alphabet could even ensure that Gemini has access to these features on all Google Android devices, including those where Gemini (more precisely, the Google app) is not pre-installed, by simply changing the Google Android access rules applicable to App Functions (see Section 5.2.1). App Functions is therefore currently available to and used by Alphabet.

*10.2.6. Current access for third-party services and hardware*

(948) Third parties cannot access and interoperate with the structured on-device integration feature, as currently implemented via App Actions and App Functions, as effectively as Alphabet's services.

[1201] See https://developer.android.com/develop/devices/assistant/overview, Doc ID [...], accessed on 20 April 2026.

10.2.6.1. App Actions

(949) Third-party services, including downloaded apps, can call App Actions, but face significant challenges in effectively using App Actions due to their difficulties discovering which actions are available. Currently, the guidance issued by Alphabet to developers wishing to donate App Actions is specifically targeted at Google Assistant and relies on the Google Play Console discovery channel which is exclusively available to Alphabet.[1202]

(950) By contrast, third-party user-downloaded apps must rely on developers making their App Actions discoverable through other means, such as publishing a list of supported actions. Not all developers who expose App Actions will do so, making certain App Actions unavailable to third-party user-downloaded apps. Indeed, third-party service providers, such as [AI service provider] and [AI service provider], confirmed that while in theory they can call App Actions, their inability to practically discover them prevents them from interoperating with this feature as effectively as Alphabet.[1203] This imbalance represents significant challenges for third parties, despite the fact that the interaction remains mediated by the operating system and, more specifically, the Android intents system.

10.2.6.2. App Functions

(951) Third-party services do not have equally effective access to App Functions. As described in Section 10.2.3.2, App Functions are guarded by privileged permissions. Section 5.2 explains that, to access features with privileged permissions, the OEM must grant the app the permissions upon preinstalling the app onto the device. As a result, third-party user-downloaded apps cannot access this feature.

(952) Alphabet submitted on 3 October 2025 that, aside from Gemini, the only other AI Assistant it was aware of that supported App Functions was [AI service] on [OEM] devices.[1204] Since that date, [OEM] also agreed to grant [AI service] access to the App Functions feature on certain [OEM] devices [confidential information on this access]. Other AI service providers attempted to obtain pre-installation agreements with OEMs, which is pre-requisite for gaining access to this software feature. However, even in instances in which they successfully secured pre-installation agreements, these parties were unable to secure access to the requisite system privileges which would grant them access to relevant software features, such as App Functions, as described further in Section 5.4.

(953) For completeness, Alphabet has not granted access to App Functions to any third-party service on its own Pixel mobile devices.[1205]

*10.2.7. Alphabet's response to the Preliminary Findings*

(954) Alphabet argued that the draft measures were unlawful because Alphabet already allows first-party and third-party services to call (or expose) App Functions under equal conditions.[1206] Specifically, the requirement to be preinstalled and granted system privileges applies to both Alphabet's own services and third-party services.

---

1202 See https://developer.android.com/develop/devices/assistant/get-started, Doc ID [...], accessed on 20 April 2026.

1203 Agreed minutes of meeting with [AI service provider] of 3 November 2025, Doc ID [...] and agreed minutes of meeting with [AI service provider] of 28 October 2025, Doc ID [...].

1204 Alphabet's reply to RFI AI 2 of 23 September 2025, Doc ID [...], question 58.

1205 Alphabet's reply to RFI PO 19 of 17 April 2026, Doc ID [...], question 8.

1206 Alphabet's response to the PFs, Doc ID [...], paragraph 271.

(955) Alphabet also argued that the risk that AI assistants are manipulated into performing sensitive tasks that the user did not anticipate or approve must be guarded against.[1207] Alphabet identified factors that raise the risk profile of App Functions and argued that services executing App Functions, i.e. caller apps, must have the functional capability of controlling other apps safely, which in its view requires advanced AI reasoning and LLM capabilities.[1208] In particular, Alphabet argued that App Functions is vulnerable to the "confused-deputy" problem, whereby legitimate AI assistants are tricked by malicious apps into misusing their capabilities, or prompt injections as sources of vulnerabilities.[1209] Moreover, Alphabet argued that the measures would harm end users, OEMs, business users and Alphabet, given the risks involved, which would outweigh any benefits associated with improving access to this feature.[1210]

(956) As a result, Alphabet argued that limiting access to App Functions to services with pre-granted permissions by OEMs is commensurate with the risks associated with opening up access this feature, which cannot be mitigated through less restrictive means.[1211] In the context of restricted features, as described within Section 4.8 and expanded upon within Section 12.4.5, Alphabet argued that the Commission's measures must be expressly scoped to capture App Functions due to malware concerns analogous to the accessibility API.[1212]

(957) With regards to App Actions, Alphabet argued that the Commission misdescribes the feature and scope of Alphabet's obligations under Article 6(7) DMA. Specifically, Alphabet characterises App Actions as a branded product which performs the role of proprietary discovery mechanism for Google Assistant and therefore acts as an implementation mechanism which does not form part of the software feature controlled by Google Android.[1213]

(958) Finally, Alphabet argued that the measures are disproportionate because they do not account for the alternative channels available to developers to engage in structured app integration, such as Android OS SDK APIs, the Android OS intent system and bespoke integrations.[1214]

*10.2.8. The Commission's assessment*

10.2.8.1. Concerning the existing level of interoperability and integrity

(959) Alphabet's claim that there is no interoperability gap because third-party and first-party services can gain access to App Functions through OEMs is already addressed in Section 10.2.6, which Alphabet does not factually contest. As further explained in Section 4.6 above, Article 6(7) DMA requires actual "interoperability" and "access", not the mere possibility of obtaining feature access subject to conditions and importantly subject to the unilateral decision of individual OEMs. As such, the fact that OEMs can theoretically grant a third-party app access to the feature does not suffice to ensure effective interoperability across Google Android operating system, as evidenced by the fact that securing preinstallation agreements, through which access can be granted to privileged permissions, represents a significant hurdle that

1207 Alphabet's response to the PFs, Doc ID [...], paragraph 275.
1208 Alphabet's submission of 11 May 2026, Doc ID [...], page 17.
1209 Annex 1 of Alphabet's response to the PFs, Doc ID [...], paragraphs 45 and 46.
1210 Alphabet's response to the PFs, Doc ID [...], paragraph 279.
1211 Alphabet's response to the PFs, Doc ID [...], paragraph 274.
1212 Alphabet submission dated 20 May 2026, Doc ID [...], Section 3.1.
1213 Alphabet's response to the PFs, Doc ID [...], paragraph 273.
1214 Alphabet's response to the PFs, Doc ID [...], paragraph 279.

very few (to the Commission's knowledge only one) third-party non-OEM AI assistants have been able to overcome.

(960) Alphabet positions its claims regarding the risks of the feature and the resulting limitation to pre-installed apps as relating to integrity.[1215] Alphabet partially puts forward arguments which in fact relate to the protection of the user's security (by addressing misuse by "untrustworthy" callers) rather than protection of operating system integrity, in fact calling out the need for "security guardrails".[1216] The Commission nonetheless addresses Alphabet's arguments in this respect below. Alphabet also raises arguments relating to "*unauthorised execution of sensitive actions*" or "*the chaining of actions in ways that a user did not […] consent to*".[1217] Insofar as this relates to the impairment of controls designed to prevent unauthorised access to the feature, integrity measures can be applied to ensure that any access is authorised (see Section 4.8.1).

(961) In any event, as further detailed in Section 4.8.4 above, a gatekeeper cannot justify an integrity measure (or for that matter, a security measure, given that equally effective interoperability must be provided under equal conditions) implemented in relation to third parties' services or hardware solely based on whether the gatekeeper controls or trusts such third parties. In particular, a gatekeeper may not justify measures by the mere fact that third parties are not the gatekeeper, and therefore they cannot be trusted.

(962) Going forward, Alphabet explained that it was also [describes Alphabet's proposed requirements for AI assistants].[1218] Alphabet explained that, as part of the OEM's preinstallation process for a particular AI assistant, the OEM would nominate assistants, at which point Alphabet would add the assistant to the allowlist if Alphabet considers that it complies with the security or privacy guidelines.[1219]

(963) [Describes Alphabet's commercial and strategic discussions regarding App Functions].[1220] [Describes Alphabet's commercial and strategic discussions regarding App Functions].[1221] This shows that more tailored solutions are available to Alphabet to address any risks associated with widening access to App Functions.

(964) Moreover, the need for continued OEM involvement appears to be at odds with other internal documents. For instance, when assessing the security concerns and allocation of responsibilities associated with App Functions, Alphabet states that [quote from ad description of Alphabet internal document].[1222] Alphabet's multi-party consent model for AI agents also [description of Alphabet's development plans regarding a multi-party consent model for AI agents].[1223]

---

1215 Alphabet's response to the PFs, Doc ID [...], paragraph 271.

1216 Alphabet's response to the PFs, Doc ID [...], paragraph 275.

1217 Alphabet's response to the PFs, Doc ID [...], paragraph 275.

1218 Alphabet's reply to RFI PO 9 of 04 March 2026, Doc ID [...], paragraph 13. Alphabet submitted that this allowlist mechanism would be implemented on Android 17 Google Android mobile devices, delivered through a system of server-side flags by a GMSCore-managed service and enforced via the operating system.

1219 Alphabet's reply to RFI PO 9 of 04 March 2026, Doc ID [...], question 10.

1220 Alphabet's reply to RFI PO 9 of 04 March 2026, Doc ID [...], question 10.

1221 See, for instance, Alphabet internal document, Document [Alphabet internal document], Doc ID [...], pages 4 and 5.

1222 Alphabet internal document, Presentation "App Functions DevEx – Plan of Record", Doc ID [...], slide 60.

1223 Alphabet internal document, Document [Alphabet internal document], Doc ID [...], page 5.

(965) As a result, additional OEM approvals provide no discernible advantages from a risk mitigation perspective, despite creating significant hurdles for third parties seeking access to App Functions. Furthermore, the requirement to obtain a preinstallation agreement with OEMs to access App Functions does not merely take into account the risk profile of AI assistants but also includes other commercial and financial considerations, as discussed in Section 5.4.1. It is therefore clear that less restrictive measures are available to Alphabet to mitigate any risks.

(966) In any event, due to the sensitivity of this feature, this Decision explicitly allows Alphabet to limit access to the structured on-device integration feature only to third parties that meet certain eligibility criteria, which include criteria relating to integrity, security, and privacy, as described further within Section 12.4.

(967) With regards to the discoverability challenges associated with App Actions, Alphabet's internal documents acknowledge [description of Alphabet internal documents concerning impact of discoverability challenges on interoperability]. Alphabet considered [description of Alphabet's commercial and strategic discussion regarding App Functions][1224] [description of Alphabet's commercial and strategic discussion regarding App Functions].

(968) As explained in Section 4.6, interoperability pursuant to Article 6(7) DMA needs to be effective, meaning that it must enable the desired result to be achieved in practice. The interoperability solution should therefore be workable in practice without unnecessary hurdles. Yet Alphabet's development of a proprietary discovery mechanism for App Actions, which is exclusively available to Google Assistant and is embedded into Alphabet's primary app distribution channel, fundamentally undermines this objective and prevents third parties from interoperating with App Actions as effectively as Alphabet's own services.

10.2.8.2. Concerning proportionality

(969) Alphabet also made a proportionality argument. However, Alphabet does not substantiate why the measures would be more restrictive than necessary to achieve the objectives of Article 6(7) DMA. In particular, Alphabet fails to propose any less restrictive, yet equally effective means of complying with that provision. Nor does Alphabet explain and substantiate why the feature and/or a specific measure imposes a disproportionate burden on Alphabet. Instead, its arguments focus on questioning the scope of Article 6(7) and the actual need for interoperability – a line of reasoning which does not meet the legal standard for a valid proportionality claim, as explained in Section 13 below.

(970) The Commission nevertheless addresses Alphabet's arguments regarding the availability of other app integration channels, which the Commission does not consider to be equally effective for several reasons. Regarding the intent system, structured on-device integration channels offer several advantages over the intent system, such as an established discoverability mechanism, ease of integration and enhanced user experience, as described further in Section 10.2.3.2.

(971) App Functions also offer several benefits over app-level (i.e. bilateral) integrations, such as:

(a) On-device access, leading to lower latency, ability to work offline and privacy advantages and automated discoverability; and

[1224] Alphabet internal document, Document [Alphabet internal document], Doc ID [...].

(b) Obviating the need for app developers to do any extra work to support third parties. Developing a parallel and bespoke integration for a third party involves additional work that apps might be less inclined to invest in if they already integrate with App Functions. This position aligns with concerns raised by a third party that, absent an extension of equivalent access to App Functions for third-party assistants, "*the possibility of a significant limitation on interoperability for competitors to Gemini remains, making it harder for competing AI developers to plan and develop new services for users on reasonable timelines, potentially placing those competitors at a material disadvantage.*"[1225]

(972) Furthermore, the limitations associated with app-level integrations are illustrated by the experiences of third-party AI assistants that currently use Alphabet's public APIs to integrate with Alphabet's services, such as Gmail, Calendar and YouTube. Currently, both Alphabet and third parties mostly integrate with some of Alphabet's services via cloud APIs. However, the APIs available to third parties are not equally effective to those available to Gemini. For example, Alphabet explained that the [details on effectiveness of private APIs and public APIs for integration with some of Alphabet first-party services].[1226] [details on effectiveness of private APIs and public APIs for integration with some of Alphabet first-party services].[1227]

(973) The substantial barriers in attempting to integrate services with Alphabet's apps via such cloud APIs is reflected in the experiences of third-party AI developers. For example, [AI service provider] explained that Alphabet applies to public APIs several types of quotas and rate limitations that do not apply to Alphabet's own services (which use private APIs) and that are atypical in the industry.[1228] Similarly, quotas and rate limitations affect third parties' ability to provide the same semantic search[1229] functionality that Gemini supports. [AI service provider] explained that if it had to use only the currently available public API for semantic search, at a minimum it would have to first retrieve all the user data. However, in practice this approach would not be feasible due to quotas and rate limitations, as well as latency issues.[1230] These limitations affect the ability of third-party AI services to scale, especially during peaks of activity or new user registrations. [AI service provider] explained that, in addition to Alphabet's approval, these APIs also require user consent and periodic reauthorisation/renewal, introducing friction that does not apply to Alphabet's services.[1231]

---

1225 Third-party submission dated 16 April 2026, Doc ID [...], paragraph 8.

1226 Alphabet's reply to RFI PO 10 of 6 March 2026, Doc ID [...], Annex to question 2, Doc ID [...].

1227 Alphabet's reply to RFI PO 10 of 6 March 2026, Doc ID [...], Annex to question 2, Doc ID [...].

1228 [Describes Alphabet's commercial agreements with third parties]. [AI service provider's] reply to RFI of 6 march, Doc ID [...], question 8.

1229 Semantic search allows to find and retrieve information in a way that understands, on one hand, the user's intent and, on the other hand, the contextual meaning of terms and concepts in the dataset where the information is searched. Semantic search therefore goes beyond searching for literal matches. For example, a user can ask an AI assistant "When is my trip to Spain?" and the AI assistant will be able to find the calendar event with title "Flight to Barcelona", which contains neither "trip" nor "Spain". Semantic search is generally implemented by processing data (often server-side) to add special labels and metadata that can be efficiently searched for in an "AI-native" way. Technically, this often relies on vector embeddings, like those mentioned above in relation to Alphabet's private APIs.

1230 Agreed minutes of meeting with [AI service provider] of 20 March 2026, Doc ID [...].

1231 [AI service provider's] reply to RFI of 20 February, Doc ID [...], question 8.

(974) Integration with Alphabet's services is critical to provide an AI assistant that can compete on par with Gemini.[1232] Supporting some prominent functionalities in key Alphabet apps is important to offer a complete and powerful agentic experience to users, who expect the agent to execute all sorts of actions on their behalf based on user's contextual data retrieved across services, including in terms of scope of the data and in terms of supported use cases, such as semantic search and similar use cases that enable contextual interaction between the user and the service using natural language.[1233]

(975) In order for any AI assistant to be able to compete on par with Gemini, it is crucial to be able to meet users' needs where they are today. A very high number of Android users use Alphabet's services that are seamlessly integrated with Android and Gemini. Most users will not or cannot move away from these services just to use another AI assistant, because that would require the user to undertake time-consuming and/or high-friction migrations to services that are different from those that they use today – which most users are not willing to do.[1234] This is exactly the type of ecosystem lock-in that Alphabet leverages to develop, integrate, and favour new AI services in Google Android. Interoperability as mandated by Article 6(7) DMA is critical to ensure that Alphabet's unmatched ecosystem advantage does not automatically translate into an unmatched advantage for Alphabet's new services, such as Gemini.

(976) Consequently, to put third-party AI assistants in a position to have a chance to compete, they must be able to integrate with the main Alphabet first-party apps (namely Gmail, Calendar, Docs, Drive, Messages, Dialer, YouTube and Maps) through structured on-device integration channels to support some important use cases that are supported by Gemini – see next section. This is necessary to ensure that interoperability with the structured on-device integration is effective in practice, alleviating the current disparities of access between Alphabet and third parties that stem from Alphabet's preferential access to cloud APIs.

(977) The Commission therefore concludes that there is currently an interoperability gap with respect to the structured on-device integration feature of Google Android. This gap creates significant challenges for third parties and in particular user-downloaded apps, which cannot perform important functionalities that are expected of an AI assistant and puts them at a disadvantage relative to Google's own AI powered service. As a result, the Commission considers that it should specify how Alphabet must fill this gap to comply with Article 6(7) DMA.

*10.2.9. Measures that Alphabet should implement*

(978) To meet the requirements of Article 6(7) DMA, Alphabet should implement the measures set out in Section 3.1 of the Annex to this Decision. The interoperability solution for the structured on-device integration feature, including App Actions and App Functions, as described in Section 10.2.2 of this Decision, must be equally effective as the solution available to Alphabet's services.

(979) Given that developers exposing structured on-device integrations may choose to do so only via one channel, Alphabet should make available all channels of structured

---

1232 Agreed minutes of meeting with [AI service provider] of 20 March 2026, Doc ID [...], agreed minutes of meeting with [AI service provider] of 6 March 2026, Doc ID [...], [AI service provider's] submission of 22 March 2026, Doc ID [...].

1233 Agreed minutes of meeting with [AI service provider] of 6 March 2026, Doc ID [...].

1234 Agreed minutes of meeting with [AI service provider] of 20 March 2026, Doc ID [...].

on-device integration to third parties, as well as any possible future implementation of structured on-device integration.

(980) To interact with other apps installed on the user's device and perform tasks within those apps on the user's behalf, third party assistants must integrate with apps on device in a structured way, whereby they can discover and execute specific actions declared by other services. Alphabet should therefore provide effective interoperability with all functionalities of the structured on-device integration feature, as available to Alphabet's services and in a way that is equally effective as the solution available to Alphabet's services. As described in Section 12.4.5, App Functions meets the exceptionality conditions that warrant limiting access to third parties that meet certain eligibility conditions.

(981) To ensure that third-party AI services benefit from an interoperability solution going forward, and that this interoperability solution is equally effective in a sense that it has access to all necessary functionalities, Alphabet should also grant third parties access to additional functionalities available to Alphabet's services, if necessary to enable effective interoperability with the structured on-device integration feature.

(982) For the reasons explained in Section 4.5.3, Alphabet should also refrain from exposing functionalities of its first-party apps via structured on-device integration channels that would allow only Gemini (or other Alphabet services) to invoke them. Any functionality of Alphabet's first-party apps exposed via any implementation of the structured on-device integration feature should be available through implementations that are interoperable for third parties, such as App Functions.

(983) Furthermore, to avoid a situation whereby Alphabet can reserve integration for important use cases in key apps to Alphabet's AI assistants by resorting to preferential cloud-based integration, Alphabet should also enable Qualified services to execute specific functionalities in the relevant Alphabet first-party apps via structured on-device integration channels. These functionalities enable key use cases that are particularly relevant for integration with AI assistants in response to a user query, such as: the ability to retrieve emails, messages, and other documents based on contextual search queries expressed by users in natural language; the ability to create calendar events and send emails; the ability to control navigation in Google Maps – e.g. to change destination or show points of interest on the map – and to control playback in YouTube, e.g. to play/pause a video. Alphabet should be permitted to apply strictly necessary and proportionate limits to prevent abuse, e.g. via quotas, provided that third parties can use the functionalities as effectively as Alphabet's services, including in terms of user friction and latency to retrieve the necessary user data.

(984) For the purpose of ensuring that effective interoperability continues in the future, Alphabet should also provide effective interoperability with any future updates, including new functionalities, of the structured on-device integration feature insofar as they are available to Alphabet's own services or hardware. To this end, the general measures in Section 12 of this Decision apply, including in particular the measures concerning future updates and new functionalities set out in Section 12.9 in this Decision.

*10.2.10. Timing for implementation of the measures*

(985) To meet the requirements of Article 6(7) DMA, and taking into account Alphabet's proposal and the available evidence,[1235] Alphabet should implement the measures in the Annex (Section 3.1) according to the timelines indicated therein.

## 10.3. Measures for screen automation

*10.3.1. Introduction*

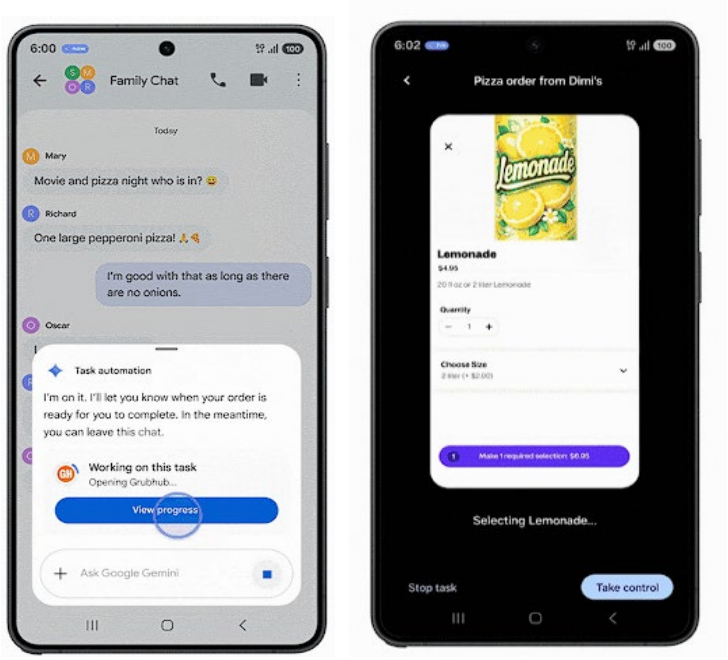
(986) To perform actions on the user's request, apps can autonomously take over control of the user's installed apps to execute tasks inside the app, without the need for explicit support by the app developer. To interact with apps, the controlling app observes the screen to learn and understand which functionalities are offered by the controlled app, and then traverses the user interface, using simulated user interactions and inputting information through simulated touch, click, and type events, to complete the user's desired task. In this Section the following terms apply:

(a) A "controlling app" is an app that takes control over apps and interacts with those apps by imitating user interactions.

(b) A "controlled app" is an app that is controlled by a controlling app.

(987) For example, the user can task a controlling app with ordering the items from their shopping list from the local supermarket. Screen automation allows the controlling app to open the shopping list app in the background, look at its contents, and then open the app of the supermarket to order each item from the list.[1236]

**Figure 5. Left screenshot shows Gemini starting a task with Computer Control. Right screenshot shows how the user can observe the AI control of the app.**




*Source: McCullough, M. (2026), 'The Intelligent OS: Making AI agents more helpful for Android apps', Android Developers Blog, 25 February, https://android-developers.googleblog.com/2026/02/the-intelligent-os-making-ai-agents.html, Doc ID […].*

---

1235 Technical meeting Alphabet/Commission of 12 June 2026, Doc ID […], slide deck page 13. Alphabet's reply to RFI PO 21 of 8 June 2026, Doc ID […], question 6, paragraph 17.

1236 See McCullough, M. (2026), 'The Intelligent OS: Making AI agents more helpful for Android apps', Android Developers Blog, 25 February, https://android-developers.googleblog.com/2026/02/the-intelligent-os-making-ai-agents.html, Doc ID […].

(988) Access to this feature means that the controlling app is guaranteed to be able to interact with all the apps on the user's device. Automated app interactions are highly useful for cases where app developers have not updated their apps to offer explicit integration points (see Section 10.2), or if they offer integration with only some features. As adoption of dedicated app integrations progresses (e.g. through App Functions, Section 10.2), screen automation fills the gap such that services can still offer powerful automation to its users.

(989) In recent Android versions, Alphabet has implemented automation features which allow apps to take control of other apps and interact with the user interface (UI). Alphabet has launched a screen automation API and Gemini is using this API to automate a small number of apps in the US and South Korea.[1237] However, access to this API is subject to system privileges, which creates a substantial barrier to access for third parties.

*10.3.2. Description of the feature*

(990) Screen automation bridges the gap between services and apps that do not offer integration points for these services yet or apps that do not offer integration points for all features that the app supports. The controlling app can take control of entire apps running on the user's device.[1238] It would control the app similar to a user by observing the screen, tapping buttons, inputting text and performing gestures, such as swiping to scroll through the content.

(991) In the scenario with the most automation, the user would not see any of the interactions by the controlling app with apps, and tasks would be completed end-to-end in the background. However, the OS or the controlling app can offer the user visibility into the automation, e.g. by observing how the controlling app is automating the controlled app, as well as let the user intervene, in particular for important decisions or consequential actions, such as confirming orders or completing purchases.[1239] Observation is also necessary to build trust between the user and the controlling app.

(992) The functionalities of the Screen automation feature include, amongst others, the following functionalities.

(a) **Taking control over apps.** Ability for an app (controlling app) to take control of other apps (controlled apps) including those of third parties, the OEM, and Alphabet. The controlling app can perform interactions with the controlled apps by imitating user interactions, such as taps, clicks, and typing;

(b) **Accessing the on-screen content.** Ability for the controlling app to access the screen content of the controlled app to understand its user interface;

(c) **Discovering apps.** Ability for the controlling app to determine which apps are installed on the device and retrieve relevant information to decide which app to automate;

---

1237 See McCullough, M. (2026), 'The Intelligent OS: Making AI agents more helpful for Android apps', Android Developers Blog, 25 February, https://android-developers.googleblog.com/2026/02/the-intelligent-os-making-ai-agents.html, Doc ID […].

1238 See McCullough, M. (2026), 'The Intelligent OS: Making AI agents more helpful for Android apps', Android Developers Blog, 25 February, https://android-developers.googleblog.com/2026/02/the-intelligent-os-making-ai-agents.html, Doc ID […].

1239 See McCullough, M. (2026), 'The Intelligent OS: Making AI agents more helpful for Android apps', Android Developers Blog, 25 February, https://android-developers.googleblog.com/2026/02/the-intelligent-os-making-ai-agents.html, Doc ID […].

(d) **Controlling apps in background.** Ability for a controlling app to fully control of other apps in background. The process of controlling apps is not visible to the user. The controlled app will be run on a virtual display to enable background execution. The controlling app can complete the task while the user uses the phone otherwise or decides not to use their phone;

(e) **User observation.** Ability for the user to surface the controlled app and observe the controlling app interacting with it. In addition, the user can stop the controlling app from interacting with the controlled app;

(f) **Seamless transition.** Ability to transition seamlessly between background and foreground when the user chooses to observe the controlled app;

(g) **OS indicators.** Ability for OS indicators to demonstrate to the user a controlling app is currently interacting with another app;

(h) **Concurrency management.** Ability for Google Android to manage concurrent access, including scheduling or preventing multiple accesses. This can include user-facing mechanisms, such as greying out the app icon of the controlled app so that the user cannot use the same app at the same time as the controlling app; and

(i) **Blocking sensitive areas.** Ability for controlled apps to block sensitive views from access of the controlling app. This functionality allows apps to protect sensitive user data, critical settings, or purchase decisions from being automated by the controlling app.

*10.3.3. Technical implementation of screen automation in Google Android*

(993) Screen automation, is currently implemented in Android via the Computer Control API, an Android AOSP API.[1240] The screen automation feature launched in March 2026 in the third quarterly platform release of Android 16 (QPR3). The feature is currently available on the Google Pixel 10 series and on the Samsung Galaxy S26 series and it is used by Gemini to automate certain food, groceries, and rideshare apps.[1241]

(994) The controlling app can discover all installed apps on the user's device and retrieve metadata that is required to use the Computer Control API with any of the installed apps. The required metadata includes the app's name to show to the user which app should be controlled and the app's package name, which identifies the app towards the Android OS.

(995) The Computer Control API uses a virtual display on which the controlling app can automate apps. Using a virtual display is key to Computer Control as it allows the controlling app to do all interactions in background. The controlling app can launch apps, take screenshots of them to understand its content and imitate user interactions such as tapping, swiping, and typing text.

---

1240 See Google Git ComputerControlSession.java, https://android.googlesource.com/platform/frameworks/base/+/45034f0663f960d9ee5fb0a101a4732b71f6e2f4/core/java/android/companion/virtual/computercontrol/ComputerControlSession.java, Doc ID […].

1241 See McCullough, M. (2026), 'The Intelligent OS: Making AI agents more helpful for Android apps', Android Developers Blog, 25 February, https://android-developers.googleblog.com/2026/02/the-intelligent-os-making-ai-agents.html, Doc ID […].

(996) The controlling app uses the Computer Control API to start a so-called *Computer Control Session*,[1242] passing a list of apps to take control of. For example, the controlling app indicates it seeks to control a shopping list app and the local supermarket app.

(997) Figure 6 shows the Computer Control loop that allows it to interact with apps.[1243]

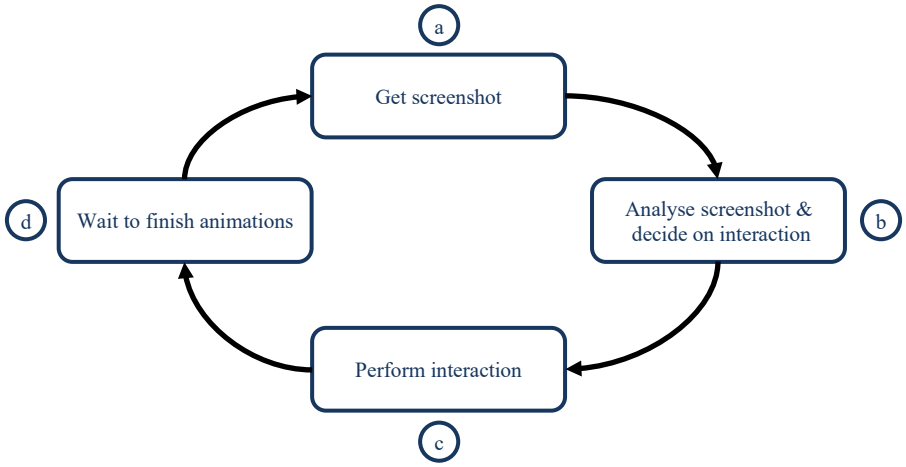
(a) The controlling app requests a screenshot of the controlled app, which the OS provides.

(b) An AI model, managed by the controlling app, analyses the screenshot and decides which actions to perform to achieve the given task.

(c) The controlling app follows the results of the AI model and imitates one user interaction on the controlled app.

(d) The controlling app waits until the controlled app has finished all ongoing animations that followed its interactions to prevent that animations, which change the screen layout, cause the controlling app to e.g. hit the wrong point of the screen. Next, the controlling app goes back to step (a).

**Figure 6. The Control loop on which the Screen Automation feature works.**




*Source: Google's Blog post on the Gemini Computer Use Model.*[1244]

(998) Computer Control offers the user visibility into the automation. When the user taps on an app with which the controlling app is currently interacting with, the user can observe these interactions. The user has always the ability to stop the interaction or take control over the app. At decision points, such as confirming orders or entering payment information, the controlling app will pause automation and send the user a

1242 See Google Git ComputerControlSession.java, https://android.googlesource.com/platform/frameworks/base/+/45034f0663f960d9ee5fb0a101a4732b71f6e2f4/core/java/android/companion/virtual/computercontrol/ComputerControlSession.java, Doc ID […].

1243 See Google Deep Mind (2025), 'Introducing the Gemini 2.5 Computer Use model', Google website, 7 October, https://blog.google/innovation-and-ai/models-and-research/google-deepmind/gemini-computer-use-model/, Doc ID […].

1244 See Introducing the Gemini 2.5 Computer Use model, https://blog.google/innovation-and-ai/models-and-research/google-deepmind/gemini-computer-use-model/, Doc ID […].

notification asking them to confirm and proceed.[1245] While the controlling app interacts with the controlled app, the app icon is marked as inactive.[1246]

(999) The Android 16 implementation in AOSP allowed up to five concurrent Computer Control sessions, i.e. up to five apps can be controlled by another app at the same time.[1247] However, an internal discussion at Alphabet suggests that the implementation [description of Alphabet's internal discussions concerning the implementation of Computer Control]. Indeed, the latest Android 17 release limits Computer Control to only one session at a time.[1248] [Description of internal discussions regarding Computer Control product development].[1249]

(1000) Similarly to App Functions (see Section 10.2.3.2), Computer Control follows an *app caller agnostic* model. That is, an app cannot selectively decide which apps can automate it. However, controlled apps can hide and block certain areas or views from being accessed and controlled using the screen automation feature. If an app uses the window flag *FLAG_SECURE,* Android does not allow to take screenshots or to display the flagged window on a non-secure display. As the Computer Control API allows the controlling app to analyse the controlled app's content through screenshots, this functionality would be blocked.[1250] This mechanism applies equally to all apps that have access to the Computer Control API: if an app uses the *FLAG_SECURE* flag, no app can access and control the flagged parts of the app. This behaviour of the flag has not been publicly documented yet.[1251]

(1001) Computer Control offers several important protection mechanisms:[1252]

(a) Computer Control cannot observe user input outside a Computer Control session. Therefore, an app using Computer Control cannot exfiltrate entered passwords and PIN codes.[1253]

---

1245 See McCullough, M. (2026), 'The Intelligent OS: Making AI agents more helpful for Android apps', Android Developers Blog, 25 February, https://android-developers.googleblog.com/2026/02/the-intelligent-os-making-ai-agents.html, Doc ID […].

1246 Alphabet internal document, Document "Computer Control – High-level challenges" of 23 September 2025, Doc ID […], page 23.

1247 See Google Git on ComputerControlSessionProcessor, https://android.googlesource.com/platform/frameworks/base/+/ccf3bb8e9f8fe6a7767ffb36c61ea20277811cf4/services/companion/java/com/android/server/companion/virtual/computercontrol/ComputerControlSessionProcessor.java, Doc ID [...].

1248 See https://android.googlesource.com/platform/frameworks/base/+/94b4c163b7dfe5ce3607f7bb8456f9573f7de57d/services/companion/java/com/android/server/companion/virtual/computercontrol/ComputerControlSessionProcessor.java#92. Doc ID [...], line 92.

1249 Alphabet internal document, Document [Alphabet internal document], Doc ID […],

1250 Technical meeting Alphabet/Commission of 26 February 2026, Doc ID [...], slide deck 11-17, also orally confirmed by Google employees. Google for Developers, 'Secure sensitive activities | Fraud prevention', Android Developers website, 6 March 2026, https://developer.android.com/security/fraud-prevention/activities, Doc ID [...].

1251 See Android Computer Control, https://developer.android.com/ai/computer-control, Doc ID […].

1252 These protection mechanisms follow from the implementation in Google Android in the open-source AOSP. See AOSP source code on ComputerControlSession.java, https://android.googlesource.com/platform/frameworks/base/+/45034f0663f960d9ee5fb0a101a4732b71f6e2f4/core/java/android/companion/virtual/computercontrol/ComputerControlSession.java, Doc ID […].

1253 The Computer Control API only allows to launch applications, not to interact with OS-surfaces, such as the lock screen, see See AOSP source code on ComputerControlSession.java, https://android.googlesource.com/platform/frameworks/base/+/45034f0663f960d9ee5fb0a101a4732b71f6e2f4/core/java/android/companion/virtual/computercontrol/ComputerControlSession.java, Doc ID […], line 160.

(b) An app can block Computer Control from accessing sensitive areas, such as sensitive data, one-time passwords or transferring money. Therefore, an app that uses Computer Control cannot exfiltrate sensitive data or perform unauthorised payments.[1254]

(c) With Computer Control, the controlling app can only control one app and not operate on the operating system level. Users can always stop any control or automation.[1255]

(d) Computer Control cannot interfere with the current interactions of the user. Therefore, an app using Computer Control cannot stop the user from deleting that app.[1256]

(1002) To implement the feature as a developer, the app must own the *ACCESS_COMPUTER_CONTROL* permission. This is a privileged permission, requiring the OEM to preinstall the app and grant that permission to the app.[1257]

(1003) As a second condition, starting with Android 17, apps that would like to use the Computer Control API also have to be on an Alphabet managed allowlist. An OEM that would like to preinstall an app that uses the Computer Control API would have to request the app to be added to the allowlist.[1258] The allowlist for Computer Control is handled similarly to the allowlist for App Functions. The same [policy regarding Alphabet's commercial agreements with OEMs] will apply for apps that want to be added to the allowlist of App Functions or Computer Control (see Section 10.2.3). Therefore, all third-party developers must pass additional tests and certifications before Alphabet will add their app to the allowlist.

*10.3.4. Feature falls within the scope of Article 6(7) DMA*

(1004) The Commission considers that the screen automation feature as described in Section 10.3.2 – together with its functionalities – is controlled via Google Android and falls within the scope of Article 6(7) DMA.

*10.3.5. Current access for Alphabet's own services and hardware*

(1005) The screen automation feature implemented via the Computer Control API is available to Alphabet, as the APIs underpinning its functionalities, are part of the AOSP source code. As explained in Section 6.2, Alphabet controls which features form part of the AOSP source code and under which conditions those features can be accessed.

(1006) Computer Control is a recent feature that launched in Android 16 on Pixel 10 and Samsung Galaxy S26 devices in the United States of America and South Korea. On these devices, Gemini uses the screen automation feature in beta, only supporting certain food, groceries, and rideshare apps.[1259] Alphabet aims to broaden the screen

---

1254 Google for Developers, 'Secure sensitive activities | Fraud prevention', Android Developers website, 6 March 2026, https://developer.android.com/security/fraud-prevention/activities, Doc ID [...].

1255 Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID […], question 15.

1256 Computer Control runs on a separate virtual display on apps that the user is not using. See AOSP source code on ComputerControlSession.java, https://android.googlesource.com/platform/frameworks/base/+/45034f0663f960d9ee5fb0a101a4732b71f6e2f4/core/java/android/companion/virtual/computercontrol/ComputerControlSession.java, Doc ID […], line 160.

1257 Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID […], question 12.

1258 Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID […], question 12.

1259 See Google (2026), 'Let Gemini handle your multi-step daily tasks on Android', Google website, 25 February, https://blog.google/innovation-and-ai/products/gemini-app/android-multi-step-tasks/, Doc ID […].

automation feature in Android 17 to more users, developers, and OEMs,[1260] including to let the agent control *any* app by that time.[1261] Today Gemini uses Computer Control for the following purposes:

(a) Through Computer Control, Gemini can use food delivery apps, such as Grubhub, to order food from a local restaurant. Gemini opens the app, then searches for the restaurant, adds the food that the user demanded and, in the end, only asks the user to finalise the order, by adding the payment details.[1262]

(b) Through Computer Control, Gemini can also use ridesharing apps, such as Uber, to order a ride for the user. Gemini will open the Uber app, entering the requested destination. Gemini will ask the user to finalise the booking (e.g. selection of the type of car) in the Uber app.[1263]

(1007) However, the same feature became available on devices in the European Union, as of the Pixel 10 update in March 2026[1264] and an update to the Google app includes the required privileged permission to use the screen automation feature.[1265]

(1008) As the provider of Google Android, Alphabet could even ensure that its services, including Gemini, have access to this feature on all Google Android devices, including those where Gemini is not pre-installed, by simply changing the Google Android access rules applicable to the privileged APIs used to implement the feature (see Section 5.2.1). Therefore, the screen automation feature of Google Android is available to Alphabet's services.

*10.3.6. Current access for third-party services and hardware*

(1009) Third-party services do not have equally effective access to the screen automation feature.

(1010) A user-downloaded third-party app does not have access to the implementation of the screen automation feature, being Computer Control, as it is guarded by privileged permissions. For an app to access features with privileged permissions, the OEM must grant the app the permissions upon preinstalling the app onto the device (see Section 5.2). In addition, the app must be added to a Computer Control allowlist by Alphabet.

(1011) At the time of writing, there is no known user-downloaded or preinstalled third-party app that has access to the Computer Control on Android.

---

1260 See McCullough, M. (2026), 'The Intelligent OS: Making AI agents more helpful for Android apps', Android Developers Blog, 25 February, https://android-developers.googleblog.com/2026/02/the-intelligent-os-making-ai-agents.html, Doc ID […].

1261 [Alphabet internal document],

1262 See McCullough, M. (2026), 'The Intelligent OS: Making AI agents more helpful for Android apps', Android Developers Blog, 25 February, https://android-developers.googleblog.com/2026/02/the-intelligent-os-making-ai-agents.html, Doc ID [...].

1263 See Chokkattu, J. (n.d.), 'Gemini Can Now Book You an Uber or Order a DoorDash Meal on Your Phone. Here's How It Works | WIRED', https://www.wired.com/story/google-gemini-task-automation-galaxy-s26-uber-doordash/, Doc ID [...].

1264 "*On Pixel 10 devices, the Gemini app has access to the Computer Control API.*", see Alphabet's reply to PO RFI 11 of 9 March 2026, Doc ID [...], question 44b. See also Levin, M. (2026), 'March Pixel Drop: New personalization and AI tools', Google website, 3 March, https://blog.google/products-and-platforms/devices/pixel/march-2026-pixel-drop/, Doc ID [...], accessed 17 April 2026.

1265 The privileged permission allowlist on a Google Pixel 10 test devices contained the ACCESS_COMPUTER_CONTROL permission granted to the Google app. This permission allows the Google app to control other apps using Computer Control on a device in Europe.

(1012) Third-party developers have access to the public Accessibility Service API in Android OS. The Accessibility Service API offers some of the functionalities of Computer Control. Concretely, apps using this API can observe the screen and interact with it.[1266] However, it still lacks important functionalities that Computer Control offers:

(a) The Accessibility Service API can only control what is visible and what the user could interact with on their own. An app that aims to perform tasks on behalf of the user would then have to perform these tasks in front of the user and cannot run them in the background.

(b) The Accessibility Service API cannot create a virtual display on which the apps could be controlled by the controlling app while the user is using another app.

(c) The Accessibility Service API has no dedicated interface for the user to interrupt an app from interacting with another app. Therefore, the controlling app could perform actions that were not intended by the user and the user would have no way to intervene.

(d) The Accessibility Service API has no limitations in place that could stop an app (accessibility tool) from accessing certain sensitive areas.[1267] For example, a banking app cannot protect sensitive data or functionalities to restrict an app from transferring money without user consent. In contrast, Computer Control API allows an app to flag screens as secure, such that the controlling app cannot view and not interact with them.

(e) The Android OS does not show any privacy indicators when the Accessibility Service API is used. Therefore, the user might be unaware that the Accessibility Service API is controlling the app they are using.

(1013) The Accessibility Service API "should only be used to assist users with disabilities in using Android devices and apps".[1268] Therefore, they require deep access to everything that the user could see or listen to. In contrast, Computer Control was expressly built for apps to automate and control other apps on behalf of the user. Although they have some overlap in functionality, as they have been designed for different purposes, they cannot replace each other, in particular as they differ in the functionalities offered. Alphabet also confirmed that the Accessibility Service APIs "*[…] perform a more limited range of functions, intended for users with disabilities. […] [the accessibility service APIs] would be a completely unsuitable set of APIs to replicate the capabilities of Computer Control*."[1269]

(1014) The Accessibility Service API also does not allow third parties to control apps with the same degree of security and privacy protection as available in Computer Control

---

1266 See Google for Developers (n.d.), 'AccessibilityService | API reference', Android Developers website, https://developer.android.com/reference/android/accessibilityservice/AccessibilityService, Doc ID [...].

1267 The Accessibility Service API distinguishes between accessibility tools (apps that are made for aiding people with disabilities) and other apps using the API to implement other features. An app that declares itself to be an accessibility tool can always access all content of an app. For apps distributed via the Google Play Store, Google imposes additional requirements and verification on such accessibility tools. Developers of controlled apps can restrict other (non-accessibility) apps from using the Accessibility Service API to control (parts of) their app. See Google for Developers (2026), 'Use of the AccessibilityService API - Play Console Help', https://support.google.com/googleplay/android-developer/answer/10964491?hl=en, Doc ID […]. See https://www.esper.io/blog/android-14-accessibility-security-feature, Doc ID [...].

1268 See Google for Developers (n.d.), 'AccessibilityService | API reference', Android Developers website, https://developer.android.com/reference/android/accessibilityservice/AccessibilityService, Doc ID [...].

1269 Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID [...], paragraphs 42-43.

(see Section 10.3.3). Therefore, an app that uses the Accessibility Service API can perform the following malicious actions:

(a) Apps using the Accessibility Service APIs can always observe the entire display of the device. A malicious app could observe passwords, device PINs, and other sensitive information.

(b) Apps using the Accessibility Service APIs can interact with any other app and OS-level surfaces, such as the lock screen or the OS settings. A malicious app could prevent the user from using their device or alter the behaviour of the device without the user noticing.

(c) Apps using the Accessibility Service APIs can interfere with the interactions of the user. A malicious app could prevent the user from deleting the malicious app by constantly redirecting the user to another view or tapping the button the cancel the deletion.

(1015) The Accessibility Service API therefore does not provide effective interoperability for the screen automation feature.

(1016) Even if the Accessibility Service API were to provide effective interoperability, *quod non*, it is not effectively available to all third-party user-downloaded apps, as Alphabet imposes limitations on the distribution of apps that use the API. In particular, the Google Play Store conditions for developers state that *"[a]ny use of the Accessibility API that enables an app to autonomously initiate, plan, and execute actions or decisions is strictly prohibited*".[1270] This prohibition is directly against the purpose of screen automation.

*10.3.7. Alphabet's response to the Preliminary Findings*

(1017) In its response to the PFs, Alphabet argued that the draft measures are unlawful because they exceed the scope of Article 6(7), Alphabet considered that their decision to restrict access to OEM preinstalled apps is justified on the basis of integrity. Alphabet also considered that the draft measures are disproportionate.[1271]

(1018) Alphabet argued that Computer Control is a new and experimental feature that is not to be considered part of the Android OS and is not yet a platform API. Alphabet submitted that this feature is not yet available outside of the United States and South Korea and that the draft measures blur the line between existing OS features and mandatory product design, arguing that the Commission should delay any legal and technical assessment until the point at which it is launched in the EU, rather than imposing measures based on a possible future expansion.[1272]

(1019) Alphabet also argued that Computer Control provides powerful capabilities that could be subject to misuse and drew parallels with the existing capabilities of the Accessibility Service API, which are reported to have been widely abused by malicious actors. Alphabet's position is that the risks associated with Computer Control are more elevated than the Accessibility Service API due to the use of virtual display, through which the controlling app could observe and inject input, such as taps to the app, and is not visible to users. Therefore, Alphabet argues that the limiting access to services via pre-granted OEM permissions is necessary.[1273]

---

1270 See Google for Developers (2026), 'Use of the AccessibilityService API - Play Console Help', https://support.google.com/googleplay/android-developer/answer/10964491?hl=en, Doc ID [...].

1271 Alphabet's response to the PFs, Doc ID [...], paragraphs 283-290.

1272 Alphabet's response to the PFs, Doc ID [...], paragraph 285.

1273 Alphabet's response to the PFs, Doc ID [...], paragraph 287.

(1020) Furthermore, Alphabet raised concerns about the impact of providing concurrent access to Computer Control, as proposed by the draft measures, which Alphabet argued would endanger system health and create a poor user experience. Alphabet described the use of Computer Control as resource-intensive since it requires the OS to simultaneously support actions by (i) the app controlling another app, (ii) the app under control of the controlling app and (iii) the user's continued use of other apps. Alphabet submitted that this would be impossible on devices with memory constraints and these issues would be exacerbated if multiple apps could automate tasks simultaneously. In turn, Alphabet alleges that these issues would create a poor user experience, which would not be resolved through sequential concurrency since users would be unable to use another agent to execute an immediate task.[1274]

(1021) Finally, Alphabet argues that the measures are disproportionate because, in addition to the costs associated with integrity risks created by the proposed measures, the benefits associated with Computer Control are speculative and are only intended to fill the gaps in structured on-device or cloud-based integrations, which are otherwise quicker and more seamless.[1275]

*10.3.8. The Commission's assessment*

10.3.8.1. Concerning scope and integrity

(1022) *First,* with regards to the novel nature of the screen automation feature, the Commission notes that Alphabet is continuing to progress its roll-out of this feature. In May 2026, Alphabet published a guide which provides an overview of the Computer Control framework and technical considerations for target apps and AI assistant apps, noting that the Computer Control API, the API required to implement screen automation, can currently interact with a set of target apps, without any reference to geographic limitations, and expressly stating that Alphabet was planning to expand this support over time.[1276]

(1023) The feature falls under Article 6(7) DMA, as it is already available to Alphabet's services in the EU.

(a) As already explained in Section 10.3.5, Alphabet made the Computer Control API available on its Pixel 10 devices in the EU in March 2026.[1277] Today, the Google app on a Pixel 10 smartphone in the EU possesses the required privileged permissions to use Computer Control.

(b) In response to the PFs, Alphabet wrote that "*the Android CPS consists of the AOSP source code implemented in accordance with the CDD […]*."[1278] The Computer Control API is implemented in the AOSP source code[1279] and, as such, is distributed to Android devices, which implement AOSP in accordance with the CDD.

(1024) Alphabet is not actively using the feature in the EU yet as Alphabet restricts its usage to the USA and South Korea. However, Gemini could start using Computer Control

---

1274 Alphabet's response to the PFs, Doc ID [...], paragraph 287.

1275 Alphabet's response to the PFs, Doc ID […], paragraph 289.

1276 See Android Computer Control, https://developer.android.com/ai/computer-control, Doc ID […].

1277 "*On Pixel 10 devices, the Gemini app has access to the Computer Control API.*", see Alphabet's reply to PO RFI 11 of 9 March 2026, Doc ID […], question 44b.

1278 Alphabet's response to the PFs, Doc ID […], paragraph 158.

1279 See AOSP source code: https://android.googlesource.com/platform/frameworks/base/+/45034f0663f960d9ee5fb0a101a4732b71f6e2f4/core/java/android/companion/virtual/computercontrol/ComputerControlSession.java, Doc ID […].

any day in the EU, as it is already available to Alphabet via the Google app. Therefore, it is a OS feature which "*is available to services or hardware provided by the gatekeeper*" pursuant to Article 6(7) DMA.

(1025) In conclusion, the feature already exists, and Gemini is already performing task automation on several target apps. Therefore, the feature is subject to the interoperability obligation of Article 6(7) DMA. The development of AI services is fast-moving and as the DMA is an *ex ante* regulation, the Commission's measures must capture newly released features to achieve the objective of Article 6(7) and these proceedings, as well as ensure that third-party services can compete on a level playing field with the gatekeeper's services.

(1026) *Second,* Alphabet argues that screen automation could be subject to misuse and, therefore, granting OEMs the right to decide on interoperability would be a strictly necessary and proportionate measure to ensure device integrity.[1280] Alphabet partially puts forward arguments which in fact relate to the protection of the user's security (by addressing misuse by malicious actors) rather than protection of operating system integrity.[1281] The Commission nonetheless addresses Alphabet's arguments in this respect below.

(1027) OEM delegation is not a strictly necessary and proportionate integrity measure, as explained in Section 6.3.1. In addition, preinstallation is not the only measure that is required to use Computer Control API. Starting from Android 17, Alphabet also implemented an allowlist that ensures that only apps on the allowlist can control other apps (see Section 10.3.3). Apps that would want to be on the allowlist would have to ensure that they follow strict security and privacy guidelines and obtain certification.[1282] This measure shows that Alphabet does not trust OEMs to decide which apps should get access to the feature.

(1028) Furthermore, the screen automation feature exhibits several distinguishing capabilities and safeguards that can serve to mitigate the risks that Alphabet raises, as described within Section 10.3.6. For instance, developers can flag an app's window as secure, such that the controlling app cannot view and not interact with the controlled app. Furthermore, users can see if an app is being automated (e.g. through a notification or OS indicators), and users can view what the controlling app is doing and always interrupt any ongoing automation.

(1029) Controlling and accessing other apps is a feature that has been part of the Android OS for almost 17 years.[1283] Through the Accessibility Service API, users with disabilities should be enabled to use their device. Therefore, accessibility services can already access the contents of other apps and control those apps on the user's behalf.

(1030) Alphabet decided to extend these capabilities via the screen automation feature to other apps that should be able to "perform task automation" installed on the user's device.[1284] With Android 17, they are *"looking to broaden these capabilities to reach*

---

1280 Alphabet's response to the PFs, Doc ID […], paragraph 287.

1281 Alphabet's response to the PFs, Doc ID […], paragraph 287.

1282 See Alphabet's response to RFI PO 9, Annex Q1, [Alphabet internal document], Doc ID […], Section 13.2.27, page 24-29.

1283 The Accessibility Service API has been available on Android since Android 1.6, which was released in 2009.

1284 See: Android Computer Control, https://developer.android.com/ai/computer-control, Doc ID [...].

*even more users, developers, and device manufacturers.*"[1285] Therefore, Alphabet decided that their services' capabilities should be extended and use the screen automation feature, which has been expressly designed to let apps automate and control other apps. Article 6(7) DMA then requires Alphabet to provide effective interoperability with this feature for third parties as well.

(1031) Alphabet stated that the risks associated with opening the screen automation feature to third-party apps are demonstrated by the Accessibility Service API.[1286] As lessons learned, Alphabet considers that access to the new screen automation feature, which has similar capabilities, must be restricted. Yet, Alphabet has not chosen to apply similar restrictions to the Accessibility Service API in the recently released Android 17 version. Alphabet admitted that such changes have "a high risk of damaging legitimate applications and commercial models" (see Section 6.3.2). Therefore, all changes made to Accessibility Service API throughout Android updates are more functionality-based restrictions.

(a) Android 17 restricts the use of Accessibility Service APIs when the user enables Advanced Protection mode for apps that are not an accessibility tool, i.e. apps not made to aid people with disabilities.[1287]

(b) Android 14 and 16 allowed developers to mark certain parts of their app as inaccessible for apps using the Accessibility Service API for other purposes than accessibility.[1288]

(c) Android 12 added platform-wide protections against apps that were using the Accessibility API but not declared as tool to aid people with disabilities and Android 13 made it harder for sideloaded apps to obtain access to the Accessibility API.[1289]

(1032) Alphabet explained that any changes to the API could disrupt accessibility tools used by people with disabilities.[1290] Yet, there would be ways to implement further restrictions without causing disruptions, e.g. through grace periods or by automatically granting the required permissions to already installed apps, when the user updates the device. Therefore, it is not clear to the Commission why the feature that grants more extensive access, the Accessibility Service API (see Section 10.3.6), which is also actively exploited,[1291] could be less restricted than a feature that is more restricted by design, and allows users to observe and interrupt any interaction, screen automation; a feature that would also allow third-party assistants to compete on a level playing field with Alphabet's assistant Gemini.

(1033) In any event, due to the sensitivity of this feature, this Decision explicitly allows Alphabet to limit access to screen automation only to third parties that meet certain eligibility criteria, which include criteria relating to integrity, security, and privacy

---

1285 See McCullough, M. (2026), 'The Intelligent OS: Making AI agents more helpful for Android apps', Android Developers Blog, 25 February, https://android-developers.googleblog.com/2026/02/the-intelligent-os-making-ai-agents.html, Doc ID [...].

1286 Alphabet's response to the PFs, Doc ID [...], paragraph 287.

1287 See https://thehackernews.com/2026/03/android-17-blocks-non-accessibility.html, Doc ID [...].

1288 For Android 16 changes see https://android-developers.googleblog.com/2025/12/enhancing-android-security-stop-malware.html, Doc ID [...]. For Android 14 changes, see https://www.esper.io/blog/android-14-accessibility-security-feature, Doc ID [...].

1289 See https://www.esper.io/blog/android-14-accessibility-security-feature, Doc ID [...].

1290 Alphabet's response to RFI PO 11 of 9 March 2026, Doc ID [...], para 107-110.

1291 Expert report of [Academic], Annex 1 of Alphabet's response to the PFs, Doc ID […], paragraphs 17, 22, 23.

(see Section 12.4.4). At the same time, OEM preinstallation does not qualify as the strictly necessary, proportionate integrity measure.

(1034) *Third,* Alphabet raises concerns about the impact on system health of providing concurrent access to Computer Control APIs used for screen automation. The Commission does not see an increased risk to system health on all Android mobile devices, compared to existing Google Android features.

(1035) The current implementation of screen automation generally results in three apps actively running at the same time: the controlling app (e.g. Gemini), the controlled app (e.g. Uber), and the app that the user is currently seeing in the foreground (which could be any app). Concurrent access to Computer Control could result in running additional controlling apps (e.g. other assistants) and controlled apps.

(1036) Fundamentally, modern operating systems are designed to run multiple apps at the same time. Concretely, Android already provides the ability for executing multiple apps in parallel:

(a) Starting in Android 7, released ten years ago, Android introduced a multi-window system. The default implementation is a split-screen option allowing a user to run two apps at the same time on a regular smartphone.[1292]

(b) Android tablets support freeform windows for apps.[1293] The user could perform complex multi-window tasks such as listening to music in one app, browsing the web in another app, and drafting a document in yet another app.

(c) Android 16 introduced support for desktop windowing for smartphones when connected to an external display. This allows a user to connect their Android smartphone to an external display, then connect a keyboard and mouse and use their smartphone like a desktop computer, including by running multiple apps at once.[1294]

(d) Foldable Android phones have a smaller outside screen and a larger folded screen. When unfolded, the screen allows for multitasking running multiple apps at the same time, using a split-screen mode with up to three apps running in parallel or desktop windowing as available to tablets.[1295]

(1037) Android mobile devices today, already support to run multiple apps at the same time, including showing a user interface for each app. An agentic app that would control another app would not require a lot of additional resources, as the main computation will likely happen in the cloud. For example, the Computer Use models from Google, that can be used to perform screen automation, only run in the cloud.[1296]

(1038) Therefore, the screen automation feature should only be limited by the device's hardware. Arbitrarily prohibiting concurrent use of the screen automation feature, which fails to acknowledge that the system may be sufficiently capable to enable multiple apps concurrently, would render interoperability ineffective, as it prevents multi-homing.[1297] However, Android devices vary by hardware resources that are

---

1292 See https://source.android.com/docs/core/display/multi-window, Doc ID […].

1293 See https://developer.android.com/develop/adaptive-apps/guides/support-desktop-windowing, Doc ID […].

1294 See https://developer.android.com/develop/adaptive-apps/guides/support-connected-displays, Doc ID […].

1295 See https://developer.android.com/develop/adaptive-apps/guides/foldables/learn-about-foldables#multitasking, Doc ID […].

1296 See https://ai.google.dev/gemini-api/docs/computer-use, Doc ID […].

1297 See Section 12.5.

available to them, and it may be necessary to limit parallel automation to what can be supported by the device.

(1039) Alphabet also argued that it could lead to a "poor user experience" in case multiple controlling apps seeking to control the same app.[1298] The Commission does not accept a potentially poor user experience as a qualifier that would grant an exemption from effective interoperability. The final user experience in case of conflicts will be implemented by app developers of apps using this feature, who might very well inform the user in a graceful manner that a task can (temporarily) not be processed because an app is unavailable for control. Alphabet cannot possible already identify that the user experience would be poor, and this cannot be a reason to limit third parties from using the feature in the first place.

(1040) Therefore, the Commission considers that the current interoperability solution does not meet the standard of effective interoperability within the meaning of Article 6(7) DMA.

10.3.8.2. Concerning proportionality

(1041) Alphabet claims that the measures for screen automation are disproportionate. However, Alphabet does not substantiate why the measures would be more restrictive than necessary to achieve the objectives of Article 6(7) DMA. In particular, Alphabet fails to propose any less restrictive, yet equally effective means of complying with that provision. Nor does Alphabet explain and substantiate why the feature and/or a specific measure imposes a disproportionate burden on Alphabet. Instead, its arguments focus on questioning the scope of Article 6(7) and the actual need for interoperability – a line of reasoning which does not meet the legal standard for a valid proportionality claim, as explained in Section 13 below.

(1042) The Commission nevertheless addresses Alphabet's argument in this respect.

(1043) The screen automation provides benefits that other integration channels cannot achieve. Computer Control has been developed precisely to plug the gaps left by structured on-device or cloud-based integrations. Alphabet's own guidance acknowledges that developers "*don't need to make any additional changes to integrate a target app with assistant task automation through the Computer Control framework*"[1299] and has directed developers to best practices such as adaptive design and lifecycle management to ensure Alphabet can fully utilise this capability. In turn, this feature significantly lowers the technical barriers developers must overcome to integrate with other apps, allowing caller apps to integrate with a much broader set of apps, which will improve their capabilities for users.

(1044) The Commission therefore concludes that there is an interoperability gap with respect to screen automation feature of Google Android. As a result, third parties and in particular user-downloaded apps, cannot perform important functionalities which places them sub-par in comparison with Google's own AI-powered services. As a result, the Commission considers that it should specify how Alphabet must fill this gap to comply with Article 6(7) DMA.

*10.3.9. Measures that Alphabet should implement*

(1045) To meet the requirements of Article 6(7) DMA, Alphabet should implement the measures set out in Section 3.2 of the Annex to this Decision. The interoperability solution for the screen automation feature, as described in Section 10.3.2 of this

1298 Alphabet's response to the PFs, Doc ID […], paragraph 287.

1299 See Android Computer Control, https://developer.android.com/ai/computer-control, Doc ID […].

Decision, must be equally effective as that available to any of Alphabet's own services.

(1046) As described in Section 12.4.5, the screen automation feature meets the exceptionality conditions of a restricted feature within the meaning of this Decision. Consequently, insofar as the functionalities referenced above pertain to screen automation, access to the functionalities may be subject to eligibility conditions.

(1047) An app that uses the screen automation feature to control other apps must be able to discover which apps are installed on the user's mobile device. Otherwise, the app could not effectively decide which app is best placed to perform the task that the user wanted to perform. For example, if the user asks an assistant to set a reminder, the assistant will check which apps for reminders are installed. If multiple reminder apps are installed, the assistant can confirm with the user which app to use.

(1048) To create a level playing field among competing services, multiple services should be enabled to use the screen automation feature, and this should not be limited to the user's default assistant. Otherwise, a user would be limited in their experience, depending on their default assistant. A user should have the ability to use the full potential of an assistant before choosing to set it as their default.[1300] In addition, a user might use different assistants for different purposes. Therefore, enabling multiple assistants to use the screen automation feature will allow for multi-homing.

(1049) The capabilities of Google Android mobile devices differ. Some devices might not have enough hardware resources to support advanced features, such as screen automation, at all.[1301] Even when screen automation is supported, there may be different limits depending on the devices, e.g. a flagship device might be able automate multiple apps in parallel, while a mid-range device could only automate one app at a time. These limits should be bound by the available hardware resources of the device, and they should not limit that multiple apps can use the screen automation feature sequentially.

(1050) To allow for third-party services to compete with Alphabet's services, such as Gemini, these services must be free to decide which apps they want to control. For example, a service might want to control regionally important ride hailing apps or taxi apps that are not supported by Gemini. It is necessary that third parties can differentiate themselves and support functionalities that are not offered by Gemini. Otherwise, a third-party service would be limited by the innovation of the Gemini assistant. Therefore, third-party services should not be prevented by Alphabet from controlling any app that is installed on the user's device, including Alphabet's own apps, OEM apps and third-party apps.

(1051) However, Alphabet may enable third-party apps to restrict automation of their app for certain parts. This would allow third parties to block controlling apps to access sensitive areas of their apps, e.g. to disallow a controlling app to purchase something or to change the user's password. If developers decide to restrict access to their app,

---

1300 The submission to the public consultation by [Academic] emphasised that the default position can maintain a monopoly, Doc ID […], Section I: "*The search cases found, on the basis of evidence Alphabet itself produced at trial, that this kind of default-driven exclusion is sufficient to maintain a monopoly*."

1301 Today, only the most recent Google Android flagship devices support screen automation, being the Google Pixel 10 series and the Samsung Galaxy S26 series. See McCullough, M. (2026), 'The Intelligent OS: Making AI agents more helpful for Android apps', Android Developers Blog, 25 February, https://android-developers.googleblog.com/2026/02/the-intelligent-os-making-ai-agents.html, Doc ID […].

this restriction shall apply equally across all apps that use the screen automation feature as a controlling app. As the screen automation feature is designed to imitate user behaviour, blocking only some screen automation services while others are allowed would limit the competition among services. In particular, for the reasons explained in Section 4.5.3, if Alphabet allows its own apps to control its first-party apps in any way (via the Computer Control API or any other means), then it should allow third parties to automate the same apps.

(1052) For the purpose of ensuring that effective interoperability continues in the future, Alphabet should also provide effective interoperability with any future updates, including new functionalities, of the screen automation feature insofar as they are available to Alphabet's own services or hardware. To this end, the general measures in Section 12 of this Decision apply, including in particular the measures concerning future updates and new functionalities set out in Section 12.9 in this Decision.

*10.3.10. Timing for implementation of the measures*

(1053) To meet the requirements of Article 6(7) DMA, and taking into account Alphabet's proposal and the available evidence,[1302] Alphabet should implement the measures in the Annex (Section 3.2) according to the timelines indicated therein.

## 10.4. Measures for system integration

*10.4.1. Introduction*

(1054) A modern assistant should be able to integrate seamlessly with the user's device. This integration goes beyond integrating with installed apps and captures the integration with the operating system and its settings. This deep system integration allows an assistant to handle all tasks that the user would demand from its device. For a third-party assistant, these integrations are paramount to compete with the preinstalled Alphabet assistant, which handles these tasks out of the box.

(1055) Alphabet's Gemini assistant has a "Utilities" extension that allows it to accomplish integrations with the operating system, such as changing the display brightness, enabling do not disturb mode, or turning off Bluetooth. Third-party assistants cannot acquire the same capabilities unless they are pre-installed by an OEM, which creates a substantial barrier for third parties.

*10.4.2. Description of the feature*

(1056) The system integration feature includes, amongst others, the functionality to interact with device settings at the user's request, both reading the current state and changing the settings. Such settings include:[1303]

(a) Turning on/off system modes, such as low power, airplane, or do not disturb mode.

(b) (De)activating hardware, such as Bluetooth, Wi-Fi, the flashlight, or NFC.

(c) Interacting with notifications, including reading and replying.

(d) Asking for the current battery level.

---

1302 Technical meeting Alphabet/Commission of 12 June 2026, Doc ID […], slide deck page 13. Alphabet's reply to RFI PO 21 of 8 June 2026, Doc ID […], question 6, paragraph 17.

1303 Technical meeting Alphabet/Commission of 26 February 2026, Doc ID […], slide deck pages 24-25, Alphabet's reply to RFI PO 3 of 13 February 2026, Doc ID […], Annex 1, Doc ID […], page 6, row […]. See https://support.google.com/gemini/answer/15235441, Doc ID […].

(e) Checking and controlling playing media, device volume, and display brightness.

(f) Taking pictures or screenshots.

(g) Managing alarms, timers, and stopwatches.

(h) Discovering installed apps, opening apps, websites, and settings menus.

(i) Powering off or restarting the device.

(j) Integrate with the phone functionalities of the device, if available. This includes making phone calls, accessing the list of recent calls, and accessing contacts.

(k) Integrate with the text messaging functionalities of the device, if available. This includes sending text messages, reading previous text messages.

(1057) To summarise, the functionalities of system-integration feature include, amongst others, the following functionalities:

(a) **Ability to interact with Google Android settings.** These settings include turning on/off Bluetooth, turning on/off battery saver mode, turning on/off do not disturb, reading the current battery level;

(b) **Ability to interact with ongoing activities on the Google Android mobile device.** These activities include controlling playing media and volume;

(c) **Ability to interact with Google Android OS-level functionalities.** These functionalities include restarting the device, turning the device off, taking screenshots, and integrate with notifications; and

(d) **Ability to integrate with common functionalities of Google Android.** These functionalities include setting timers, managing alarms, integrate with text messaging functionalities.

(1058) These interactions can be combined: for example, the user could ask in one prompt to "*turn my media volume and notification volume to [percentage] and turn on battery saver*", which would cause multiple distinct system interactions.[1304]

*10.4.3. Technical implementation of system integration in Google Android*

(1059) Android offers a variety of means for apps to integrate with the system.

(a) Certain integrations use intents, such as for taking a picture, setting an alarm, or turning the device flashlight on or off.[1305] These intents do not require any privileged permissions.

(b) Certain integrations rely on public Google Android APIs, such as for changing device audio levels, connecting to the device camera; or displaying images.[1306] These public OS APIs do not require a privileged permission for an app to use them.

(c) Certain integrations, in particular with OEM apps, are implemented via App Functions (cf. Section 10.2), for example for managing alarms, timers, and

---

1304 See https://support.google.com/gemini/answer/15235441, Doc ID […].

1305 INTENT_ACTION_STILL_IMAGE_CAMERA, SET_ALARM, and TURN_FLASHLIGHT_ON/OFF, respectively. Alphabet's reply to RFI PO 3 of 13 February 2026, Doc ID […], Annex 1, Doc ID […], paragraph.

1306 AudioManager API, CameraManager API, and Bitmap API, respectively. Alphabet's reply to RFI PO 3 of 13 February 2026, Doc ID […], Annex 1, Doc ID […], page 6-7, row […].

stopwatches in OEM clock apps.[1307] [Plans concerning the potential implementation via App Functions of the measures imposed in relation to the system integration feature].[1308] To execute App Functions, the app needs a privileged permission (see Section 10.2.3.2).

(d) Certain integrations, in particular those to more sensitive settings, use Android system APIs. These settings include turning on or off Bluetooth, NFC, Wi-Fi, and low power mode to save battery.

(1060) Most intents and APIs are generally accessible to user-downloaded third-party apps without privileged permissions. Certain APIs require privileged permissions or selection as the default assistant.[1309] App Functions also require privileged permissions, cf. paragraph (942).

*10.4.4. Feature falls within the scope of Article 6(7) DMA*

(1061) The Commission considers that the system integration feature as described in Section 10.4.2 – together with its functionalities – is controlled via Google Android and falls within the scope of Article 6(7) DMA.

*10.4.5. Current access for Alphabet's own services and hardware*

(1062) Gemini uses a "Utilities" extension to integrate with OS functionalities such as controlling device settings.[1310] The functionalities in this extension are based on a wide range of access controls: some are publicly available; some require privileged permissions; some require integration with the settings app; some require Gemini to be set as the default assistant. Alphabet tells users to set Gemini as the default assistant to help the user with actions that require such APIs.[1311]

(1063) Gemini can discover all installed apps and use this information to improve speech recognition. This information allows Gemini to understand user requests better, as it can recognise if the user makes a request towards a specific app.[1312]

*10.4.6. Current access for third-party services and hardware*

(1064) Third parties do not have equally effective access to system integration as Alphabet.

(1065) Some APIs for system integration, such as turning the flashlight on/off or setting do not disturb mode, are available to all user-downloaded apps,[1313] either because they do not require permissions or require permissions that are available to all apps.[1314]

(1066) However, Gemini accesses certain functionalities, such as turning Bluetooth on/off, globally muting the microphone, or setting alarms, through APIs or App Functions that are only available with privileged permissions.[1315] Such APIs are not available to third-party downloaded apps, cf. Section 10.2.3.2.

---

1307 Alphabet's reply to RFI PO 3 of 13 February 2026, Doc ID […], Annex 1, Doc ID […], page 6-7, row […].

1308 Alphabet's reply to RFI PO 9 of 04 March 2026, Doc ID […], question 20d, paragraph 53.

1309 Alphabet's reply to RFI PO 9 of 04 March 2026, Doc ID […], question 21, paragraph 54, question 22, paragraph 56.

1310 Alphabet's reply to RFI PO 3 of 13 February 2026, Doc ID […], Annex 1, Doc ID […], page 6-7, row […].

1311 See https://support.google.com/gemini/answer/15235441, Doc ID […].

1312 Alphabet's reply to RFI PO 15 of 27 March 2026, Doc ID […], paragraph 24.

1313 Alphabet's reply to RFI PO 9 of 04 March 2026, Doc ID […], question 20b, paragraph 50.

1314 Alphabet's reply to RFI PO 9 of 04 March 2026, Doc ID […], question 21, paragraph 54b and 54c(i).

1315 Alphabet's reply to RFI PO 3 of 13 February 2026, Doc ID […], Annex 1, Doc ID […], page 6-7, row […]. Alphabet's reply to RFI PO 9 of 04 March 2026, Doc ID […], question 20c, paragraph 52.

(1067) For certain system integrations, such as setting airplane mode, Gemini relies on integration with an OEM's settings app.[1316] Such integration is at the discretion of the OEM and therefore not available to all third-party downloaded apps.[1317]

(1068) Certain APIs for system integration, such as changing low power mode or turning off the device, are only available to apps that have been selected as the default assistant.[1318] Such APIs are therefore not available to all third-party downloaded apps (cf. Section 5.2.4).

(1069) Alphabet submitted that third-party downloaded apps could "access settings functionalities via intent-based routes, (e.g., prompting the user to open settings pages)".[1319] This effectively means that the user must be brought to settings to configure the desired setting themselves, which is not equally effective to the solution available to Gemini, which can change the setting directly. Such unnecessary switching of apps introduces unwarranted friction into the user journey, which, as explained in Section 4.6, may undermine the effectiveness of interoperability solutions and have an outsized impact on end user behaviour, especially for a service that is meant to assist the end user and reduce their need to do manual tasks.

(1070) Alphabet submitted that third-party user-downloaded assistants would not be able to discover all apps that are installed on the Google Android device. Third-party user-downloaded assistants would only be able to discover if a specific app is installed.[1320] For example, Gemini could check which apps are installed to understand if the user asked Gemini to "open Google" or "open Hoogle". A third-party app could not tune speech recognition towards the installed apps. Third-party apps cannot integrate on an equal level with the system as Alphabet's first-party services.

*10.4.7. Alphabet's response to the Preliminary Findings*

(1071) Alphabet considers that the draft measures are unlawful because they exceed the scope of Article 6(7), insofar as they require the ability to integrate with common functionalities of Google Android. These functionalities, such as timers, alarms, and text messaging, involve separate apps and are not part of the OS or OS-mediated.[1321] Separately, Gemini's integration with OEMs' settings apps, for integrations such as setting airplane mode, is on a bilateral app-to-app basis, not OS-based.[1322]

(1072) Alphabet considers that the draft measures are further unlawful because Alphabet already complies with Article 6(7) DMA. Alphabet states that the system integration feature is available to third parties under equal conditions, which may be default status or OEM preinstallation with permission pre-granting where applicable.[1323] Alphabet states that any further alleged interoperability gap is justified as a strictly

---

1316 Alphabet's reply to RFI PO 9 of 04 March 2026, Doc ID […], question 21, paragraph 54f(i).

1317 Alphabet's reply to RFI PO 9 of 04 March 2026, Doc ID […], question 21, paragraph 54c. "Any app (regardless of whether pre-installed or downloaded) can therefore interact with the OEM's Settings app in the same way if the OEM allows it."

1318 For example, ACTION_VOICE_CONTROL_BATTERY_SAVER_MODE, cf. Alphabet's reply to RFI PO 9 of 04 March 2026, Doc ID […], question 21, paragraph 54e(ii). See also Alphabet's reply to RFI PO 12 of 13 March 2026, Doc ID […], Annex Q4.1, Doc ID […].

1319 Alphabet's reply to RFI PO 9 of 04 March 2026, Doc ID […], question 20b, paragraph 51.

1320 Alphabet's reply to RFI PO 15 of 27 March 2026, Doc ID […], paragraph 24.

1321 Alphabet's response to the PFs, Doc ID […], paragraphs 293-295.

1322 Alphabet's response to the PFs, Doc ID […], paragraph 296.

1323 Alphabet's response to the PFs, Doc ID […], paragraphs 297-298.

necessary and proportionate integrity measure due to the sensitivity of the settings at hand.[1324]

(1073) Alphabet considers that the draft measures are disproportionate, by undermining OEM agency and by being of limited benefit considering the integrity trade-offs.[1325]

*10.4.8. The Commission's assessment*

10.4.8.1. Concerning scope, existing level of interoperability and integrity

(1074) *First,* common functionalities, such as scheduling an alarm and starting a timer, are default features of every smartphone today. The Android OS does not directly implement the alarm or timer as OS features, but the OS mediates the creation, scheduling, cancellation of alarms and timers through well-defined Android intents, in particular through the public *AlarmClock* Android intents.[1326] These intents allow an assistant app or another app to schedule an alarm, set a timer or cancel an alarm in the user's clock app.[1327]

(1075) To implement these features, Alphabet offers a Clock app that OEMs licence through [commercial agreement between Alphabet and OEMs] and that can be installed as [confidential information concerning a commercial agreement between Alphabet and OEMs].[1328] Alternatively, many OEMs implement their own clock apps to add functionalities or provide a design that matches the OEM's design language. Either route support these integrations with common functionalities, e.g. via Android intents.

(1076) [Describes Alphabet's recommendations via commercial agreements on basic actions to be supported by OEMs in their apps, and that should work by default].[1329] The recommendations were made to ensure that Google Assistant always has the ability to execute these commands upon the user's request. The following actions are recommended to be supported by default: call a contact or number, set an alarm, cancel an alarm, set a timer, take a note, take a picture, open a web page, show alarms, and show contacts. Therefore, it can be expected that OEMs preinstall either the GMS Clock app[1330] or build their own clock app that supports these intents.

(1077) This structured on-device integration through app intents is in no way different from App Functions. Equally to App Functions it should be true that a third-party app should be able to use the same intents for their use cases as Alphabet's services. This includes the basic actions that are available on most Android devices by default.

(1078) According to the Commission's knowledge these intents are already available to third-party apps. Therefore, there is no imminent interoperability gap concerning scheduling alarms, setting timers, and other basic actions. The same level of interoperability should continue to exist. If Alphabet should start building custom OS-mediated integrations, through intents, App Functions,[1331] or other means, for

---

1324 Alphabet's response to the PFs, Doc ID […], paragraphs 299-300.

1325 Alphabet's response to the PFs, Doc ID […], paragraph 302.

1326 See https://developer.android.com/reference/android/provider/AlarmClock, Doc ID […].

1327 The AlarmClock intents in Android cover a variety of functionality to schedule, cancel, look-up, dismiss, snooze an alarm and to set, dismiss, and show a timer. See https://developer.android.com/reference/android/provider/AlarmClock, Doc ID […].

1328 See [Alphabet internal document], Alphabet's response to PO RFI 19, Annex Q2.8, Doc ID […], table […].

1329 Alphabet's reply to RFI AI 4 of 28 November 2025, Annex Q3a.13, Doc ID […].

1330 Google's Clock app is installed as a system app and cannot be uninstalled on Pixel devices. See Section 5.2 for information on system apps.

1331 Some functionalities are already available via App Functions, see Section 10.4.5.

basic functionalities these should be made available to third-party apps as well. If the Alphabet's services cannot access these common functionalities of Google Android, e.g. because the OEM did not implement support for these intents, the Commission does not require an implementation that would allow third-party apps to integrate with these functionalities.

(1079) *Second,* with regards to Gemini using bilateral integrations with the OEM's settings app, the Commission assesses that the underlying feature is in the end an OS feature and the integration is mediated through OS APIs. OS features and their functionalities that are available to Alphabet's services should be made available to third-party services.

(1080) The base implementation of the settings app is provided in AOSP. OEMs can modify the app, but they are bound to the GMS requirements in their modifications. For example, the [commercial agreement between Alphabet and OEMs] requirements specify that the settings app must [confidential information concerning the provision in the commercial agreement between Alphabet and OEMs].[1332] Therefore, the settings app is an integral part and a GMS-regulated part of Google Android. In theory, Alphabet could mandate any *bilateral* integrations with its services by adding them in the AOSP settings app and requiring OEMs to keep this integration in their version of the settings app.

(1081) Alphabet made the example of the integration with setting the do-not-disturb mode. Do-not-disturb mode is an OS-level feature that disables notifications when active, allowing the user to not be disturbed. The feature is also available through other APIs, such as the Android notification API. The integration used by Gemini uses the Slice API to integrate with the OEM's settings app.[1333] The Slice API is a public API that allows apps to integrate parts of the user interface from other apps in their app. An integration via the Slice API is OS-mediated, comparable with structured on-device integration via App Functions (see Section 10.2.2) or the Android intents described in this section. The OEM decides if they integrate access via the Slice API into their settings app.

(1082) The current interoperability solution is not equally effective, resulting in an interoperability gap between user-downloaded third-party services and Alphabet's services. As the integrations of common functionalities and the integration via the settings app are OS-mediated, if the OEM grants Gemini access to these functionalities via the settings app, the Slice API, or another integration mechanism, the same functionalities should be available to third-party services. If the same functionalities that are available to Alphabet's services are not available to third-party services, these third parties cannot compete with Alphabet on a level playing field.

(1083) *Third,* Alphabet claims that granting OEMs the right to decide on interoperability and making available certain functionalities only to the default assistant would be a strictly necessary and proportionate integrity measure. Today, third parties cannot access the feature under equal conditions, as Alphabet claimed,[1334] due to the requirement of either OEM preinstallation as discussed in Section 5.4.1, or of the default assistant role, as discussed in Section 5.2.4. OEM preinstallation cannot be

---

1332 Alphabet's reply to RFI AI 2 of 23 September 2025,Annex Q.13.2, Section 6.7 [Commercial agreement between Alphabet and OEMs], Doc ID […].

1333 Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID […] question 21, para 54.

1334 Alphabet's response to the PFs, Doc ID […], paragraph 297-298.

construed as a strictly necessary and proportionate integrity measure either,[1335] as discussed in Section 6.3; neither can the default role be, as discussed in Section 12.5.

(1084) Alphabet correctly remarked that Gemini does not hold the *WRITE_SECURE_SETTINGS* permission,[1336] which allows an app to change many low-level settings of the Google Android device. Alphabet's statement that not all OS settings are available to or used by Gemini is irrelevant as the measures only require to enable interoperability with the OS features that are available to or used by Alphabet's services. A broader, all-encompassing access to all levels of device settings is not required to enable effective interoperability.

(1085) Alphabet submitted that the role-based restrictions, i.e. only allowing the default assistant to access them, are a necessary integrity measure.[1337] The Commission does not object against additional measures that would protect against any downloadable app being able to access such functionalities. However, the features should be available to any app and not be restricted by a role without undue justification (see Section 12.5).

10.4.8.2. Concerning proportionality

(1086) Alphabet claims that the measures for system integration are disproportionate. However, Alphabet does not substantiate why the measures would be more restrictive than necessary to achieve the objectives of Article 6(7) DMA. In particular, Alphabet fails to propose any less restrictive, yet equally effective means of complying with that provision. Nor does Alphabet explain and substantiate why the feature and/or a specific measure imposes a disproportionate burden on Alphabet. Instead, its arguments focus on questioning the scope of Article 6(7) and the actual need for interoperability – a line of reasoning which does not meet the legal standard for a valid proportionality claim, as explained in Section 13 below.

(1087) The Commission nevertheless addresses each of Alphabet's arguments in this respect.

(1088) *First*, Alphabet's contention that access to OEMs' settings feature access would result in a loss of control for OEMs[1338] is addressed in Section 6.1.2.

(1089) *Second*, system integration, such as changing device settings, lies among the core tasks of an on-device assistant, and equivalent access is therefore of high benefit, unlike what Alphabet claims.[1339] To the extent that opening of this feature to third parties would raise any integrity risks, Alphabet is free to implement integrity measures under the framework outlined in Section 4.8.

(1090) Alphabet recognises that user-downloaded apps currently experience "additional friction" in this ability to change device settings.[1340] Such friction is unwarranted, in particular if it is increased compared to Alphabet's services, as explained in Section 10.4.6. Nevertheless, the measures only require access under equal conditions, including in terms of user journeys and friction. Alphabet may still decide on these user journeys, as long as they apply equally to Alphabet's services, including Gemini. Alphabet must decide on the trade-off: if the ability to change device settings is sensitive, then additional steps and user interactions may well be

1335 Alphabet's response to the PFs, Doc ID […], paragraph 299-300.
1336 Alphabet's response to the PFs, Doc ID […], paragraph 300.
1337 Alphabet's response to the PFs, Doc ID […], paragraph 300.
1338 Alphabet's response to the PFs, Doc ID […], paragraph 302, first bullet.
1339 Alphabet's response to the PFs, Doc ID […], paragraph 302, second bullet.
1340 Alphabet's response to the PFs, Doc ID […], paragraph 302, second bullet.

warranted, and would also benefit users of Alphabet's own services; but if Alphabet considers that such additional user friction would be unacceptable for its services, then it would be discriminatory to consider it acceptable for third parties.

(1091) The Commission therefore concludes that there is an interoperability gap with respect to System Integration feature of Google Android. As a result, third parties and in particular user-downloaded apps, cannot perform important functionalities which places them sub-par in comparison with Google's own AI-powered services. As a result, the Commission considers that it should specify how Alphabet must fill this gap to comply with Article 6(7) DMA.

*10.4.9. Measures that Alphabet should implement*

(1092) To meet the requirements of Article 6(7) DMA, Alphabet should implement the measures set out in Section 3.3 of the Annex to this Decision. The interoperability solution for the system integration feature, as described in Section 10.4.2 of this Decision, must be equally effective as that available to any of Alphabet's own services.

(1093) The access to the system integration feature should generally be available to all third-party services. However, as described in Section 12.4.4, depending on the implementation of some functionalities of the interoperable system integration feature can meet the requirements of a restricted feature within the meaning of this Decision. Consequently, access to these functionalities may be subject to eligibility conditions. If functionalities will be implemented using other restricted features, such as structured on-device integration, those will fall under the same eligibility conditions to access them. However, other functionalities, such as the existing public Android intents to set a timer or schedule an alarm, should not be further limited.

(1094) An assistant app can help the user to quickly access other apps by opening an app when the user asks them to "Open Spotify". These are functionalities of the system integration feature of the OS. Such an assistant requires accessing a list of installed apps on the device, because it needs to know which apps it can open and because it needs to tune its speech recognition to detect the names of the apps installed. In addition, the assistant needs to access the relevant information that the system integration functionality requires to open a specific app.

(1095) Any system integration functionalities should be available in the same manner as they are available to Alphabet. This means that a system integration should work the same way und require the same number of steps, no matter if it is used from a third-party service or a first-party service, such as Gemini. Concretely, this means if a user can request that Gemini turns Bluetooth off and Gemini can perform the request without any additional steps, third-party services should be able to do the same. Currently, that integration differs between third-party services and first-party services. Gemini can perform the request to turn Bluetooth off immediately, while third-party services can only open the settings app, then the user would have to turn off Bluetooth manually.

(1096) The Commission updated the measures to remove the requirement that the functionalities of system integration feature should all be implemented via App Functions. As some functionalities are already available via Android intents, it would not be necessary to move these functionalities to App Functions. Therefore, each functionality should be integrated using the mechanism that fits best.

(1097) For the purpose of ensuring that effective interoperability continues in the future, Alphabet should also provide effective interoperability with any future updates,

including new functionalities, of the system integration feature insofar as they are available to Alphabet's own services or hardware. To this end, the general measures in Section 12 of this Decision apply, including in particular the measures concerning future updates and new functionalities set out in Section 12.9 in this Decision.

*10.4.10. Timing for implementation of the measures*

(1098) To meet the requirements of Article 6(7) DMA, Alphabet should implement the measures in the Annex (Section 3.3) according to the timelines indicated therein.

(1099) Alphabet requested an implementation timeline of two years,[1341] as they expect that it would take until then to update their Gemini assistant to use the new interoperability solution. However, Alphabet explained that the implementation of the interoperability solution itself would take one year, and that this solution would already provide equally effective interoperability with all the functionalities of the system integration feature.

(1100) The Commission therefore sees this as a misunderstanding. The timeline only applies to the availability of an equally effective interoperability solution to third parties, and as Alphabet indicated that this is feasible within one year, the Commission has set the timeline accordingly. Alphabet is free to take a longer time to update the Gemini app to adopt the new interoperability solution; in any event, Alphabet is not required to adopt the same interoperability solution for its services (recital (112)), though doing so provides further assurance that Alphabet's services do not have preferential access to the feature.

## 11. MEASURES ON FEATURES FOR ACCESS TO RESOURCES

### 11.1. Introduction

(1101) Adequate access to resources is required for an AI service to execute its tasks with high levels of performance, availability, and responsiveness. This covers the non-discriminatory allocation of hardware resources and runtime for the service and the components it uses, such as on-device models. Without sufficient processing power, these services could become slow or unavailable or could give a worse answer to the user's query. At the same time, AI services may also want to reuse pre-existing on-device resources or components, in particular to leverage the advantages of on-device processing, such as reduced latency and improved privacy.

(1102) Non-discriminatory access to resources is important for third-party services to deliver an experience that is at least as smooth as the experience delivered by first-party services, without being disadvantaged purely due to a lack of resources. If the first-party service has exclusive access to resources that allow it to run more quickly, reliably, continuously, etc., third-party services cannot match or exceed the first-party service's level of performance. Users would not tolerate the sluggish experience of the third-party alternative and would therefore be more likely to switch (back) to the incumbent gatekeeper service.

(1103) Access to resources includes the ability to share on-device models (ODMs) with other apps. AI models compete on functionality such as speech recognition, neural translation, and text-to-speech models, and AI model providers are constantly releasing better, more accurate, and more performant models, including on-device.

---

[1341] Technical meeting Alphabet/Commission of 12 June 2026, Doc ID [...], slide deck page 13. Alphabet's reply to RFI 21 of 8 June 2026, Doc ID [...], question 6, paragraph 17.

(1104) The ability to make ODM functionality centrally available to other services enables competition among ODMs on parameters such as performance, privacy-protections and the offered functionalities. Third parties must therefore be able to offer their on-device models to other apps, with the same ease of access by developers, choice for users, and performance as first-party models. If only first-party models are available to other apps, including centrally, third-party AI model providers cannot compete by offering the same or better (for example, more privacy-preserving) models that are easily accessible to developers and users.

(1105) Based on the assessment developed hereafter, the Commission considers that there is currently an interoperability gap or significant uncertainty as to how Alphabet will implement interoperability when it comes to features for access to resources for third-party services on Google Android mobile devices.

(1106) The Commission therefore specifies how Alphabet must fill this gap, to comply with Article 6(7) DMA. The relevant measures cover access to existing system-level on-device models (Section 11.2) as well the ability to install and use other on-device models (Section 11.3). This combination of interoperability at a high and low level enables effective interoperability in a way that allows third parties fully take advantage of on-device models. Moreover, AI services must have sufficient resources to always be available at the request of users. Next to allocation of hardware resources, the ability to reliably execute the service in the background (Section 11.4) is therefore necessary for effective interoperability.

## 11.2. Measures for system-level on-device models

### *11.2.1. Introduction*

(1107) Recent mobile devices often run machine learning models, including large language models (LLMs), directly on the device hardware ("on-device models" or ODMs). This means that ODMs handle queries, analyse and process data locally instead of via a cloud-based model, where the query and data would be sent to external servers. Using on-device models has several advantages over cloud-based models:

(a) **Lower latency**: ODMs respond faster, as there is no need for the data to go back and forth to cloud servers via the internet.

(b) **Offline availability**: ODMs provide output even if there is no internet connection, while cloud-based models will not work in that case. Alphabet estimates that every day, [5%-15%] of Pixel users spend [25%-35%] of the day without connectivity.[1342]

(c) **Privacy protection**: As the data and the model remain on the mobile device, rather than on the cloud, users have more control over their data.[1343]

(d) **Resilience:** Reducing the dependency on the cloud improves the resilience of the services relying on mobile devices with positive effects on European sovereignty and autonomy.[1344]

(e) **Differentiation (e.g., language support)**: Enabling different ODMs on a mobile device further improves the capabilities of a mobile device as ODMs as are differentiated in many respects (functionality, size, efficiency, speed). One

---

1342 [Alphabet internal document], of May 2025, Doc ID […], page 3.

1343 An Alphabet internal document states that [OEM] uses its [ODM] on [OEM] devices to "handle tasks that require low latency and privacy, without relying on a cloud connection". [Alphabet internal document], Doc ID […], page 2.

1344 Agreed minutes of meeting with [third party] of 2 March 2026, Doc ID [...].

important early example is that the Gemini Nano ODMs may not be optimised for certain languages.[1345] A user in an EU Member State may prefer an ODM optimised for the language of that Member State. ODM differentiation can also enable other innovative use cases.

(1108) Alphabet's Gemini Nano ODMs are preinstalled on virtually all new high-end Android mobile devices. Alphabet provides that Gemini Nano ODM-powered functionality is accessible on 8 Pixel device models, 8 Samsung device models, 8 Xiaomi device models, 9 vivo device models, 9 OPPO device models and five Honor device models, in addition to devices from Lenovo, Motorola, OnePlus, POCO, realme, iQOO, and Sharp.[1346] On Samsung Google Android mobile devices, Samsung's Gauss ODM is also pre-installed.[1347] The Gemini Nano ODMs run within Alphabet's AICore, which is also preinstalled on most new Google Android mobile devices.

(1109) Alphabet's internal documents confirm the importance of ODMs for contestability in AI-based services. Alphabet's internal documents show that [description of internal discussions regarding potential future use of Gemini Nano ODMs for Gemini functionality], for example to support [confidential information regarding the type of functionality to be supported].[1348] The third-party AI service provider [AI service provider] explained that access to certain system-wide ODMs is important.[1349]

(1110) If system-level on-device models installed on the device are exclusively available to first-party services, third-party AI services cannot fully benefit from the advantages that such models bring, especially when their innovation happens by how they use the model and not by the performance of the model itself. While third-party AI services could embed their own on-device models, this increases development cost, which is unnecessary as a system-level model is already available on the end user's device, and requires that the end user downloads and stores an additional, potentially large model on their device, with a risk that the end user abandons that user journey and instead favours the first-party service over the third-party service.

*11.2.2. Description of the feature*

(1111) The system-level on-device model feature (system-level ODM feature) consists in the ability to use and adjust all system-level ODMs. The system-level ODM feature covers the Gemini Nano model family, including its variants and any successor models performing a similar role (including Gemma ODMs), and any other ODMs implemented via AICore or a successor or software performing a similar role to AICore on Google Android mobile devices ("system-level ODMs").

(1112) The functionalities of the system-level ODM feature include, amongst others, the following:

(a) Discovery of system-level ODMs. The ability to discover system system-level ODMs;

---

1345 [Alphabet internal documents describe OEM choices]. See [Alphabet internal document], Doc ID […]. See also Alphabet internal document, email with subject […], Doc ID […], p. 3.

1346 Alphabet lists devices offering Gemini Nano ODM functionality on https://developers.google.com/ml-kit/genai, Doc ID [...], accessed on 8 July 2026.

1347 Alphabet's response to the PFs, Doc ID [...], paragraph 329. According to Alphabet, [OEM] has also deployed an ODM.

1348 [Alphabet internal document], of May 2025, Doc ID […], pages 3, 12.

1349 Agreed minutes of meeting with [AI service provider] of 30 March 2026, Doc ID […], paragraphs 9-10.

(b) Access to system-level ODMs**.** The ability to access and use system-level ODMs and all their functionalities, including specialized APIs for specific tasks; and

(c) Ability to customise functionality. The ability to customise the system-level ODMs, its functionalities.[1350]

(1113) The feature covers the following modalities:

(a) Access subject to the same restrictions. The feature can be used without access restrictions which do not equally apply to Alphabet's services and hardware using the system-level ODM feature (except for the restriction in the next recital).

(b) Use functionalities for all use cases. The feature can be used for all use cases, including those that Alphabet does not (yet) offer and those that Alphabet supports at the system level.

(c) Access conditions. Access to the feature can be used based on transparent, objective, precise, and non-discriminatory rules that equally apply to Alphabet's services, including for use cases that Alphabet does not (yet) offer.

(1114) The feature does not cover functionalities that are available only to internal OS components.

*11.2.3. Technical implementation of system-level ODM in Google Android*

(1115) Alphabet submitted [confidential technical information regarding the implementation of certain Alphabet AI services through AICore][1351] – which is described in Section 11.3.3 – and that AICore and the Gemini Nano ODMs are Google Android middleware.[1352] These models are regularly updated: for example, Gemini Nano is currently at version 3,[1353] with version 4 reportedly being released later in 2026.[1354]

(1116) The Gemini Nano ODMs are implemented on many Google Android mobile devices. They are implemented on Pixel devices since the Pixel 8 generation. The Gemini Nano ODMs are implemented for instance on Alphabet's Pixel 9 and 10 and Samsung S26 devices. Alphabet allows third-party OEM to decide whether to implement AICore and Gemini Nano ODMs in their Google Android mobile devices.

(1117) Alphabet submitted that the Gemini Nano ODMs provide high-level interfaces for use-case-specific APIs, including summarization, proofreading, rewriting, image description, and speech recognition as well as the low-level Prompt API for flexible use cases.[1355] These interfaces are the ML Kit GenAI APIs that expose access to Gemini Nano via AICore.[1356] Information about these APIs is made available to third parties via Alphabet's online developer documentation.[1357] According to Alphabet,

---

1350 One example is LoRA (Low-Rank Adaptation), a parameter-efficient fine-tuning technique for large language models and other deep neural networks.

1351 Alphabet's reply to RFI PO 16 of 01 April 2026, Doc ID [...], paragraphs 48-49, 53-55.

1352 Alphabet's reply to RFI PO 11 of 09 March 2026, Doc ID [...], paragraph 47.

1353 These include the BERT Action model, Intent Classifier, the MT5 model and the gecko model for embeddings. Alphabet's reply to RFI PO 16 of 01 April 2026, Doc ID [...], paragraphs 48-49.

1354 "Google previews Gemini Nano 4 for Android AICore, coming this year", 9to5Google, accessed on 20 April 2026, https://9to5google.com/2026/04/02/gemini-nano-4-android/, Doc ID [...].

1355 See https://developer.android.com/ai/gemini-nano#:~:text=as%20well%20as%20the%20low-level%20Prompt%20API%20for%20flexibility, Doc ID [...], accessed on 20 April 2026.

1356 Alphabet's reply to RFI PO 9 of 04 March 2026, Doc ID [...], paragraph 50.

1357 See https://developers.google.com/ml-kit/genai, Doc ID [...], accessed on 20 April 2026.

the following Gemini Nano ODM functionality is available to third parties via the ML Kit GenAI APIs:[1358]

(a) Summarization: Summarize articles or conversations as a bulleted list.

(b) Proofreading: Proofread short chat messages.

(c) Rewrite: Rewrite short chat messages in different tones or styles.

(d) Image description: Generate a short description of a given image.

(e) Speech Recognition: Transcribe spoken audio to text.

(f) Prompt: Generate text content based on a custom text-only or multimodal prompt. This provides a general interface to Gemini Nano, which can be used for several use cases.

(1118) Alphabet restricts the use of certain system-level ODMs. Alphabet implemented a quota system for apps to use the Gemini Nano ODMs through the ML Kit GenAI APIs (see discussion in Section 11.2.6.2).[1359]

(1119) Alphabet submitted that users cannot replace the Gemini Nano ODMs with another model. Alphabet stated that allowing the user to swap out an ODM would compromise system integrity and undermine app functionality.[1360] According to Alphabet, the experience and reliability of the ML Kit GenAI APIs for developers is fundamentally tied to the model used.[1361]

(1120) Alphabet submitted that its services and hardware cannot modify the ODMs when they call on the ML Kit GenAI APIs.[1362]

(1121) Alphabet submitted that it introduced additional flexibility for OEMs by modularizing AICore, offering three implementation options with different hardware requirements and features to account for the fact that different devices have different capabilities and hardware resources.[1363] Each option offers different features, ranging from "Highest feature depth" to "Majority of existing feature support" to "Lightweight AI capabilities".[1364]

*11.2.4. Feature falls within the scope of Article 6(7) DMA*

(1122) The Commission considers that the system-level ODM feature as described in Section 11.2.2 – together with its functionalities – is controlled via Google Android and falls within the scope of Article 6(7) DMA.

(1123) The system-level ODMs made available to Alphabet services outside the operating system constitute Google Android middleware. As explained in Section 6.2.2.1, these ODMs are designated under the Designation Decision.

*11.2.5. Current access for Alphabet's own services and hardware*

(1124) The system-level ODM feature is used by and available to Alphabet's services and hardware. Alphabet's services can use and adjust system-level ODMs, which are part of Google Android.[1365]

---

1358 See https://developers.google.com/ml-kit/genai, Doc ID [...], accessed on 20 April 2026.
1359 Alphabet's reply to RFI PO 15 of 27 March 2026, Doc ID [...], question 13.c).
1360 Technical meeting Alphabet/Commission of 5 March 2026, Doc ID [...], slide deck page 14.
1361 Alphabet's reply to RFI PO 9 of 04 March 2026, Doc ID [...], paragraph 56.
1362 Alphabet's response to the PFs, Doc ID [...], paragraph 313.
1363 Alphabet's reply to RFI PO 11, response to question 22.
1364 Alphabet's reply to RFI PO 16 of 01 April 2026, Doc ID [...], paragraphs 39-47.
1365 Including Google Android middleware.

(1125) Alphabet's own services use the feature. System-level ODMs are used by Google apps (such as Dialer, Messages, Gboard and Talkback use) and Pixel apps (including Pixel Recorder, Pixel Weather and Pixel Screenshots). They are also used by Alphabet's services via the software components Android System Intelligence and Pixel System Intelligence (for Magic Cue) (see Section 9.3.5).[1366] Alphabet submitted that it uses private APIs to access AICore and Gemini Nano for its own applications (see discussion in Section 11.2.6.1).[1367]

(1126) The system-level ODMs are available to the Gemini service. Alphabet submitted that Gemini is not currently using the Gemini Nano ODMs on Pixel 10 devices.[1368] Alphabet also stated that it currently has no concrete plans to use Gemini Nano ODMs for Gemini on Pixel 11 devices.[1369] This is not supported by Alphabet's internal documents which show that Alphabet is considering using Gemini Nano ODMs for Gemini. [Confidential information from Alphabet internal documents concerning Alphabet's planned use cases, prototypes and Gemini mobile app features using Gemini Nano ODMs].[1370] [1371]

(1127) Alphabet's submissions indicate that it is able to customize system-level ODMs through the use of private APIs. One possibility to customize is through the LoRA (low-rank adaptation) feature (see discussion in Section 11.2.6.1).

*11.2.6. Current access for third-party services and hardware*

(1128) Third parties can access system-wide ODMs through the ML Kit GenAI APIs. Alphabet submits that these APIs are available to its own and to third-party services on equal terms.[1372]

11.2.6.1. Private APIs

(1129) Alphabet submitted that Google makes use of some private APIs to access Gemini Nano functionality for its own applications,[1373] i.e., these applications do not use the ML Kit GenAI APIs. These private APIs are not available to third parties. The use of different APIs by the gatekeeper and third parties makes it possible to discriminate against third parties: the private APIs could offer functionality to Alphabet's services that is not accessible to third parties via the public ML Kit GenAI APIs.

(1130) Alphabet submitted that these private APIs work in a similar way and face similar constraints to the ML Kit GenAI APIs;[1374] there would be no "material" differences.[1375] Alphabet does not explain what the differences are and why they are not material. Alphabet has not published any documentation on how these private APIs work.

---

1366 Alphabet's reply to RFI PO 11 of 09 March 2026, Doc ID [...], paragraph 54.

1367 [Confidential technical information identifying internal APIs used to access system-level on-device models]. See Alphabet, Alphabet's reply to RFI PO 16 of 01 April 2026, Doc ID […], paragraphs 37-38.

1368 Alphabet's reply to RFI PO 16 of 01 April 2026, Doc ID [...], paragraphs 54-55.

1369 Alphabet's reply to RFI PO 16 of 01 April 2026, Doc ID [...], paragraphs 54-55.

1370 [Alphabet internal document], of May 2025, Doc ID […], pages 3, 6, 7, 12, 14.

1371 [Alphabet internal document], Doc ID […], p. 2.

1372 Alphabet's response to the PFs, Doc ID [...], paragraph 312.

1373 [Describes Private APIs available to Gemini Nano]. See Alphabet's reply to RFI PO 16 of 01 April 2026, Doc ID […], paragraphs 37-38; Alphabet's response dated 23 April 2026 to RFI 19, paragraph 35.

1374 Alphabet's response to the PFs, Doc ID [...], paragraph 313; Alphabet's reply to RFI PO 16 of 01 April 2026, Doc ID [...], paragraphs 37-38.

1375 Alphabet's reply to RFI PO 19 of 23 April 2026, Doc ID [...], paragraph 37.

(1131) *First,* Alphabet provided that "certain features such as LoRAs and audio input are not currently available through public APIs" and that "as the AICore product matures, Google is evaluating third-party demand for these features and how that access would be implemented".[1376]

(1132) There are therefore an unknown number of features which are not accessible via the ML Kit GenAI APIs. Alphabet only revealed "certain" such features (namely LoRAs and audio input), not all. The PFs specifically state the private APIs may provide functionality not accessible through the ML Kit GenAI APIs and referred to LoRAs and audio input as features not accessible by third parties.[1377] Instead of clearly stating the differences and providing information on these two features, Alphabet submitted in its response to the PFs that the functionality is "similar".[1378]

(1133) Alphabet provided that it is evaluating "third-party demand for these features".[1379] For the two features not accessible to third parties through the ML Kit GenAI APIs the Commission considers that this may well be the case. Alphabet has publicly reported on the benefits of customizing LLMs through LoRAs.[1380] Similarly, "audio input" could refer to user or ambient data important for AI-powered services.

(1134) It is thus unclear which Alphabet's services, in particular first-party apps, use private APIs and benefit from the additional functionality accessible through the private APIs. Alphabet has also remained vague on which of its services use the private APIs. Asked for an exhaustive list, Alphabet only provided examples (including "Google apps such as the Dialer, Messages, Gboard and Talkback").[1381] Alphabet submitted that "[i]n some cases, Google services have now been adapted to use the ML Kit [GenAI APIs] (such as Gboard)".[1382] This statement is vague. It only provides one example instead of a full list. It does not state that GBoard will use the public ML Kit GenAI APIs, but merely suggests this stating that GBoard has been "adapted" to do so.

(1135) Alphabet provided no details as to if, when, and under what circumstances it would make the features LoRAs and audio input accessible to third parties. It merely provided that "[a]s the AICore product matures, Google is evaluating third-party demand for these features and how that access would be implemented".

(1136) *Second,* Alphabet's claim that the private APIs face "similar constraints" to the ML Kit GenAI APIs[1383] admits that there are differences that might discriminate against third-party services today. Alphabet has not provided any other information on the access conditions applicable to its private APIs, nor stated that such discrimination

---

1376 Alphabet's reply to RFI PO 19 of 23 April 2026, Doc ID [...], paragraph 38.

1377 PFs, paragraph 794.

1378 Alphabet provided that "[i]n a few cases, private system APIs are intended for system functionality not general use (for instance, in connection with the PCC framework)." It is unclear which cases this refers to. Moreover, the "intended" use may not coincide with the actual use. See Alphabet's response to the PFs, Doc ID [...], paragraph 313 4th bullet.

1379 Alphabet's response dated 23 April 2026 to RFI 19, paragraph 38.

1380 See https://developer.android.com/blog/posts/how-automated-prompt-optimization-unlocks-quality-gains-for-ml-kit-s-gen-ai-prompt-api, Doc ID [...] ("Even when needed, more advanced options such as LoRA (Low-Rank Adaptation) fine-tuning can be feasible options."), Alphabet provided this link by email dated 04.06.2026. See also https://ai.google.dev/gemma/docs/core/lora_tuning, Doc ID [...].

1381 Alphabet's reply to RFI PO 11 of 09 March 2026, Doc ID [...], question 20, in combination with Alphabet's reply to RFI PO 19 of 23 April 2026, Doc ID [...], paragraphs 35.

1382 Alphabet's response to the PFs, Doc ID [...], paragraph 313.

1383 Alphabet's response to the PFs, Doc ID [...], paragraph 313.

does not take place, despite the Commission's concerns about difference in access conditions set out in the PFs.[1384]

(1137) There is therefore an interoperability gap with respect to the features LoRAs and audio input. There is also significant uncertainty as a result of Alphabet's vague submissions and incomplete responses to requests for information as to how Alphabet plans to close this gap, whether other features are only accessible to Alphabet's services through private APIs and whether Alphabet's services enjoy advantageous access conditions through private APIs.

11.2.6.2. Non-discriminatory conditions, including quotas

(1138) Alphabet applies certain conditions for access to the ML Kit GenAI APIs. These conditions are opaque for business users. Alphabet does not publish these conditions.

(1139) *First,* it is unclear which access conditions Alphabet sets for use of the ML Kit GenAI APIs by its own and third-party services. The Commission is therefore not able to assess whether Alphabet ensures that transparent, objective, precise, and non-discriminatory rules apply in granting access to the Gemini Nano ODMs that also apply to Alphabet's services, including for use cases that Alphabet does not offer.

(1140) The Commission investigated whether Alphabet sets non-discriminatory conditions for access to the ML Kit GenAI APIs. The Commission asked Alphabet for the "precise rules" which govern access for apps to the ML Kit GenAI APIs[1385] and the limitations which Alphabet imposes on third-party apps to make use of Gemini Nano models such as "*limit*[ing] *the number of prompts, or the duration of use for a certain period of time*".[1386] Alphabet submitted that it implemented a quota system for apps to use the Gemini Nano ODMs through the ML Kit GenAI APIs to avoid excessive use of system resources and preserve system health.[1387]

(1141) Alphabet did not produce the rules and limitations that govern access. It explained that "at present AICore has both a daily quota and a time-related quota on frequent calls" and "[i]*f an app exceeds the quotas, it will receive an error code*".[1388] Alphabet referred the Commission to the public Google documentation for developers,[1389] which merely states that "AICore enforces an inference quota per app", that "[m]a*king too many GenAI API requests in a short period will result in an ErrorCode.BUSY response*" and that an error code also "*can be returned if an app exceeds a long-duration quota (e.g. daily quota)*".[1390] In response to the PFs, Alphabet does not provide any additional detail on what quotas it sets. Alphabet only submitted that the quotas imposed on apps using the ML Kit GenAI APIs apply to all apps, including Google's own and that it does not publish precise quotas because

---

1384 PFs, paragraph 796.

1385 Alphabet's reply to RFI PO 19 of 23 April 2026, Doc ID [...], question 13.

1386 Alphabet's reply to RFI PO 15 of 27 March 2026, Doc ID [...], question 13.

1387 Alphabet's reply to RFI PO 15 of 27 March 2026, Doc ID [...], question 13.c); Alphabet's response to the PFs, Doc ID [...], paragraph 317.

1388 Alphabet's reply to RFI PO 19 of 23 April 2026, Doc ID [...], paragraphs 32-33.

1389 Alphabet's reply to RFI PO 19 of 23 April 2026, Doc ID [...], paragraph 32. Alphabet also referred the Commission to the Google APIs Terms of Service and the Generative AI Prohibited Use Policy (see paragraph 31 which refers to Alphabet's reply to RFI PO 15 of 27 March 2026, Doc ID [...], fn. 11. These policies accessible at https://developers.google.com/terms#a_api_prohibitions, Doc ID [...], accessed on 9 July 2026 and https://policies.google.com/terms/generative-ai/use-policy, Doc ID [...], accessed on 9 July 2026, do not contain the precise rules. The first policy provides with respect to any Google API that "Google sets and enforces limits on your use of the APIs (e.g. limiting the number of API requests that you may make or the number of users you may serve), in our sole discretion."

1390 See https://developers.google.com/ml-kit/genai, Doc ID […], accessed on 24 April 2026.

they would rapidly become stale as the feature is complex, developing and device specific.[1391]

(1142) *Second,* the Commission considers that Alphabet might grant third-party services which use the ML Kit GenAI APIs more limiting access conditions (including inference quota) than its own services via private APIs (see Section 11.2.6.1).

(1143) *Third,* Alphabet is able to disadvantage specific apps in setting access conditions for the ML Kit GenAI APIs and could use that power to disadvantage third-party apps.

(1144) Alphabet submitted that it is possible to ban a specific app and that it could make use of that possibility when an app misbehaves (e.g., due to bugs or unexpected behaviour), such that its usage needs to temporarily be suspended until the issue is resolved.[1392]

(1145) Alphabet may also use this and similar powers to discriminate against third-party services more generally and not related to system health reasons. According to an internal document, Alphabet considered to discriminate against third-party apps in general to advantage its own apps and that features in third-party apps relying on the Gemini Nano ODMs would be [significantly worsened] in terms of predictability and latency. [The document provides that Alphabet and OEM apps could be prioritized regarding the use of AICore/ Gemini Nano ODMs when an app is not in the foreground. Alphabet anticipates that this would significantly affect the predictability and latency of third parties using AICore] ."[1393] This document does not concern guarding against specific apps causing system health issues, but Alphabet's powers over AICore/Gemini Nano ODMs if [OEM] were to implement its own ODMs.

(1146) Alphabet submitted that the PFs "wrongly suggest that Google considered discriminating against third-party apps in Gemini Nano calls" and that this rests on a misunderstanding of the word [reference to wording used in Alphabet internal document] in the internal document referenced in the previous paragraph.[1394] However, even if this was correct, it would not call into question the conclusions drawn from the other statements presented in the previous paragraph from the same document, including that third-party access to features would be less predictable and slower and have lower priority when the app is running in the background.

(1147) Alphabet submitted that it granted OEMs the power to decide about quotas because an ODM runs on device hardware.[1395] The same Alphabet internal document indicates that even if [Alphabet internal document discussing the impact of allotted quotas on the usage of an app]. This indicates that Alphabet retains the ability to set quotas for specific apps for third-party OEM mobile devices.

(1148) *Fourth,* the Commission considers that Alphabet may not ensure that third-party OEMs set transparent, objective, precise, and non-discriminatory rules in granting access to the Gemini Nano ODMs, including for use cases that Alphabet does not offer.

---

1391 Alphabet's response to the PFs, Doc ID […], paragraph 313.

1392 Alphabet's response to the PFs, Doc ID […], paragraph 317.

1393 [Alphabet internal document], 15 March 2025, Doc ID […], page 7. [Alphabet internal document discusses applicability of quotas]. It provides that [Alphabet internal document discusses configurable AICore usage quotas].

1394 Alphabet's response to the PFs, Doc ID […], paragraph 313.

1395 Alphabet's response to the PFs, Doc ID […], paragraph 313.

(1149) Alphabet submitted that it granted OEMs to decide about quotas because an ODM runs on device hardware. Further, every OEM's hardware has different characteristics and levels of performance that the OEM understands best.[1396]

(1150) As a result, third-party OEMs could set quotas or other rules in a manner which is not transparent, objective, precise, and non-discriminatory in granting access to the Gemini Nano ODMs, including for use cases that Alphabet does not offer. This does not apply to Alphabet's own Google Android mobile devices as Alphabet controls how these rules are set.

(1151) Alphabet did not contest this possibility in its response to the PFs nor explain why third-party OEMs would be prevented from doing so.

(1152) Therefore, the Commission cannot assess whether the access rules are based on transparent, objective, precise, and non-discriminatory rules. Moreover, there is a risk that Alphabet or third-party OEMs could use the access rules to advantage Alphabet's services over third-party services.

11.2.6.3. Technological changes

(1153) ODMs, as well as LLMs and AI capabilities more broadly, are an evolving area subject to rapid technological change which is of high commercial importance for several Alphabet's and third-party services. Alphabet submitted that AICore/Gemini Nano ODMs is a complex and developing feature.[1397] Alphabet also submitted that the recently launched [reference to AICore features] offers different features sets and that it remains to be determined which features are included and which Google services can use these models.[1398] There is therefore a high potential for future developments and changes. Compared to mature technology and features, it is therefore more likely that the degree of interoperability with respect to AICore/Gemini Nano ODMs existing today could change in the future.

(1154) The Commission considers that Alphabet's use of private APIs could result in certain functionalities being only available to Alphabet's services. For instance, the ML Kit GenAI APIs offer a generic Prompt API, through which third parties can issue "a custom text-only or multimodal prompt" to Gemini Nano, which returns "text content". Private APIs may also allow richer prompts or enable that multimedia content (e.g. images) is returned in addition to text.

(1155) Recent product announcements attest to the high level of development. The next version of Gemini Nano (version 4) will have multiple variants, including "Fast" and "Full", with different speed and reasoning power.[1399] It is unclear how Alphabet will make these new versions interoperable. As explained above, Alphabet is able to grant its own services preferential (including exclusive) access to some of these variants through private APIs or setting discriminatory access conditions.

(1156) Alphabet submitted that the Commission frequently speculates about what Google "could do" rather than what it does in alleging a gap in interoperability. It further submitted that there is no interoperability gap and that it is disproportionate to impose measures based on unevidenced postulation about what might happen in the

---

1396 Alphabet's response to the PFs, Doc ID […], paragraph 313.

1397 Alphabet's response to the PFs, Doc ID […], paragraph 313.

1398 Alphabet's reply to RFI PO 16 of 01 April 2026, Doc ID […], paragraphs 41-42.

1399 "Google previews Gemini Nano 4 for Android AICore, coming this year", 9to5Google, accessed on 20 April 2026, https://9to5google.com/2026/04/02/gemini-nano-4-android/, Doc ID […]. Technical meeting Alphabet/Commission of 5 March 2026, Doc ID […], slide deck page 7.

future, including speculation about Google using private APIs in future to allow richer prompts or multimedia content.[1400]

(1157) Sections 11.2.6.1 and 11.2.6.2 demonstrate the existence of interoperability gaps and the risk of further interoperability gaps in the future, including because Alphabet did not provide sufficient details on the access conditions. The scenarios provided above illustrate the pace of technological development. In its response to the PFs, Alphabet did not contest that these scenarios are plausible and relevant.[1401]

(1158) As regards private APIs, Alphabet has remained vague about the extent to which it will use private APIs in the future. Functionality which is only accessible via private APIs (see 11.2.6.1) and the use of private APIs enables gatekeepers to discriminate against third-party services as regards feature accessibility. Moreover, there is stark information asymmetry because only Alphabet knows the functionality accessible through the private APIs so interoperability gaps may not be observable. The lack of observability also applies to the access conditions. Due to this information asymmetry and lack of observability third parties may not be able to identify interoperability gap or do so only with delay.[1402]

(1159) Article 8(2) DMA is not limited to cases of existing interoperability gaps. Such a reading is incompatible with the ex-ante nature of the DMA as it risks that a gatekeeper implements a new feature in a way that does not effectively comply with Article 6(7) DMA in the first place. Third parties or the Commission subsequently have to identify this gap and the Commission possibly needs to initiate proceedings thereafter, which take time and bind resources. The gatekeeper may then have to implement an interoperability solution. As illustrated by Alphabet's proposed timelines in these proceedings to close existing interoperability gaps,[1403] this could result in lengthy delays in interoperability for third-party services. Such delays are incompatible with effective compliance.

*11.2.7. Commission's assessment of Alphabet's response to the Preliminary Findings*

(1160) Alphabet submitted that the draft measures annexed to the PFs in relation to the on-device model implementation feature are unlawful and disproportionate.

11.2.7.1. Concerning designation, scope, existing level of interoperability and integrity

(1161) *In the first place,* Alphabet submitted that Gemini Nano and other ODMs are not designated as part of the Android CPS.[1404]

(1162) Alphabet's first claim that Gemini Nano is not designated is addressed in Section 6.2.2.1.

(1163) *In the second place,* Alphabet submitted that the definition of "system-level ODM" is overbroad. While "AICore is designed as a general-purpose interface for other apps to access centralised model functionality", other ODMs are "simply internal software features which are "not designed to be general-purpose interfaces for other apps".[1405]

---

1400 Alphabet's response to the PFs, Doc ID […], paragraphs 314-315.

1401 Alphabet's response to the PFs, Doc ID […], paragraphs 314-315.

1402 Alphabet submitted that [describes Alphabet's position regarding private APIs] (response to the PFs, paragraph 312). Third parties have no visibility into the private APIs that Alphabet's service use. As explained in Section 11.3.6, even OEMs may lack visibility into the specific setup adopted by Alphabet.

1403 Alphabet's reply to RFI PO 21 of 8 June 2026, Doc ID […], question 6, paragraph 17.

1404 Alphabet's response to the PFs, Doc ID […], paragraphs 306, 310.

1405 Alphabet's response to the PFs, Doc ID […], paragraphs 304, 307-309.

(1164) In response to Alphabet's submission, the Commission limited the feature to system-level ODMs as defined in Section 11.2.2 because Alphabet explained that certain other ODMs are customised to bespoke requirements of the operating system.[1406]

(1165) *In the third place*, Alphabet submitted that it already complies with Article 6(7) DMA in respect of the system-level ODM feature because it already provides interoperability with the Gemini Nano ODMs via the ML Kit GenAI APIs which are available to first-party and third-party services on equal terms. According to Alphabet, the use of quotas, private APIs and customization would not amount to an interoperability gap. Moreover, Alphabet submitted that the Commission speculates about what Alphabet could do rather than might do in the future instead of demonstrating a current interoperability gap.[1407]

(1166) The Commission addresses these arguments in Section 11.2.6.

(1167) *In the fourth place,* Alphabet submitted that quotas, use case restrictions and limitations on customisation through LoRAs are covered by the integrity justification.[1408]

(1168) As regards quotas, the Commission accepts that the quota system in AICore can contribute to preserving system health in the resource-constrained environment of a smartphone.[1409]

(1169) The measures in this Decision allow Alphabet to implement a quota system to preserve system health provided that this system is based on transparent, objective, precise, and non-discriminatory rules that also apply to Alphabet's services, including for use cases that Alphabet does not offer.

(1170) As regards use case restrictions, Alphabet must ensure that it complies with applicable law when taking steps to comply with the DMA and this Decision.[1410] In particular, under the AI Act, providers of general-purpose AI models with systemic risk shall mitigate such risks which include actual or reasonably foreseeable negative effects on public health, safety, public security, fundamental rights, or the society as a whole.[1411] This may cover, depending on the circumstances, several examples of harmful uses raised by Alphabet, such as terrorism, illegal gambling and certain uses that may harm the rights of others. Alphabet submitted that it should be allowed to also prevent "unethical" uses of AI. The Commission considers that such a broad remit is not provided for in Article 6(7) or the DMA more broadly. The Commission considers that this is not sufficiently specific.

(1171) As regards LoRAs, Alphabet submitted that not allowing third parties to customize the AICore/Gemini Nano ODMs through LoRAs is justified for integrity reasons. Further, allowing third parties to customize through LoRAs "*would run into severe practical and technical difficulties*", and Google is not able to provide third parties with certain information from chipset vendors as this information is "*in some cases proprietary*".[1412] Alphabet did not provide a sufficient justification for this integrity measure. First, Alphabet has not explained what the feature consists in and which of its services make use of it (see Section 11.2.6.1). Alphabet's claim of a "wide-spread

---

1406 The Commission does not take a view whether these other ODMs are subject to Article 6(7) DMA or are important for contestability, fairness, or innovation for third-party app developers.

1407 Alphabet's response to the PFs, Doc ID […], paragraphs 304, 310-316.

1408 Alphabet's response to the PFs, Doc ID […], paragraph 317.

1409 Alphabet's response to the PFs, Doc ID […], paragraph 317.

1410 See Section 4.8.2.

1411 See Articles 55(1)(b) and 3(65) of Regulation (EU) 2024/1689.

1412 Alphabet's response to the PFs, Doc ID […], paragraph 317.

move to develop alternatives to LoRAs for both first-party and third-party apps" is unsubstantiated. It is short (one paragraph) and vague ("severe practical and technical difficulties"; "in some cases"). It is unclear which first-party apps are affected, when such development would be concluded, and when Alphabet's first-party apps would no longer use the feature. The importance of the feature is therefore unclear. Second, it is not clear how feature access for third parties would compromise the integrity of the operating system and in relation to which part of the operating system.

11.2.7.2. Concerning proportionality

(1172) *In the first place*, Alphabet submitted that the measures envisaged by the Commission are disproportionate in the circumstances of Google and the Android OS because they would harm end users, OEMs, business users and Google and would risk undermining the ability to manage system health, protect device integrity and address risks of harm through misuse of AI.[1413]

(1173) Alphabet does not substantiate why the measures would be more restrictive than necessary to achieve the objectives of Article 6(7) DMA. In particular, Alphabet fails to propose any less restrictive, yet equally effective means of complying with that provision (see Section 13 below). Nor does Alphabet substantiate why a specific measures imposes a disproportionate burden on it.

(1174) The Commission addressed these claims in Sections 6.3 and 6.1.2 above. It is unclear from Alphabet's short argument (one sentence) why the measures would be disproportionate "*in the circumstances of Google and the Android OS*". The Commission is not aware of a less restrictive and equally effective means to achieve interoperability pursuant to Article 6(7) DMA.

(1175) *In the second place*, Alphabet submitted that interoperability for the system-level ODM feature is already provided at the level of the OS features it uses (i.e., the ODM implementation feature in Section 11.3) and that further access would be disproportionate.[1414]

(1176) As a starting point, both features are part of Google Android and therefore in scope of Article 6(7) DMA. As explained (Section 4.2), interoperability with different features of the operating system can be required and proportionate as these features provide the building blocks for apps to achieve the goals of that provision, namely contestability, fairness and innovation.

(1177) Interoperability with both features enables contestability and innovation in different, important, and complementary ways.

(1178) Alphabet recognizes these benefits for third parties and the Android ecosystem by allowing third parties interoperability with both features to a significant extent already today. Alphabet has developed specific APIs and made them accessible to all developers to enable them to build feature-rich apps and implement their own ODMs on Google Android mobile devices. Alphabet did so at least in part due to pressure from OEMs seeking to implement their own ODMs in addition to the Gemini Nano ODMs (see Sections 11.2.6 and 11.3.6).

---

1413 Alphabet's response to the PFs, Doc ID […], paragraph 318. Alphabet's claim on the scope of the definition of "system-level ODMs" is addressed in this section as the first item in the list. Alphabet's claims that the problems identified by the Commission are largely speculation is addressed in Section 11.2.6.3.

1414 Alphabet's response to the PFs, Doc ID […], paragraph 304.

(1179) Interoperability with the system-level ODM feature enables third-party apps to use ODMs in the same way as Alphabet's services to offer feature-rich apps. It therefore creates a level playing field between e.g. the Alphabet messages and a third-party messaging app or between Gemini and ChatGPT, both of which could use these ODMs to offer innovative functionality. If this functionality was reserved to Alphabet's services, contestability and innovation would be undermined. Alphabet's services make use of the system-level ODM feature. Alphabet points to advantages such as enhanced app functionality, improved user privacy, ease of deployment (no need to download or update models and take additional device storage space/memory), and integration with each device's underlying hardware interfaces.[1415] As explained in Section 6.2.2.1, the feature contributes to providing a basic function of the operating system to apps, namely more advanced text creation and editing, including speech recognition, proofreading, word prediction and text suggestion.

(1180) Continued interoperability with the system-level ODM is important for contestability. Developers rely on the Gemini Nano ODMs based on Alphabet's decision to include and market them as Google Android features. The inclusion of the Gemini Nano ODMs in Google Android creates an expectation of their accessibility on Google Android mobile devices (see Section 6.2.1). Developers create apps which rely on the ML Kit GenAI APIs. Through their inclusion in Google Android, these APIs have become the standard for the functionalities they provide. Developers cannot simply switch away from these APIs because using other ODMs may undermine app functionality,[1416] requires costly development of own ODMs or access to third-party ODMs, and assurances that these ODMs are accessible on as many Google Android mobile devices as possible. In sum, developers would need to forego the structural advantages which having access to operating system features entails.

(1181) Interoperability with the ODM implementation feature enables third parties to implement their own ODMs on Google Android devices. It therefore creates a level playing field in the implementation of ODMs, e.g., the Google Nano ODMs and Microsoft's Phi or Samsung's Gauss. As explained (see Section 11.3.1), different ODMs have different strengths, such as size, speed, accuracy, or language support. For example, on-device models can go beyond just text and can be capable of ingesting and outputting images, sound, or video.

(1182) Continued interoperability with the ODM implementation feature enables contestability and innovation in these areas. Third-party ODMs can contest Alphabet's ODMs in the areas of differentiation and innovation listed in the previous recital. They may also improve on functionality of the Gemini Nano ODMs in the areas of summarization, proofreading, rewriting, image description, speech recognition and prompting.[1417] [With respect to an OEM implementing] its own ODM on Google Android mobile devices due to better language coverage and quality.[1418] As LLMs and ODMs are a new and evolving technology harbouring great potential for different use cases, innovation is particularly important in this area and third parties should not be locked into choices taken by a gatekeeper. Several

---

1415 See https://developer.android.com/ai/gemini-nano, Doc ID […], accessed on 20 April 2026; https://developers.google.com/ml-kit/genai, Doc ID […], accessed on 20 April 2026.

1416 See technical meeting Alphabet/Commission of 5 March 2026, Doc ID […], slide deck page 14.

1417 See Section 11.2.2.

1418 See Section 11.3.1.

third parties highlighted the importance of interoperability for ODMs for that reason.[1419]

(1183) As regards Alphabet's argument that intra-OS components should not be caught by Article 6(7),[1420] as explained in Section 4.2, depending on the situation, several related features (even those at different "layers") may need to be made interoperable to meet the objectives of the DMA and ensure effective interoperability in individual situations.

(1184) Moreover, at least on Pixel devices, interoperability with both features is critical to account for the fact that Alphabet's design choices prevent third parties from implementing ODMs on a level playing field with Alphabet's AICore. Alphabet recognizes that interoperability with AICore/Gemini Nano ODMs compensates for restricted interoperability as regards the ODM implementation feature. Alphabet therefore accepts the necessity of interoperability with the system-wide ODM feature (see Section 11.3.7).

(1185) As a result, the Commission considers that specifying measures both for the system-wide ODM feature and the ODM implementation features is covered by Article 6(7).

(1186) *In the third place*, Alphabet submitted that it developed the Gemini Nano ODMs at great expense and forced interoperability amounts to expropriation of Google's property. Article 6(7) does not extend into a broad mandate for compelled service provision. Alphabet cited [third party's] submission to the public consultation which referred to the costs of development of a gatekeeper's on-device models and stated that mandatory access to them would pose a direct threat to competition and undermine incentives to introduce new innovations in the EU.[1421]

(1187) The Commission considers that specifying access to system-wide ODMs is covered by Article 6(7) DMA for the reasons set out under point two above, in particular that such access is required for third-party services to have equally effective access to operating system features as Alphabet's services. This conclusion is not called into question by the Alphabet's argument for the following reasons.

(1188) *First,* Alphabet voluntarily exposed the Gemini Nano ODMs to third parties and presents them as part of the Google Android operating system (see Section 6.2.2.1). Alphabet presents these ODMs to developers as "*the foundation model on Android*" which are "*directly integrated to the Android OS via AICore*."[1422] As operating system features, the Gemini Nano ODMs are interoperable through the ML Kit GenAI APIs.[1423]

(1189) Alphabet made the Gemini Nano ODMs interoperable despite the feature's development costs. This decision indicates that interoperability of the Gemini Nano ODMs is consistent with Alphabet's incentives, including monetary incentives.

(1190) *Second,* Alphabet's claim of expropriation is inconsistent with its claim that Article 6(7) DMA "requires interoperability with OS-mediated features", but not with "server features". Alphabet argues that the "logic of this interpretation is straightforward: OS mediation does not impose extra costs on the OS provider, as it draws on the user's device resources." Alphabet considers that "[f]ree provision is therefore proportionate" and distinguishes features on the device from "server

1419 See Section 11.2.1.
1420 Alphabet's response to the PFs, Doc ID […], paragraph 309.
1421 Alphabet's response to the PFs, Doc ID […], paragraph 311.
1422 See Section 6.2.2.1.
1423 See Section 11.2.3.

features" which entails disproportionate "infrastructure costs".[1424] Alphabet therefore considers interoperability with on-device resources proportionate, and ODMs are by definition on-device resources.[1425]

(1191) *Third,* in any case, Alphabet does not substantiate the claim of expropriation that would violate Article 17 of the Charter.[1426] In principle, the Union legislator has already weighed the public interests against the private interests of economic operators impacted by the DMA, including the protection of property rights.

(1192) The Commission therefore concludes that there is an interoperability gap with respect to the system-level ODM feature of Google Android and uncertainty as to whether the feature will be fully accessible to third-party apps in the future. As a result, the Commission considers that it should specify how Alphabet must fill this gap and remove uncertainty to comply with Article 6(7) DMA.

*11.2.8. Measures that Alphabet should implement*

(1193) To meet the requirements of Article 6(7) DMA, Alphabet should implement the measures set out in Section 4.1 of the Annex to this Decision. The interoperability solution for the system-level ODM feature, as described in Section 11.2.2 of this Decision) must be equally effective as the solution available to Alphabet's own services.

(1194) This means that Alphabet should provide third parties access to the functionalities available to Alphabet's own services or hardware which enable Alphabet to discover system-level ODMs, to access and use these ODMs and all their functionalities, and to customize these ODMs, its functionalities and output.

(1195) In addition, to ensure effective interoperability for third parties, Alphabet should grant third parties access to additional functionalities available to Alphabet's services if necessary to enable effective interoperability with the system-level ODM feature. Absent such access, third parties would not be able to compete on a level playing field.

(1196) Alphabet should provide third parties with an interoperability solution for the system-level ODM feature that is equally effective as that available to Alphabet's services or hardware. Alphabet should therefore grant access to third parties without restrictions which do not equally apply to Alphabet's own services and hardware, allow third-party services and hardware to use the features for all use cases, and ensure access to the feature according to transparent, objective, precise, and non-discriminatory rules that also apply to Alphabet's services, including for use cases that Alphabet does not offer.

(1197) Alphabet is not required to make accessible to third parties functionalities that are available only to internal operating system components, including the ability to customize the system-level ODMs. This reflects Alphabet's submission that certain system-level functionality uses AICore/Gemini Nano ODMs and that such system-level functionality may have specific requirements.[1427]

(1198) Alphabet should provide effective interoperability with future updates, including new functionalities, of the system-level ODM feature insofar as they are available to

---

1424 Alphabet's response to the PFs, Doc ID […], paragraph 66.

1425 This also applies to the features that Alphabet currently only makes accessible to its own services. Alphabet submitted that it "is evaluating third-party demand for these features and how that access would be implemented", see Alphabet's response dated 23 April 2026 to RFI 19, paragraph 38.

1426 OJ C 326, 26.10.2012, pages 391–407.

1427 Alphabet's response to the PFs, Doc ID […], paragraphs 331, 327.

Alphabet's own services or hardware. This ensures that the measures remain relevant in the future. Such updates or functionalities may include new system-level ODMs in AICore or a successor or similar Alphabet solution to AICore, such as those related to image or video creation or updates to existing system-level ODMs such as Gemini Nano ODMs on-device models. To this end, the general measures in Section 12 of this Decision apply, including in particular the measures concerning future updates and new functionalities set out in Section 12.9 in this Decision.

*11.2.9. Timing for implementation of the measures*

(1199) To meet the requirements of Article 6(7) DMA, and taking into account Alphabet's proposal and the available evidence,[1428] Alphabet should implement the measures in the Annex (Section 4.1) according to the timelines indicated therein.

## 11.3. Measures for on-device model implementation

*11.3.1. Introduction*

(1200) Many developers develop and offer on-device models (ODMs) which can be used to run and improve their own and third-party services. As described in Section 11.2.1, on-device models offer several advantages that make them attractive to adopt and provide.

(1201) Choice and innovation as regards ODMs running on a Google Android device is important, because these models have different strengths, such as size, speed, or accuracy. For example, on-device models can go beyond just text and can be capable of ingesting and outputting images, sound, or video. Alphabet submitted that the development of ODMs (such as the Gemini Nano ODMs) is fast-moving and that leading small models such as Google's Gemma (on which Gemini Nano is based), [third-party model], and [third-party model] have each set new performance benchmarks and improved basic functionalities over the past two years, only to be quickly overtaken.[1429] Alphabet's internal documents indicate that [Alphabet internal documents discussing language coverage, performance and controllability concerns perceived by OEMs in relation to Alphabet's on-device model]. [1430] [1431] [1432]

(1202) Mobile devices launching today are already able to run ODMs. As the hardware resources of mobile devices become more powerful, they will be able to run larger and more sophisticated ODMs. Simultaneously, as ODMs become more sophisticated and efficient, they may require less device hardware resources than today, making ODMs even more accessible.

(1203) This choice and innovation should not be prevented by a lack of access to functionalities that enable running an ODM on a mobile device efficiently and effectively.[1433] Otherwise, the gatekeeper can give their own ODM an unfair advantage, while arbitrarily degrading third-party ODMs that might otherwise be able to outperform the gatekeeper's ODM.

---

1428 Technical meeting Alphabet/Commission of 12 June 2026, Doc ID […], slide deck page 13. Alphabet's reply to RFI PO 21 of 8 June 2026, Doc ID […], question 6, paragraph 17.

1429 Alphabet's reply to RFI PO 9 of 04 March 2026, Doc ID […], paragraph 50.

1430 [Alphabet internal document], Doc ID […], page 1.

1431 Alphabet internal document, Doc ID […], page 1.

1432 [Alphabet internal document], Doc ID […], page 2.

1433 [Alphabet internal document discusses OEMs' dual-model approach involving its own ODM and Gemini Nano ODMs]. [Alphabet internal document], Doc ID […]. Another internal document provides that allowing [OEMs ODMs] [Alphabet internal document discusses potential OEMs innovation outside AICore functionality], see [Alphabet internal document], March 2025, Doc ID […], page 6.

(1204) Companies that raised the importance of ODMs for AI-based services and interoperability in that respect with the Commission include [AI service provider], [AI service provider] and [third party].[1434] This importance is also clear to Alphabet.[1435]

(1205) For instance, [discussion on OEM incorporating or planning to incorporate its own ODM on its devices and the use of this ODM for OEM AI services]. Alphabet was concerned about an [OEM ODM competing] with AICore for the device resources (e.g., RAM, NPU cycle, etc.). [Alphabet internal documents discussing that this would undermine the basis on which AICore was developed]. Alphabet's internal documents show that Alphabet was reluctant to enable this. In particular, [Alphabet internal documents discussing Alphabet's concerns on making third-party ODMs functionality accessible to third-party apps]. Alphabet sought to not raise this as a competitive concern or focus on the negative impact on Google's market position.[1436] [1437] [1438]

*11.3.2. Description of the feature*

(1206) The on-device model implementation feature (ODM implementation feature) consists in the ability to install, run and use on-device models on Google Android mobile devices. As on-device models are software components that might have a primitive interface and lack certain management functionality (including certain safeguards), those models are usually accompanied by additional software that supports such models, such as "wrappers" that add easily accessible APIs. The ODM implementation feature covers features available to AICore and ODMs implemented via AICore (including Gemini Nano and Gemma ODMs), as well as any future ODMs implemented via AICore or a successor or software performing a similar role to AICore on Google Android mobile devices ("AICore ODMs").

(1207) The functionalities of the ODM implementation feature include, amongst others, the following:

(a) Operation of on-device models. The ability to install, run and use on-device models and software that supports such models on Google Android devices.

(b) Access to hardware. The ability to timely and reliably access hardware resources that are necessary to run on-device models or software to implement such models, including:

(1) Access to specialised chips to run ODM tasks, such as the Neural Processing Unit (NPU) or Tensor Processing Unit (TPU);

(2) Access to the Central Processing Unit (CPU) and Graphics Processing Unit (GPU);

(3) Access to RAM (including Direct Memory Access) and other memory or storage.

---

1434 [Third party's] reply to RFI of February 2025, Doc ID […]; agreed minutes of meeting with [AI service provider] of 28 October 2025, Doc ID […], p. 4-5; [AI service provider], submission of 18 February 2026, Doc ID […], p. 5.

1435 Alphabet internal documents provide that "[o]n-device ML is [highly relevant for Android], see Alphabet internal document, Presentation […], Doc ID […].

1436 [Alphabet internal document], Doc ID […].

1437 [Alphabet internal document], 15 March 2025, Doc ID […], page 2.

1438 [Alphabet internal document], 15 March 2025, Doc ID […], page 3 [Alphabet internal document discusses OEM proposing APIs for functionality outside AICore].

(c) Background execution. The ability to provide and use on-device models or software to implement such models in the background, i.e. when the app hosting the model and/or the app using the model are not in the foreground.

(d) Central availability of ODMs to other apps. The ability for an app to centrally expose on-device models and software that implements such models to another app, and the ability for the other app to use such models and software. This includes the ability to exchange information between the service implementing the on-device model and the service making use of the on-device model.

(1208) The following sub-sections further describe access to hardware, background execution, and central availability.

11.3.2.1. Access to hardware resources

(1209) The performance of ODMs depends on the level of the access to hardware resources they are granted.

(1210) An important hardware resource is access to a mobile device's random-access memory (RAM), a type of temporary but fast memory that is particularly suited to running ODMs with low latency and high responsiveness. On a typical mobile device, RAM is much more limited than permanent storage: e.g. the Pixel 10 as a high-end device has 12 GB RAM, and at least 128 GB of permanent storage.

(1211) ODMs are typically sized from 500 MB to 3 GB today.[1439] To load an ODM can therefore take multiple seconds, as the device is loading from permanent storage into RAM. Only when the model is loaded into RAM, can it start executing.

(1212) RAM is a scarce and rivalrous resource on mobile phones. As ODMs can be large, the allocation of RAM among ODMs of different providers becomes particularly important to ensure a level playing field.

(1213) A service can receive a benefit from preferential RAM access. For instance, a certain amount of RAM can be reserved to a specific service; other services cannot use this RAM. This reservation of RAM is also called RAM residency or memory persistence. A service which is RAM-resident initiates faster, which results in it executing actions faster than services which are not RAM-resident and need to be reloaded into RAM so that their initiation may take several seconds.[1440]

(1214) Certain operating system functionality may rely on an on-device model. Such an on-device model may require substantial hardware resources (e.g., RAM, NPU, GPU, and CPU usage). According to Alphabet, the NPU cannot run two LLMs at the same time, nor run multiple requests from the same ODM at the same time.[1441]

11.3.2.2. Background execution

(1215) To work as intended, many apps and services must stay active and running even when the user is not actively using them, or must at least be able to timely resume their activity when necessary (see Section 11.4 for detail on background execution). This ability depends on whether the operating system provides adequate interfaces and allocates sufficient system resources to the app or service. System resources include, but are not limited to, access to the hardware resources mentioned in the

1439 E.g., [third party] reply to RFI of 12 February 2025, Doc ID […], page 6.

1440 The time depends among other things on the size of the ODM.

1441 [Alphabet internal document], Doc ID […], page 1; Alphabet's response to the PFs, Doc ID […], paragraph 329 7th bullet.

preceding section. This ability is also referred to as the operating system allocating "runtime" to an app.

(1216) To effectively and reliably power AI-based services such as AI assistants, ODMs must often run in the background to process information, for instance to prepare and show proactive suggestions. It is therefore important to have sufficient access to background execution capabilities.

11.3.2.3. Central availability

(1217) Central availability means that an app developer can make their ODMs available to other apps (own and third-party apps) via APIs. This is important to increase the adoption of the ODM by end users and other app developers. While it is possible to implement an ODM in each app that uses it, such an approach has severe downsides. Due to their size, app developers cannot easily include ODMs in their app, as the app size would increase dramatically. This would result in the app taking a considerable amount of permanent storage as well as very long download times, both very common reasons for end users to delete an app or abandon app installation. Central availability thus addresses two issues: (i) it avoids the need of having to download and store the ODM as part of individual apps, as the ODMs can be used by many apps; (ii) it enables widespread distribution of a company's ODM.

*11.3.3. Technical implementation of ODM implementation in Google Android*

(1218) Implementation of AICore. Alphabet currently implements the Gemini Nano ODMs via a software component called AICore. AICore incorporates Gemini Nano, Google's GenAI model for on-device deployment. Gemini Nano can be understood as the "engine" of AICore, and AICore as the "wrapper" for Gemini Nano.[1442] Alphabet submitted that AICore consists of certain Alphabet ODMs and three further components: scheduling functionality, model management, and safety features.[1443]

(1219) Alphabet's internal documents [Alphabet internal documents describe AICore's role in hosting and serving on-device models, and in establishing a standardised foundation necessary for the delivery of hardware-accelerated execution for AI services across device hardware by developers and OEMs].[1444]

(1220) Alphabet submitted that users cannot replace the Gemini Nano ODMs with another model and they can also not replace AICore with another software wrapper. Alphabet submitted that it is not possible to replace Gemini Nano because the experience and reliability of the ML Kit GenAI APIs "is fundamentally tied to the model used"[1445] and swapping out a ODM would compromise system integrity and undermine app functionality.[1446] Alphabet's internal documents show that Alphabet considers it possible to extend Google AICore to support third-party models: [Alphabet internal documents on enabling third-party models in AICore].[1447] [1448]

(1221) AICore is preinstalled on virtually all new high-end Android mobile devices (see recital (1108)). Alphabet preinstalls AICore on Pixel devices[1449] and grants it privileged system permissions. Alphabet requires the same of OEMs seeking to use

1442 Alphabet's reply to the RFI of 23 May 2025, Doc ID […], paragraph 4.
1443 Technical meeting Alphabet/Commission of 5 March 2026, Doc ID [...], slide deck page 6.
1444 Alphabet internal document, Presentation […], Doc ID […], page 24.
1445 Alphabet's reply to RFI PO 11 of 09 March 2026, Doc ID […], response to question 21.
1446 Technical meeting Alphabet/Commission of 5 March 2026, Doc ID […], slide deck page 14.
1447 [Alphabet internal document], Doc ID […], page 1.
1448 [Alphabet internal document], Doc ID […], page 1; see also email with subject [Alphabet internal document], Doc ID […], page 1 [confidential information from Alphabet internal document].
1449 Alphabet's reply to RFI PO 19 of 17 April 2026, Doc ID […], paragraph 26.

AICore.[1450] Alphabet submitted that AICore uses three sets of system-privileged APIs, which are therefore only available to pre-installed apps (see Section 5.2.1). Two of these sets of APIs are used in connection with on-device intelligence features.[1451] The third set of APIs is used to configure system values.[1452]

(1222) Implementation of ODMs outside AICore. Google Android also allows an ODM to be implemented and run in a regular native app (such as WhatsApp). Alphabet submitted that third-party ODM developers can run their own software wrapper like AICore and can call the same Android OS APIs.[1453]

(1223) Alphabet explained that [Alphabet is considering making available to third parties functionality provided via AICore to its own ODMs].[1454] Alphabet submitted that it is [Alphabet is developing hardware scheduling functionality relevant to third-party ODMs].[1455] [1456] Alphabet expects to release this functionality [confidential information about timing of release].[1457]

(1224) Implementation of RAM residency. Google Android contains services and technical parameters (such as the Google Android memory management service) which allows OEMs to grant preferential RAM access to a service.[1458] Alphabet provided that OEMs [confidential quotations from Alphabet internal documents regarding OEM adjustments to AOSP mechanisms in relation to memory management].[1459] While Google Android determines in many areas how hardware resources are allocated, it does contain certain technical parameters allowing OEMs to allocate hardware resources.

(1225) Alphabet grants such preferential RAM access to AICore, and therefore its Gemini Nano ODMs, on Pixel mobile devices.[1460] According to Alphabet's internal documents, [Alphabet internal documents discuss OEMs' positions regarding RAM residency].[1461]

---

1450 For instance, [refers to a commercial agreement with Alphabet signed by an OEM]. See Annex Q11 to Alphabet's reply to RFI PO 19 of 17 April 2026, Doc ID […].

1451 The android.app.ondeviceintelligence APIs and the android.systemvirtualmachine.virtualmachinemanagers APIs. Alphabet's reply to RFI PO 15 of 27 March 2026, Doc ID […], paragraphs 44-46.

1452 The android.provider.settings.APIs. Alphabet's reply to RFI PO 15 of 27 March 2026, Doc ID […], paragraph 47.

1453 Technical meeting Alphabet/Commission of 5 March 2026, Doc ID […], slide deck page 10.

1454 Technical meeting Alphabet/Commission of 5 March 2026, Doc ID […], slide deck page 6 and Alphabet's reply to RFI PO 11 of 09 March 2026, Doc ID […], paragraph 38.

1455 Alphabet's reply to RFI PO 15 of 27 March 2026, Doc ID […], paragraph 38.

1456 Alphabet's reply to RFI PO 16 of 01 April 2026, Doc ID […], paragraph 56.

1457 Alphabet's reply to RFI PO 15 of 27 March 2026, Doc ID […], response to question 12.a).

1458 Google Android contains functionality to allow OEMs to grant services privileged access to RAM: those services retain RAM access, while other services are removed from RAM. OEMs can adjust the likelihood that an application no longer has access to the RAM hardware resource by giving an app priority. OEMs can change the OOM_Score of an app via the AOSP mechanism OOM Adjuster (see Alphabet's reply to RFI AI 2 of 23 September 2025, Doc ID […] and reply to RFI PO 15 of 27 March 2026, Doc ID […], response to question 11.d)). See also Alphabet's reply to RFI AI 2 of 23 September 2025, Doc ID […], response to question 62 [device OEMs may use existing mechanisms to influence application memory allocation and prioritisation, including whether certain processes remain loaded in RAM].

1459 Alphabet's reply to RFI PO 16 of 01 April 2026, Doc ID […], paragraph 26.

1460 Alphabet's reply to RFI AI 4 of 28 November 2025, Doc ID [...], paragraph 67; Alphabet's reply to RFI PO 11 of 09 March 2026, Doc ID […], question 11.

1461 [Alphabet internal document], with subject […] with messages thread, Doc ID […], page 3.

(1226) A Pixel 10 phone contains 12 GB of RAM memory of which 3.5 GB are reserved by AICore via RAM residency.[1462] Other services, including third-party ODMs, cannot use this reserved RAM space even if Google's ODMs are currently not in use. AICore being RAM-resident means that any Alphabet service using functionality of Google's Gemini Nano ODMs apps will be faster than third-party services using ODMs which are not RAM-resident.

(1227) According to Alphabet, [OEM adjustments regarding AICore] [1463] This indicates that [OEM] granted AICore preferential access to device RAM or at least implemented changes to benefit AICore.[1464]

(1228) Alphabet submitted that end users can disable and uninstall AICore. As a result, the RAM residency ceases, and the RAM is released for use by other applications. The disabled AICore also causes the removal of the Gemini Nano model, with the remaining "stub" app taking up only 30 MB of permanent storage.[1465] As end users lose access to all functionalities that depend on AICore (including critical system functionality, see below), uninstallation may not be an attractive option.

(1229) Alphabet submitted that [Alphabet is modularising AICore to account for differing device capabilities and hardware resources].[1466]

(1230) Operating system dependency. Alphabet's AICore/Gemini Nano ODMs provide functionality both for operating system functionality and functionality of Alphabet services and hardware outside the operating system.

(1231) Alphabet explained that its Gemini Nano ODMs currently are used by [certain Google and Pixel components] and Pixel's Magic Cue.[1467] While Android System Intelligence is a software component that offers *inter alia* certain operating system experiences, including Pixel's Magic Cue (Section 9.3), a Google app such as the Messages app is not part of the operating system.

(1232) Alphabet submitted that AICore supports "critical system functionality". Alphabet provides the example that AICore is used to support scam call detection. More specifically, on calls, the user "could lose the ability to detect scam calls due to latency in AICore".[1468] The scam detection functionality is powered by Gemini Nano ODMs.[1469]

(1233) Central availability. As explained in Section 11.2.3, the ML Kit GenAI APIs expose access to Gemini Nano via AICore. Alphabet submitted that [confidential information concerning the framework to expose APIs accessible by other apps].[1470]

---

1462 See Annex Q11 to Alphabet's response dated 23 April 2026 to RFI 29. See also https://www.androidpolice.com/google-pixel-10-3-5-gb-ai-only-ram-partition/, Doc ID […], accessed on 20 April 2026.

1463 Alphabet 's reply to RFI PO 17 of 1 April 2026, Doc ID […], paragraphs 26-29.

1464 [Description of Alphabet's position regarding AICore] (see Alphabet's response to the PFs, Doc ID […], paragraph 329 3rd bullet). Alphabet does not contest that [OEM] had the power to grant preferential access to device RAM to AICore and made changes to benefit Gemini Nano. Alphabet does not substantiate its claim that all apps, including those implementing third-party ODMs, would be affected in a non-discriminatory manner by [OEM's] changes.

1465 Alphabet's reply to RFI PO 16 of 1 April 2026, Doc ID […], paragraphs 62-63.

1466 Alphabet's reply to RFI PO 16 of 1 April 2026, Doc ID […], paragraphs 39-47.

1467 Alphabet's reply to RFI PO 9 of 4 March 2026, Doc ID […], paragraph 54.

1468 Alphabet's response to the PFs, Doc ID […], paragraph 164, last bullet, 324.

1469 See https://support.google.com/phoneapp/answer/15654065, Doc ID […], accessed on 9 July 2026.

1470 Binder is a system for interprocess communication in AOSP that lets two processes on an Android-powered device communicate. See https://source.android.com/docs/core/architecture/ipc/binder-

It is unclear whether the private APIs that Alphabet uses to access Gemini Nano via AICore also use the [specific implementation framework].

*11.3.4. Feature falls within the scope of Article 6(7) DMA*

(1234) The Commission considers that the ODM implementation feature as described in Section 11.3.2 – together with its functionalities – is controlled via Google Android and falls within the scope of Article 6(7) DMA.

*11.3.5. Current access for Alphabet's own services and hardware*

(1235) The ODM implementation feature is available to and used by Alphabet.

(1236) Alphabet's own AI-related service AICore uses the ODM implementation feature to run the Gemini Nano ODMs.

(1237) As explained in Section 11.3.3, AICore uses three sets of system-privileged APIs.

(1238) As explained in Section 11.3.3, Alphabet's AICore is granted preferential access to hardware resources (RAM residency) on Pixel devices. Alphabet submitted that this would not be the case on third-party OEM Google Android mobile devices.[1471]

*11.3.6. Current access for third-party services and hardware*

(1239) Third-party services can implement ODMs on Google Android devices as described in Section 11.3.3.

(1240) However, there are four areas in which access is restricted or the level of interoperability unclear.

11.3.6.1. System-privileged APIs

(1241) Three sets of APIs used by AICore are system-privileged. This means that these APIs are not accessible for user-downloaded apps; they are only accessible for apps which the OEM decides to pre-install and grant such privileges (see Section 5.2). Access to the two sets of APIs used for on-device intelligence features can be important for third-party service providers seeking to offer similar functionality.[1472] For instance, a third party may seek to offer proactive suggestions or intelligent experiences relying on an ODM (see Section 9.3), for which access to these two sets of APIs can be important.

(1242) The third set of APIs is used to configure system values. Such functionality can also be useful for third-party ODM providers. In particular, Alphabet states that this may be used to configure timeouts,[1473] which may refer to an ability to more directly manage the allocation of resources and runtime. Alphabet's internal documents also show that [confidential information from Alphabet internal documents regarding an OEM's requested access to lower-level APIs for implementing an on-device model].[1474] It is also unclear whether these private APIs are implemented via the Binder framework.

(1243) To support its claim of interoperability, Alphabet pointed to [OEM's] statement that [OEM's ODM] and AICore have the same level of access to [OEM's] underlying hardware and that [OEM] "*said that it did not need to use the system-privileged*

---

overview, Doc ID […], accessed on 24 April 2026. See also Alphabet's reply to RFI PO 11 of 09 March 2026, Doc ID […], paragraph 53.

1471 Alphabet's response to the PFs, Doc ID […], paragraph 329 4th bullet.

1472 As acknowledged by Alphabet in Alphabet's reply to RFI PO 15 of 27 March 2026, Doc ID […], response to question 9.c).

1473 Alphabet's reply to RFI PO 15 of 27 March 2026, Doc ID […], paragraph 46.

1474 Alphabet internal document, [Alphabet internal document], Doc ID […], page 1.

*permissions that the Commission raised*."[1475] However, [OEM] did not state that it does not need to use the system-privileged permissions that the Commission raised either in the public consultation or in the reply to the RFI addressed to [OEM], which Alphabet cites in its reply to the PFs.[1476] [OEM], as an OEM with the ability to customize Google Android in certain respects and to build direct interfaces to device hardware, is in a markedly different position to implement an ODM than non-OEM developers.

(1244) In its response to the PFs, Alphabet did not engage with the potential use cases set out above for which third parties could use the APIs currently inaccessible to them. Alphabet provided vague explanations on why these APIs "are not needed by third parties".[1477] Alphabet proposed that third-party ODMs should only be able to access these APIs if they support system functions.[1478] In light of the vague information submitted by Alphabet and its admission that these APIs can be useful for third parties under certain circumstances, the Commission maintains that there is an interoperability gap. Alphabet is in a position to verify, in the implementation of this Decision, whether non-OEM third-party developers seeking to implement an ODM on Google Android mobile devices in fact do not need and do not benefit from access to these APIs.

11.3.6.2. RAM residency on Pixel devices and other preferential access to device resources

(1245) On Pixel devices, third parties do not have access to the same hardware resources as Alphabet's AICore which is granted preferential access to hardware resources (RAM residency). Moreover, third-party OEMs are able to grant AICore preferential access to hardware resources compared to third-party ODM providers. There is evidence that that happened at least for one device from the OEM [name of OEM] (see recital (1227)).

(1246) Moreover, Alphabet created a dependency of Google Android functionality to its Gemini Nano ODMs and AICore. It can be appropriate to grant operating system solutions privileged access to hardware resources to ensure that critical functionality of the operating system functions properly, i.e. with low latency and high performance.[1479] As explained, the Gemini Nano ODMs enjoy such privileged access (via RAM residency) on certain Google Android devices.

(1247) Such a dependency favours Alphabet's AI services, such as Gemini, compared to third-party AI services relying on a third-party ODM. Today, Google apps such as the Messages app currently using the Gemini Nano ODMs also benefit from this privileged hardware access by corollary. Since Gemini Nano is given preferential treatment by virtue of being used by operating system solutions, the Messages app can "piggyback" on that preferential treatment and have its functionalities that depend on Gemini Nano be more rapidly accessible resulting in a more performant

---

1475 Alphabet's response to the PFs, Doc ID […], paragraph 329.

1476 Alphabet's response to the PFs, Doc ID […], footnote 258.

1477 Alphabet submitted that these APIs and their respective permissions [are meant for system feature compatibility] and they are inherently designed for the features that call on AICore, particularly on-device intelligence functionalities, and that a downloaded ODM does not need features that are designed for the internal software workings of a preinstalled software component. See Alphabet's response to the PFs, Doc ID […], paragraph 331; presentation titled […] dated 21/22 May 2026, p. 26. Alphabet also submitted that [confidential information regarding internal management of AICore resources] without any substantiation (Alphabet's response to the PFs, Doc ID […], paragraph 329 last bullet).

1478 Technical meeting Alphabet/Commission of 21 May 2026, Doc ID […], slide deck 26.

1479 [Alphabet internal document], Document [Alphabet internal document], March 2025, Doc ID […], page 3.

service. By contrast, a Messages app from a third-party provider relying on a third-party ODM does not benefit from such access. While Gemini does not currently use Gemini Nano ODMs, it may do so in the future. In such scenario, Gemini could benefit from Gemini Nano ODMs being RAM resident, while a third-party ODM is not.

(1248) As explained in Section 11.3.7, Alphabet acknowledges that preferential access to device resources through RAM residency benefits its own services using functionality provided by Gemini Nano ODMs, but considers it "simply unavoidable".[1480]

(1249) Several stakeholders expressed concerns about AICore's RAM residency or other preferential hardware access. [OEM] and other OEMs raised issues regarding memory access vis-à-vis Google. [Confidential information from Alphabet internal documents regarding OEM concerns about NPU and memory contention, about lack of visibility over when Gemini Nano API runs and impacts the third-party models, and the acknowledgment of the LLM scheduling and memory management problem as widespread across OEMs and implementation of third-party on-device models].[1481] Alphabet submitted that the challenges discussed in those documents demonstrate little more than the fact that deploying ODMs is a technically complex process that requires a degree of coordination and debugging. [Alphabet submitted that the internal documents] reflect the standard back-and-forth as an OEM seeks to deploy a new, technically complex feature and that the Commission should consider that [OEM] achieved interoperability for its ODM on [OEM] devices and that [OEM's] and Alphabet's ODMs have the same level of access on [OEM] devices.[1482] The Commission considers that these internal documents also reflect the principled issue about non-discriminatory access to device resources which is shared by third-party OEMs. As to [OEM's] ability to achieve interoperability, it is important to note that the issue of RAM residency does not currently affect third-party OEMs because this type of preferential hardware access is limited to Alphabet's Pixel devices.

(1250) A senior Google employee internally also expressed concern about RAM residency. [Confidential internal Alphabet assessment of whether AICore should receive RAM residency or be handled under ordinary memory allocation rules].[1483] That means that the normal Google Android rules to allocate system resources should apply instead of granting AICore preferential access to hardware resources compared to other services.

(1251) [Third party] submitted that RAM limitations could prevent a third-party ODMs and software to implement such models from running concurrently with AICore on a Google Android device. In particular, if AICore and Gemini Nano benefit from RAM residency, loading other ODMs which are not RAM-resident could create a delay.[1484] Considering the RAM requirements of Google Android, AICore, and OEM's apps and systems, [third party] considered that only 2 GB out of 12 GB RAM may be available for other functions such as running apps.[1485] An on-device

---

1480 Alphabet's response to the PFs, Doc ID […], paragraph 327.

1481 [Alphabet internal document], […] Doc ID […], page 314.

1482 Alphabet's response to the PFs, Doc ID […], paragraph 329, 2nd bullet.

1483 [Alphabet internal document], email with subject […] with messages thread, Doc ID […], page 4.

1484 [Third party] reply to RFI of 12 February 2025, Doc ID […], page 6.

1485 [Third party] reply to RFI of 12 February 2025, Doc ID […], page 6. Alphabet submitted that it believes that it is incorrect that apps would only have two GB of RAM as many apps would not function with such constrained resources (Alphabet's response to the PFs, Doc ID […], paragraph 329 5th bullet). The

generative AI model may take an additional 1-4 GB of RAM when running. [Third party] submitted that "[i]n this situation, therefore, RAM memory limitations could effectively prevent a third-party orchestration framework [software to implement ODMs] and model from running on this device".[1486] [AI service provider] submitted that "a third-party assistant should have the ability to install, deploy, and access its own on-device model(s) (including any required runtime/acceleration) to be on par with AICore".[1487]

11.3.6.3. Alphabet is implementing new relevant functionality

(1252) [Confidential information regarding Alphabet's ongoing implementation of functionality relevant to third-party on-device models such as NPU scheduling and model management].[1488] NPU scheduling functionality is important for the level of hardware access for third-party ODMs.[1489] [Confidential information regarding Alphabet's planned NPU scheduling functionality and prioritisation approach for AI model tasks].[1490] [1491] Similarly, Alphabet implements scheduling functionality for access to the CPU and GPU.[1492]

(1253) Such access is also important for third-party ODMs. It is unclear how existing and future scheduling functionalities allocate hardware resources between Alphabet's and third-party services, particularly in case of simultaneous access requests and insufficient hardware resources to fully fulfil both requests at the same time. [Confidential information regarding a third-party OEM's request to Alphabet concerning access to AICore scheduling for NPU resources].[1493] Alphabet submitted that for the NPU scheduling functionality the exact prioritization mechanisms remain to be determined.[1494] Alphabet's internal documents show that it considered to only grant [OEM] access to the NPU for background processes and subject to limitations and [confidential information regarding limitations].[1495]

(1254) Alphabet submitted that it is already working to improve support for ODMs for all parties through other efforts.[1496] For instance, Alphabet provided that for a "diffuse set of features" it refers to as [confidential information on how Alphabet refers to such features] and which is currently part of AICore, it is exploring for certain functionality to become an operating system-level feature. The Commission considers that the measures attached to this Decision can aid Alphabet's efforts to comply with Article 6(7) DMA as it seeks to ensure the non-discriminatory allocation of device resources.

---

Commission considers that Alphabet did not corroborate its "belief". In any case, the point of principle remains correct: If AICore is RAM resident other apps have less RAM to work with.

1486 [Third party] reply to RFI of 12 February 2025, Doc ID […], page 6.

1487 [AI service provider], submission of 18 February 2026, Doc ID […], p. 5.

1488 Alphabet's reply to RFI PO 11 of 09 March 2026, Doc ID […], paragraph 49.

1489 Alphabet's reply to RFI PO 16 of 01 April 2026, Doc ID […], paragraph 58; Alphabet's reply to RFI PO 15 of 27 March 2026, Doc ID […], response to question 9.a).

1490 Alphabet's reply to RFI PO 11 of 09 March 2026, Doc ID […], paragraphs 34, 36.

1491 Alphabet's reply to RFI PO 15 of 27 March 2026, Doc ID […], paragraph 68.

1492 Alphabet's reply to RFI PO 16 of 01 April 2026, Doc ID […], paragraph 58.

1493 [Alphabet internal document], Presentation […], Doc ID […], page 1.

1494 Alphabet's reply to RFI PO 16 of 01 April 2026, Doc ID […], paragraph 59.

1495 Alphabet internal document, Document [Alphabet internal document], 15 March 2025, Doc ID […], pages 5-6.

1496 Alphabet's response to the PFs, Doc ID […], paragraph 333.

11.3.6.4. Uncertainty on future implementation

(1255) Fourth, there is significant uncertainty as to how Alphabet will implement the ODM implementation feature and whether such implementation will provide effective interoperability for third parties. In addition, given the evolving nature of the area, the level of interoperability that exists today may not be present in the future due to changes by Alphabet or the OEM.

(1256) Alphabet submitted that ODMs and frameworks such as AICore are fast-moving and that AICore specifically is still a relatively new product whose details are still evolving in response to Google's experience.[1497] [Confidential information regarding Alphabet's ongoing development of functionality for ODM access to hardware resources]. [1498]

*11.3.7. Commission's assessment of Alphabet's response to the Preliminary Findings*

(1257) Alphabet considers that the measures in the PFs in relation to the on-device model implementation feature are unlawful and disproportionate.

11.3.7.1. Concerning system functionality, current level of interoperability and integrity

(1258) *In the first place,* Alphabet's main criticism of the draft measures for the ODM implementation feature was the inclusion of a measure allowing end user to grant any preferential hardware access enjoyed by Alphabet's AICore to a third-party ODM[1499] (see discussion of RAM residency and OS dependency in Section 11.3.3). Alphabet submitted that AICore should be allowed to enjoy preferential hardware access because it supports system functionality, including "critical" system functionality. Further, it is "simply unavoidable" that Alphabet's services such as Google Messages (or Google Gemini in the future) benefit from such preferential hardware access, and third-party apps can use the AICore/Gemini Nano ODMs to also achieve the same benefit. Finally, Alphabet submitted that users can disable AICore to remove the preferential hardware access.[1500]

(1259) Alphabet argued that access to the system-wide ODM feature and the fact that users can remove the preferential hardware access by disabling AICore provides access which sufficiently compensates for the restricted interoperability at the level of the ODM implementation feature. On this basis, the Commission decided to remove the measure allowing users to grant the same preferential hardware access to third-party ODMs, provided that the degree of preferential hardware access is necessary to enable the functioning of the system functionality. For instance, for future design decisions, it may be necessary and proportionate for Alphabet to implement different ODMs, some to support system functionalities and others to support functionality for services. It is beneficial for the availability of hardware resources, and thus for contestability and innovation, that users can continue to disable AICore.

(1260) The Commission notes that Alphabet's design choices mean that it effectively implements AICore/Gemini Nano ODMs as part of the operating system by relying on it for system functionality already today, and potentially more so in the future. Alphabet acknowledges that interoperability at the level of AICore/Gemini Nano

1497 Alphabet's reply to RFI PO 9 of 04 March 2026, Doc ID […], paragraphs 51-52.
1498 Alphabet's reply to RFI PO 9 of 04 March 2026, Doc ID […], paragraphs 34, 36.
1499 Para. 108(d) in the draft measures attached to the PFs.
1500 Alphabet's response to the PFs, Doc ID […], paragraphs 326-328, 323-324, 331.

ODMs (i.e., the system-wide ODM feature) would prevent discrimination among different services, such as Gemini and third-party AI assistants.[1501]

(1261) *In the second place,* Alphabet submitted that the draft measure in the PFs which allows third parties to implement their on-device models via AICore in case Alphabet opens up AICore to some third parties,[1502] ignores that this would require integration work for each single third-party ODM which Alphabet expects to take several years even for [confidential information concerning Alphabet's estimated duration of integration work].[1503]

(1262) In response to Alphabet's submission, the Commission removed the measure. If Alphabet implements, or allows third parties to implement, third-party ODMs in AICore, it should consider the impact on contestability, fairness and innovation for third-party ODMs and third-party services using ODMs. Depending on how Alphabet complies with the ODM implementation features in the Annex, the development and importance of AICore/Gemini Nano ODMs and the needs of third parties, it might in the future become necessary to open up access to AICore for third-party models.

(1263) *In the third place,* Alphabet submitted that the measures unlawfully require interoperability at the "upper level" (the system-level ODM feature) and the "lower-level" (the ODM implementation feature).[1504]

(1264) The Commission considers that, against the legal background set out in Section 4.2, specifying measures both for the system-level ODM feature and the ODM implementation feature is effective and proportionate for the reasons set out in Section 11.2.7.

(1265) *In the fourth place,* Alphabet submitted that there is no interoperability gap because AICore/Gemini Nano ODMs can call on the same hardware resources via Android OS APIs as Gemini Nano (including CPUs, NPUs, memory management, sensors, and outputs).[1505]

(1266) The PFs listed four points in support of interoperability gaps and uncertainty on how Alphabet will implement interoperability. Alphabet's response to the PFs only responded in a meaningful way to two of these points. The point regarding preferential hardware access is discussed as the first item in this list earlier in this section and the Commission accepted Alphabet's claims. The second point which Alphabet substantiates concerns system-privileged permissions. As detailed and addressed in Section 11.3.6, the Commission does not consider Alphabet's arguments justified.

(1267) Alphabet does not sufficiently address how Alphabet will implement [confidential information regarding ongoing product development in relation to hardware scheduling] and does not reply to the finding that there is significant uncertainty as to how it will comply with Article 6(7) DMA given the evolving nature of the area.

(1268) *In the fifth place,* Alphabet submitted that the lack of interoperability for the system-privileged permissions and the preferential hardware access to power system functionality is justified by integrity considerations.[1506]

---

1501 Alphabet's response to the PFs, Doc ID […], paragraph 323.
1502 Para. 108(g) in the draft measures attached to the PFs.
1503 Alphabet's response to the PFs, Doc ID […], paragraph 329, 8th bullet.
1504 Alphabet's response to the PFs, Doc ID […], paragraphs 322-324.
1505 Alphabet's response to the PFs, Doc ID […], paragraphs 325-328.
1506 Alphabet's response to the PFs, Doc ID […], paragraph 331.

(1269) Alphabet's claims regarding preferential hardware access to power system functionality are accepted, as explained in recitals (1257)-(1260).

(1270) Regarding privileged permissions, Alphabet claims that integrity justifications cover the interoperability gap. Alphabet states that the permissions are inherently designed for the features that call on AICore, particularly on-device intelligence functionalities, and that a downloaded ODM does not need features that are designed for the internal software workings of a preinstalled software component.[1507]

(1271) In view of Alphabet's insufficient explanation, the Commission cannot verify the existence of an integrity concern or the strict necessity and proportionality of any integrity measure to address it. It is not an integrity concern that the system-privileges allow AICore to "deliver[] quality features that perform as users expect"[1508] or that a downloaded ODM does not need the features. It is not clear which integrity measure Alphabet proposes to mitigate the perceived concern. Alphabet may submit to the Commission a substantiated integrity justification during the implementation phase.

11.3.7.2. Concerning proportionality

(1272) Alphabet submitted that the PFs propose disproportionate changes to Google Android to address narrow issues that were not raised by third parties which have actually implemented ODMs, and that [confidential information regarding Alphabet's ongoing efforts to improve ongoing model support for all parties]. In Alphabet's view, the PFs take a disproportionate approach in interpreting fast-changing technology and developing capabilities as a status quo that requires specification.[1509]

(1273) Several companies raised the importance of ODMs for AI-based services and interoperability in that respect with the Commission. These include [AI service provider], [AI service provider], and [third party].[1510] It is clear from Alphabet's internal documents that [confidential information from Alphabet internal documents regarding an OEM's implementation issues with its on-device model alongside AICore/Gemini Nano] (see Sections 11.3.1 and 11.3.6). Alphabet's claim that [OEM] opposed the draft measures[1511] is unsupported. [OEM] understands the need for the measures and submitted that access to hardware resources — including RAM, NPU, and background execution capabilities — are essential to on-device model implementation.[1512] Alphabet refers to [OEM's] statement that equal access to RAM, NPU, and background execution capabilities does not mean unrestricted or uncontrolled access. The measures grant Alphabet control over how to allocate resources by, for instance, allowing Alphabet to set transparent, objective, precise, and non-discriminatory rules for allocating resources. Alphabet can decide the level of access, provided that the same level applies, in principle, to its own ODMs.

(1274) Alphabet's claim that the Commission should not specify measures in fast-changing technology is vague. It is not clear which measures Alphabet objects to. For instance, with respect to the measures on NPU scheduling, Alphabet retains control over how it grants access, provided that it does so based on transparent, objective, precise, and non-discriminatory rules. Alphabet does not explain why this requirement is

1507 Alphabet's response to the PFs, Doc ID […], paragraph 331.
1508 Alphabet's response to the PFs, Doc ID […], paragraph 331.
1509 Alphabet's response to the PFs, Doc ID […], paragraph 333.
1510 See Section 11.2.1
1511 Alphabet's response to the PFs, Doc ID […], fn. 259.
1512 [OEM] submission in response to the Commission's public consultation, 13 May 2026, Doc ID […].

disproportionate or hinders Alphabet to adapt to technological changes. The Commission's approach does not view "*developing capabilities as a status quo that requires specification*". Instead, it takes into account the likelihood of significant changes which could give rise to interoperability gaps and warrants specification. Moreover, the legislators adopted the ex-ante obligations in the DMA precisely to ensure contestability and fairness in fast-moving digital sectors to prevent entrenchment and tipping.

(1275) As regards Alphabet's claim of disproportionality, it fails to substantiate why the measures would be more restrictive than necessary to comply with Article 6(7) DMA and achieve its objectives. In particular, Alphabet fails to propose any less restrictive, yet equally effective means of complying with that provision. Nor does Alphabet explain and substantiate why a specific measure imposes a disproportionate burden on it. Instead, its arguments focus on questioning the scope of Article 6(7) and the actual need for interoperability – a line of reasoning which does not meet the legal standard for a valid proportionality claim, as explained in Section 13 below.

(1276) The Commission therefore concludes that there is an interoperability gap with respect to the ODM implementation feature of Google Android and uncertainty as to whether the feature will be fully accessible to third-party apps in the future. As a result, the Commission considers that it should specify how Alphabet must fill this gap and remove uncertainty to comply with Article 6(7) DMA.

*11.3.8. Measures that Alphabet should implement*

(1277) To meet the requirements of Article 6(7) DMA, Alphabet should implement the measures set out in Section 4.2 of the Annex to this Decision. The interoperability solution for the the ODM implementation feature, as described in Section 11.3.2 of this Decision must be equally effective as the solution available to Alphabet's AICore ODMs and related software.

(1278) AICore ODMs and related software means AICore and ODMs implemented via AICore (including Gemini Nano and Gemma ODMs), as well as any future ODMs implemented via AICore or a successor or software performing a similar role to AICore on Google Android mobile devices. This ensures that Alphabet cannot simply replace AICore or the ODMs in AICore to render the measures ineffective.

(1279) This means that Alphabet should provide third parties access to the functionalities available to AICore ODMs and related software. In particular, third parties should have access to functionality which enables Alphabet to install, run and use on-device models and software to implement such models; to timely and reliably access hardware resources to run on-device models or software to implement such models; to centrally expose ODMs and software that implements such models (such as a "wrapper" app) to another app, and the ability for the other app to use such models and software; and to provide and use on-device models or software to implement such models in the background.

(1280) In addition, to ensure effective interoperability for third parties, Alphabet should grant third parties access to additional functionalities available to AICore ODMs and related software if necessary to enable effective interoperability with the ODM implementation feature. Absent such access, third parties would not be able to compete on a level playing field.

(1281) Alphabet should provide third parties with an interoperability solution for the ODM implementation feature that is equally effective as that available to AICore ODMs and related software. Alphabet should therefore allocate hardware resources to

AICore ODMs and related software and third-party on-device models and software to implement such models according to transparent, objective, precise, and non-discriminatory rules;[1513] apply any restrictions, time windows, and resource limitations on background execution according to transparent, objective, precise, and non-discriminatory rules that also apply to AICore ODMs and related software;[1514] ensure that any limitation or choice on the background execution capabilities of, or access to hardware resources for third-party ODMs or software to implement such models as a result of a user action shall only be permissible if the user is allowed to take the same action with the same limiting effect regarding AICore ODMs and related software; allow third-party ODMs and software to implement such models access to the same functionalities to install, run and use them as available to AICore ODMs and related software, with the exception of functionalities that are available only to internal operating system components. In addition, if Alphabet allows OEMs to customise or otherwise change the level of access to hardware resources or background execution for AICore ODMs and related software, Alphabet should ensure that such customisation complies with all the measures specified in this Annex.[1515] This ensures that third-party ODMs and software to implement such models obtain effective interoperability also on third-party OEM devices.

(1282) The Commission recognizes that Google Android functionality (i.e., system-level functionality) depends on AICore/Gemini Nano ODM functionality and that enabling such system-level functionality may require access to specific device resources for AICore/Gemini Nano ODMs. As explained in Sections 11.3.3 and 11.3.7, Alphabet designed certain (critical) system functionality to rely on AICore/Gemini Nano ODMs. As a result of this design choice, Alphabet effectively implements AICore/Gemini Nano ODMs as part of the operating system by relying on it for system functionality already today, and potentially more so in the future.

(1283) The measures attached to this Decision recognize this dependency of Google Android on AICore/Gemini Nano ODMs. The preferential hardware access granted to AICore/Gemini Nano ODMs, or other ODMs supporting system functionality, must be necessary to enable the functioning of the system functionality. That is, the preferential hardware access must not go beyond what is necessary to support the system functionality.

(1284) For the purpose of ensuring that effective interoperability continues in the future, Alphabet should also provide effective interoperability with any future updates, including new functionalities, of the ODM implementation feature insofar as they are available to AICore ODMs and related software. To this end, the general measures in Section 12 of this Decision apply, including in particular the measures concerning future updates and new functionalities set out in Section 12.9 in this Decision.

---

1513 Alphabet may do this by, for example, implementing an appropriate system-level scheduler that complies with these requirements. Such rules may distinguish between the use of hardware resources for critical system-level functionality and other functionality where appropriate.

1514 Such rules may take into account aspects of system stability and device health, such as battery and thermal management. The rules may also take into account the need of system components – both Alphabet's and OEM's – to access background execution to preserve the internal functioning of the operating system, such as system-level hardware scheduling.

1515 In particular, any such customisations or changes granted to AICore ODMs and related software must also be granted to third-party on-device models or software to implement such models.

*11.3.9. Timing for implementation of the measures*

(1285) To meet the requirements of Article 6(7) DMA, and taking into account Alphabet's proposal and the available evidence,[1516] Alphabet should implement the measures in the Annex (Section 4.2) according to the timelines indicated therein.

## 11.4. Measures for background execution

*11.4.1. Introduction*

(1286) To work effectively, many apps need to perform tasks even when the app is not in the foreground, i.e. when the app is not being displayed on the screen. This is especially the case for AI-related apps, which often need to process data and perform actions to support the user while the user is interacting with other apps, or even when the user is not interacting with the device at all. To do this, apps need access to the background execution feature.

*11.4.2. Description of the feature*

(1287) Background execution consists in the ability to execute tasks in a timely manner, including when the app or service is not in the foreground (i.e. it is not being actively used by the user). This ability depends on whether the OS provides adequate interfaces to start and resume tasks, and whether it allocates sufficient system resources (including CPU, GPU, NPU, and RAM) to the app or service. This ability is also referred to as the OS allocating "runtime" to an app.

(1288) The OS is the primary arbiter in the allocation of resources for background execution. The fair allocation and "scheduling" of limited resources is one of the core tasks of an OS. The OS must continuously balance competing demands for such resources by different apps, giving priority to different processes depending on varying circumstances. This is done according to a set of rules that govern the allocation of resources to balance speed, reliability, and power consumption (as well as other aspects, such as smoothness of the user interface and heat production).

(1289) Background execution is particularly useful for AI-related apps, which in many cases are designed to help the user while performing other tasks in other apps, or even while the phone's screen is off. As explained in Section 11.3, on-device models require significant hardware resources in the background to ensure that tasks are executed and/or resumed whenever necessary. These requirements are not limited to on-device models. For example:

(a) An AI assistant may need background execution to ensure that it can perform actions in an automated way based on environmental conditions, even when the screen is off (for example, turn off the heating when the user leaves home).

(b) Context-aware intelligence, including AI-powered proactive suggestions (see Section 9.3), may need background execution to proactively remind the user of certain information at the appropriate time, even when the user is using another app.

(c) An AI-powered meeting coach may need background execution and access to CPU to instantly transcribe conversations as they happen, while generating summaries and action points in real time.

[1516] Technical meeting Alphabet/Commission of 12 June 2026, Doc ID […], slide deck page 13. Alphabet's reply to RFI PO 21 of 8 June 2026, Doc ID […], question 6, paragraph 17.

(d) An AI-powered sleep tracking app may need background execution to track a user's movements, detect snoring, and process similar data to wake up the user at the optimal time.

(1290) The functionalities of the background execution feature available to Alphabet's services include, amongst others, the following functionalities.

(a) **Access to hardware resources.** This consists in the ability to access hardware resources – including CPU, GPU, NPU, and RAM – to execute tasks. Access to the same background execution capabilities entails, in particular, being exposed and subject to the same limitations of these capabilities as a result of a user action. This includes the effect of an end user terminating an Android app in the app switching menu ("force-quitting" or "force-killing").

(b) **Requesting resource allocation.** This consists in the ability to indicate to the OS that the service requests allocation of certain amount of resources, e.g. for a certain period of time or while a certain process is active. This can be done through pre-defined OS interfaces (APIs) that ensure certain degree of priority in access to resources, typically under certain conditions that are tied to specific use cases. For example, such an API could guarantee that an AI service can synchronise data in the background for at least 10 seconds without showing a visible notification for the user, or for longer if a notification is shown.[1517] It could also guarantee that a connected device (e.g. earbuds) maintains a persistent connection with the Google Android device that allows the user to invoke an app (e.g. an AI assistant) using a hotword detected on the connected device. While such interfaces can be generally useful for specific use cases, they typically do not enable every use case, for which more generic ways of requesting resource allocation is needed.

(c) **Starting or resuming a task while the app is in the background.** This can depend on external conditions (e.g. an instruction received by another app; time-based conditions; a user approaching a certain location, such as their home; the user taking a picture) or internal conditions (e.g. the app concluding a task, such as the summarisation of a long audio recording). The task can be executed by the app itself (e.g. an AI assistant displaying a reminder notification at a given time) or by another app (e.g. an AI assistant sending a message via another messaging app at a given time).

(d) **Device configurations.** This consists in the ability of an app or service to be treated preferably by the OS, directly or indirectly, in terms of access to resources because of configuration settings. This includes configuration settings that allow an app to be persistently stored in memory or that allow an app to be exempted from power management mechanisms. These settings can be defined by Alphabet and/or the OEM, but can also be configurable by the end user. This functionality therefore includes the ability for the user to make choices on the degree of background execution allowed to a service, including regarding time, place, and cadence, and the presentation and effect of any prompt or user interface for that purpose.

---

1517 *"Beginning with Android 12, most foreground services do not show notifications to the user until they've been running for 10 seconds"*. See https://developer.android.com/develop/background-work/background-tasks, Doc ID […], accessed on 9 July 2026.

*11.4.3. Technical implementation of background execution in Google Android*

(1291) Android allows user-downloaded apps to make use of background execution, though with many limitations. App developers are generally encouraged to not manage the allocation of resources manually, but rather to use pre-defined Android APIs to carry out certain specific use cases.[1518] Generally, such pre-defined APIs are designed to reduce complexity for developers while being effective and reliable for the specific use case: since the API fulfils a specific function, the OS can treat the app in the most appropriate way. The allocation of background execution through such APIs is subject to certain limits, but comes with the advantage that the allocation is then mostly guaranteed and transparent to the developer.

(1292) This approach is colloquially described as the "API contract", which summarises the mutual expectations between the OS provider and the app developer: if the developer uses the correct API in the appropriate ways described in the API documentation, the OS will ensure that API behaves as described in the documentation, particularly in terms of resource allocation. The CDD contains "API behavioral compatibility" requirements that seek to ensure that the Android API contracts are not altered by OEMs.[1519]

(1293) For example, apps for virtual meetings are expected to use certain APIs to access the microphone while making calls.[1520] When an app uses such an API, the OS is supposed to support the use case as expected: for example, the OS should not suddenly quit the app in the middle of a call, as that is obviously a high-priority activity for the user at that moment. Moreover, the microphone should remain accessible to the app for the full duration of the call, including if the user switches to another app during the call, or turns off the screen.

11.4.3.1. Android's standard power restrictions: Doze and App Standby

(1294) In many cases, Google Android restricts resource usage by apps in order to save battery, especially when the app is in the background. The OS can go as far putting an app into "sleep" mode, where background execution is prevented, restricted, or deferred in order to save battery.

(1295) Google Android defines two main battery-saving regimes which effectively put apps into sleep mode, limiting their access to background execution:[1521]

(a) *Doze* reduces battery consumption by deferring background CPU and network activity for apps when the device is unused for long periods of time.

(b) *App Standby* defers background network activity for apps with no recent user activity.

(1296) Android provides several mechanisms for apps to access background execution in limited ways compatibly with Doze and App Standby. For example, an app can use "foreground services" (a background operation that is noticeable to the user, e.g.

---

1518 See e.g. "Alternative APIs" in https://developer.android.com/develop/background-work/background-tasks, Doc ID […], accessed on 9 July 2026.

1519 See Android 16 Compatibility Definition, https://source.android.com/docs/compatibility/16/android-16-cdd, Doc ID […], Section 3.5. See also Section 5.1 above on Android SDK APIs.

1520 See e.g. Foreground service types | Background work | Android Developers, Doc ID […], accessed on 17 April 2026, and see android.telecom | API reference | Android Developers, Doc ID […], 17 April 2026.

1521 See Alphabet's reply to RFI AI 6 of 23 February 2026, Doc ID […], question 16 and see Optimize for Doze and App Standby | App quality | Android Developers, Doc ID […], accessed on 17 April 2026.

playing music) to prevent the system to put the app into sleep mode.[1522] An app can also obtain a permission to schedule actions at an exact time.[1523] Moreover, apps can use Alphabet's Firebase Cloud Messaging service to resume an app by using a cloud-delivered message (which may be either visible or hidden to the user).[1524] However, all of these mechanisms have downsides.

(1297) *First,* these mechanisms are highly regulated at the OS level and at the Play Store level. For example:

(a) Foreground services are reserved to certain use cases.[1525] Apps can self-declare the type of foreground service they need, but Alphabet enforces the use case restrictions via Play Store policies.[1526]

(b) The permission to schedule actions at an exact time[1527] is denied by default for all non-privileged apps, with the exception of calendar and alarm apps.[1528]

(1298) *Second,* some mechanisms force developers to accept trade-offs. For example, while Alphabet recommends the use of Alphabet's Firebase Cloud Messaging service in some cases to resume an app from the cloud,[1529] this requires an active internet connection and comes with additional maintenance costs.[1530] Moreover, it requires that developers use an Alphabet service, and can have privacy implications, as the cloud-delivered message must necessarily go through Alphabet's servers.[1531]

(1299) *Third,* some mechanisms increase user friction and/or degrade the user experience. For example, the permission to schedule actions at an exact time can be granted manually by the user, but requires multiple steps: the app must deep-link into system settings, so that the user can grant the permission and then manually return to the app. Moreover, using this functionality might show an upcoming alarm even though there is no actual alarm scheduled, which can be confusing to the user.[1532]

(1300) *Fourth,* Alphabet itself acknowledges that, for some use cases, all these mechanisms might be insufficient. Google Android therefore provides a mechanism for apps to be

---

1522 See https://developer.android.com/develop/background-work/services/fgs, Doc ID […], accessed on 17 April 2026.

1523 See https://developer.android.com/about/versions/14/changes/schedule-exact-alarms, Doc ID […], accessed on 17 April 2026.

1524 See Optimize for Doze and App Standby | App quality | Android Developers, Doc ID […], accessed on 17 April 2026.

1525 See Foreground service types are required | Android Developers, Doc ID […], accessed on 17 April 2026.

1526 See Understand the requirements for Foreground Services and full-screen intent - Play Console Help, Doc ID […], accessed on 17 April 2026.

1527 The permission is called SCHEDULE_EXACT_ALARM, but "alarms" are not limited to traditional alarms that notify the user. Indeed, these "alarms" are often used to simply keep an app alive, e.g. to track a user's sleep, see agreed minutes of meeting with [third party] on 23 March 2026, Doc ID […].

1528 See Schedule exact alarms are denied by default | Android Developers, Doc ID […], accessed on 17 April 2026.

1529 See Optimize for Doze and App Standby | App quality | Android Developers, Doc ID […], accessed on 17 April 2026.

1530 Agreed minutes of meeting with [third party] of 23 March 2026, Doc ID […].

1531 US Senator Ron Wyden said foreign officials have demanded data from Alphabet's and Apple's push notification services for cloud-delivered messages – see Governments spying on Apple, Google users through push notifications - US senator | Reuters, Doc ID […], accessed on 9 July 2026.

1532 Agreed minutes of meeting with [third party] of 23 March 2026, Doc ID […].

partially exempted from Doze and App Standby restrictions, by asking the user to grant "Unrestricted" background usage.[1533] This can be done via two mechanisms.

(a) With *ACTION_REQUEST_IGNORE_BATTERY_OPTIMIZATIONS*, user-downloaded apps can prompt the user to grant (partially) unrestricted background usage within an in-app system dialogue without going into system settings, but this introduces user friction that pre-granted apps are not subject to. Moreover, for apps that are distributed via Alphabet's Play Store, Alphabet limits the use of this mechanism to some use cases.

(b) All other apps may only use an alternative method (*ACTION_IGNORE_BATTERY_OPTIMIZATION_SETTINGS*) that comes with even higher friction: instead of showing an in-app dialogue, the mechanism brings the user to Android settings, where the user needs to locate and select the correct app from a list of all (potentially hundreds of) apps, then needs to locate and select the correct option, and finally needs to navigate back to the app.[1534]

(1301) *Fifth,* all of the above applies only to stock Google Android, i.e. Google Android without any OEM customisation. As discussed next, Alphabet allows OEMs to implement additional proprietary app restriction mechanisms.[1535]

11.4.3.2. OEM proprietary background restrictions

(1302) Alphabet allows OEMs to implement proprietary mechanisms – often referred to as "app killers"[1536] – that restrict background execution, which may be even more restrictive than Doze and App Standby. In fact, AOSP itself is designed to allow OEMs to apply background restrictions to specific apps, and OEMs are also allowed to *"continue to use their custom methods to apply restrictions on the apps"*.[1537]

(1303) Proprietary app killers can be very aggressive towards certain apps, and developers often complain that they affect the expected behaviour of Android SDK APIs.[1538] For example, internal documents show that proprietary app killers may break [one of the AOSP frameworks to use background execution], effectively violating the API contract.[1539] In fact, Alphabet's API behavioural compatibility requirements in the CDD do not apply to apps that OEMs decide to restrict.[1540]

(1304) Alphabet imposes some requirements on proprietary app restriction mechanisms via the CDD,[1541] but such requirements leave significant margin of discretion to OEMs. For example, Alphabet requires OEMs to base proprietary restrictions on criteria related to system health (e.g. apps keeping the device awake when not necessary) but

---

1533 See Optimize for Doze and App Standby | App quality | Android Developers, Doc ID […], accessed on 17 April 2026.

1534 Alphabet 's reply to RFI PO 17 of 2 April 2026, Doc ID […], question 4b.

1535 See Android 16 Compatibility Definition, https://source.android.com/docs/compatibility/16/android-16-cdd, Doc ID […], Section 3.5.1 and Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID […], question 15.

1536 Agreed minutes of meeting with [third party] of 23 March 2026, Doc ID […], agreed minutes of meeting with [AI service provider] of 30 March 2026, Doc ID […].

1537 See App power management | Android Open Source Project, Doc ID […], accessed on 17 April 2026.

1538 Agreed minutes of meeting with [third party] on 23 March 2026, Doc ID […], agreed minutes of meeting with [AI service provider] on 30 March 2026, Doc ID […].

1539 [Alphabet internal document], Presentation […], Doc ID […], page 164.

1540 See Android 16 Compatibility Definition, https://source.android.com/docs/compatibility/16/android-16-cdd, Doc ID […], Section 3.5, requirement C-0-9.

1541 See Android 16 Compatibility Definition, https://source.android.com/docs/compatibility/16/android-16-cdd, Doc ID […], Section 3.5.1.

explicitly allows OEMs to determine such criteria and decide on which apps to enforce them. While OEMs are not allowed to use an app's lack of popularity in the market as a criterion, they may still exempt popular apps from the restrictions that apply to other apps (referred to as "allowlisting" – see Section 11.4.3.3 below). Overall, these requirements do not appear to prevent discrimination of some developers.

(1305) Moreover, despite the CDD requiring OEMs to publish documentation of how proprietary restrictions are applied,[1542] developers complain that app killers are generally not transparent to third-party developers.[1543] Finally, unlike for the "Unrestricted" background usage option found in stock Google Android, there is no guaranteed way for developers to prompt the user to grant exemption from proprietary app killers, and the "Unrestricted" background usage option is not guaranteed to have any effect for proprietary app killers.[1544]

(1306) This creates a complex and inconsistent web of technical and policy constraints that impacts developers and users. In fact, several apps provide extensive troubleshooting guides for users to experiment with system settings in order to avoid that the app stops working due to restricted background execution.[1545]

(1307) An Android developer – Urbandroid – created a dedicated website named *Don't kill my app*[1546] that aims to serve as a community resource to educate users on the issues and provide them with *"hacks, workarounds and guides to keep their apps working and making their lives easier"*. The website explains that generally users blame app developers for the issues that they encounter, while app developers have limited control over these issues because they depend on OEM customisations.[1547] The website provides recommendations to users on how to improve access to background execution for a given app depending on the brand of the Android device, but the recommendations usually involve significant friction and are not guaranteed to work, especially when OEM app killing policies are particularly aggressive. Urbandroid also provides a popular Android app to help users benchmark and troubleshoot background execution on their device.[1548]

(1308) Alphabet is aware that app developers face issues in terms of access to resources. In fact, in a 2023 article, Alphabet acknowledged feedback from the developer community that restrictions to background execution make it harder for developers to create apps that work across different device models. In the same article, Alphabet announced a *"deeper partnerships with Android hardware manufacturers to help ensure APIs for background work are supported predictably and consistently across*

---

1542 See Android 16 Compatibility Definition, https://source.android.com/docs/compatibility/16/android-16-cdd, Doc ID […], Section 3.5.1, requirement C-1-10.

1543 Agreed minutes of meeting with [third party] on 23 March 2026, Doc ID […], agreed minutes of meeting with [AI service provider] on 30 March 2026, Doc ID […].

1544 Alphabet 's reply to PO RFI 17 of 1 April 2026, Doc ID […], question 4e and question 4f.

1545 See for example guides for the following apps: Tasker (Tasker: Android System Power Management, Doc ID […], accessed on 21 April 2026, Signal (Troubleshooting Notifications – Signal Support, Doc ID […], accessed on 21 April 2026, Hedy AI (Hedy Background Operation Optimization Guide - Help Center - Hedy AI, Doc ID […], accessed on 21 April 2026, Strava (Troubleshooting Android GPS Issues – Strava Support, Doc ID […], accessed on 21 April 2026), Sleep as Android (Alarms do not ring - Sleep as Android, Doc ID […], accessed on 21 April 2026).

1546 See Don't kill my app! | Hey Android vendors, don't kill my app!, Doc ID […], accessed on 21 April 2026, and see agreed minutes of meeting with [third party], Doc ID […].

1547 See Our mission | Don't kill my app!, Doc ID […], accessed on 21 April 2026.

1548 See DontKillMyApp: Make apps work - Apps on Google Play, Doc ID […], accessed on 21 April 2026.

*the ecosystem"*, starting with Samsung.[1549] While this partnership [further information about this partnership], user-downloaded apps are still facing issues.[1550] Moreover, the improvements do not prevent OEMs from restricting apps or giving preferential treatment to some apps at their discretion, as explained next.

11.4.3.3. OEM exemptions from background restrictions

(1309) Not all app developers face the restrictions and complications described above. Pre-installed apps and selected developers, including Alphabet, can avoid these issues by obtaining preferential treatments from OEMs. This is achieved via *exemptions* (also referred to as *pre-granting*, *whitelisting*, and *allowlisting*), which can be achieved in different ways, both directly and indirectly, and are typically granted to pre-installed apps. For example:

(a) Just like for restrictions, Android is designed to allow OEMs to define a "power allowlist" to directly specify which apps are exempted from Doze and App Standby.[1551] OEMs can also directly exempt specific apps from their own power management mechanisms.

(b) OEMs can use system-initiated *binding*,[1552] a mechanism that indirectly grants background execution in a way that, in practice, is equivalent to direct allowlisting. This mechanism works by instructing the OS (specifically, the system server) to bind to a specific app.[1553] Because the system server is always running, the binding mechanism effectively also *"keeps [the app] running"* in the background,[1554] even if the app is not directly added to the power allowlist.[1555] Internal documents show that system-initiated binding can be used to *"propagate the importance"* of an app towards the OS via a chain of bindings that originate from the system server.[1556]

(1310) Alphabet explains that *"[p]reloaded system apps and cloud messaging services are typically exempted from App standby and Doze by default"* and that *"users can't change the exemption status of any app or service that is exempted by default in the system image"*.[1557]

(1311) Alphabet effectively recognises that exemptions can be abused by OEMs in a way that can *"compromise the user experience"*. For this reason, Alphabet *"strongly recommend[s] that [OEMs] don't exempt third-party apps and instead exempt only*

---

1549 See Android Developers Blog: Improving Consistency of Background Work on Android, Doc ID […], accessed on 21 April 2026.

1550 Agreed minutes of meeting with [third party] , Doc ID […], agreed minutes of meeting with [AI service provider] of 30 March 2026, Doc ID […], [AI service provider], [Submission to the Commission], 22 March 2026, Doc ID […].

1551 See Power management | Android Open Source Project, Doc ID […], accessed on 21 April 2026, and Alphabet 's reply to RFI PO 17 of 1 April 2026, Doc ID […], question 4a.

1552 See Bound services overview | Background work | Android Developers, Doc ID […], accessed on 21 April 2026.

1553 More precisely, the system server is bound to a service that is hosted by an instance of the specific app.

1554 Alphabet 's reply to PO RFI 17 of 1 April 2026, Doc ID […], question 5b.

1555 While the binding mechanism is also available to user-downloaded apps, user-downloaded apps may not instruct the system server to bind to a service, meaning that the service will keep running only if the app that binds it is still running. In other words, the bound service can only get at most as much background execution as available to the app that binds it – which is more limited than what is available to the system server, which is always running.

1556 [Alphabet internal document], Doc ID […], page 2.

1557 See Power management | Android Open Source Project, accessed on 21 April 2026, Doc ID […], accessed on 21 April 2026.

*cloud messaging services or apps with similar functions"*. However, this is not a hard requirement.[1558]

(1312) In practice, just like in the case of restrictions, Alphabet and OEMs grant exemptions in discretionary ways. For example:

(a) Alphabet pre-grants some apps (including Alphabet's first-party apps, such as Google Camera, Google Clock, Google Messages, Google Play Services, Google Phone, VPN by Google) the "Unrestricted" option for background usage on Pixel devices. Users cannot revoke this option for these apps. Alphabet also uses the system-initiated binding mechanism to ensure that Magic Cue[1559] keeps running in the background.[1560]

(b) [Alphabet requires OEMs via commercial agreements to place some Google apps on a power allowlist and to exempt them from proprietary OEM restrictions].[1561] [1562]

(c) Alphabet asked and obtained that [OEM] add one of its own apps (Google Photos) to the power allowlist for [OEM] devices.[1563] Alphabet also asked [OEM] to allowlist AICore, which was frequently killed by the OS when using Gboard (Alphabet's keyboard app) due to [OEM's] proprietary restrictions of RAM usage for certain processes.[1564]

(d) [One OEM had put their app on a list exempting them from showing up in the Settings menu where the user could reduce background execution],[1565] [1566] meaning that the [OEM] app store would enjoy the same preferential access to memory as the apps enabled in the Memory Resident Apps menu, but without requiring the user to take any action and without giving the user the possibility to disable this, unlike for user-downloaded apps. Moreover, on that occasion, Alphabet explained to [OEM] that this could negatively impact users of Google Play Store (e.g., because it could interfere with the delivery of Google Play System Updates). [OEM] agreed to also exempt the Google Play Store from this feature.[1567]

(e) [OEM] offered to add [third party] apps to their whitelist, but only if [third party] would agree to distribute its apps on [OEM's] app store.[1568]

(1313) These examples show that Alphabet and OEMs not only have broad discretion in restricting or exempting certain apps, but also that such discretion can be exercised

---

1558 See Power management | Android Open Source Project, Doc ID […], accessed on 21 April 2026

1559 Magic Cue is Alphabet's implementation of proactive suggestions via context-aware intelligence – see Section 9.3.1.

1560 Alphabet 's reply to RFI PO 17 of 1 April 2026, Doc ID […], question 5c.

1561 See GMS requirement GMS-8.16-002, Alphabet internal document, Document [Alphabet internal document], Doc ID […].

1562 See GMS requirement GMS-8.16-002, Alphabet internal document, Document [Alphabet internal document]", Doc ID […].

1563 Alphabet internal document, Presentation [Alphabet internal document], Doc ID […], page 27.

1564 [Alphabet internal document], Doc ID […], page 40 and Alphabet 's reply to RFI PO 17 of 1 April 2026, Doc ID […], question 15.

1565 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID […], question 17.

1566 The Memory resident apps menu on Samsung devices informs the user that *"[m]emory resident apps run in the background all the time. Sometimes this is necessary for an app to work correctly"*. See https://www.youtube.com/watch?v=BQp2su9S7rI, accessed on 21 April 2026, Doc ID […].

1567 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID […], question 17.

1568 Agreed minutes of meeting with [third party] of 23 March 2026, Doc ID […].

based on unilateral commercial incentives (see Section 6.3.1) and can lead to discriminatory outcomes.

*11.4.4. Feature falls within the scope of Article 6(7) DMA*

(1314) The Commission considers that the background execution feature as described in Section 11.4.2 – together with its functionalities – is controlled via Google Android and falls within the scope of Article 6(7) DMA.

*11.4.5. Current access for Alphabet's own services and hardware*

(1315) Background execution is available to and used by Alphabet's services.

(1316) *First,* several Alphabet apps are pre-granted the "Unrestricted" background usage option on Pixel devices, which exempts these apps from Android's Doze optimisations (see paragraph (1312) above). Similarly, Alphabet uses system-initiated binding to keep some apps running in the background, without directly allowlisting them (see paragraph (1309) above).

(1317) *Second,* [Alphabet requires OEMs via commercial agreements to place some Google apps on a power allowlist and to exempt them from proprietary OEM restrictions].

(1318) Third, Alphabet has bespoke agreements in place with OEMs to exempt Alphabet apps. Internal documents show that Alphabet convinced [OEM] to ensure that [Google app] did not get restricted (deep-sleep) even when unused, which was hurting [internal documents referring to a type of backup negatively affected on OEM devices].[1569]

(1319) *Fourth,* Alphabet can use Google Play Services to indirectly grant unrestricted background execution to any Alphabet app and service. Indeed, Alphabet explained that Google Play Services contains *"common features of Google apps, such as components of the Google account system"*.[1570] This means that Alphabet always has the option – as the provider of Google Android – to implement some functionalities into Google Play Services and then expose it to another Alphabet app, rather implementing it into that app itself. Because Google Play Services enjoys unrestricted background execution, those functionalities will also be unrestricted, meaning that the separate app will in turn benefit from unrestricted functionalities even if the app itself is not unrestricted.

(1320) Alphabet already benefits from such background execution in AI-related services, both directly and indirectly. For example:

(a) On Google Pixel devices, Alphabet established a persistent connection between AICore and Pixel's device intelligence services (which rely on AICore/Gemini Nano). This dependency is then [assigned a higher priority in terms of access to memory],[1571] meaning that AICore/Gemini Nano has higher chances of being kept in memory in the background, so that there is no delay due to loading AICore/Gemini Nano in memory when apps – particularly Alphabet apps – use AICore/Gemini Nano functionalities in the background.

(b) Magic Cue – Alphabet's solution for proactive suggestions (see Section 9.3.1) – is granted preferential background execution via system-initiated binding on Pixel devices.[1572] In turn, all Alphabet services that are currently supported by

1569 [Alphabet internal document], Doc ID […], page 27.
1570 Alphabet's reply to the RFI of 17 July 2025, Doc ID […], paragraph 19.
1571 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID […], paragraph 60.
1572 Alphabet 's reply to RFI PO 17 of 1 April 2026, Doc ID […], question 4c.

Magic Cue (such as Gmail, Google Calendar, Google Keep) indirectly use preferential background execution.

(c) The Gemini app uses background execution via the Firebase Cloud Messaging service. Indeed, users can ask Gemini to perform "Scheduled actions", i.e. recurring actions in the Gemini app to get a response based on the timing and conditions that the user chooses.[1573] This includes, for example: daily summaries of the user's calendar, to-do list, and unread emails for the day ahead; daily updates on the weather and outfit ideas based on a list of wardrobe items that the user gives to Gemini; weekly news updates or recommendations for new nearby restaurants to try out. [Confidential information regarding the background-execution mechanism supporting Gemini scheduled actions and related OEM exemption requirements]. Such notifications are delivered promptly and reliably thanks to Alphabet's access to background execution, [confidential information regarding the background-execution mechanism supporting Gemini scheduled actions and related OEM exemption requirements].[1574]

(d) Gemini assistant indirectly uses the background execution feature via Google Photos, which is allowlisted on [OEM] devices (see paragraph (1312) above). Indeed, synced photos from Google Photos are then used by Gemini: for example, a user can ask Gemini to find photos (e.g. to find "recent selfies") or to find information based on photos (e.g. to retrieve the number of the user's driving license).[1575]

(1321) All of the examples above are achieved by Alphabet without asking the user to take any action. For example, Alphabet does not ask the user to grant unrestricted background usage to access background execution in the ways described above.

(1322) Importantly, internal documents show that [Alphabet can use preferential exemption mechanisms to temporarily improve background execution of certain software components such as AICore] for as long as is necessary to [reasons for using the exemption mechanism].[1576] Once adoption is widespread enough, the preferential treatment may no longer be necessary, because [Alphabet internal documents explaining why the preferential treatment may not be needed anymore].[1577] In other words, the preferential treatment may be required to build the initial network effects, but may be no longer necessary once the network effects are created and most developers rely on that component anyway (as opposed to third-party alternatives). While this may be a technically sound approach from an engineering standpoint, the same approach is not available to user-downloaded apps. This shows how Alphabet's ability to allocate preferential background execution (or any other type of preferential treatment) at a certain time can lead to long-lasting entrenchment of middleware (and related services, such as Firebase Cloud Messaging) that is very difficult to contest – as is currently the case for Google Play Services (see Section 5.1.2).

---

1573 See Schedule actions in Gemini Apps - Android - Gemini Apps Help, Doc ID […], accessed on 21 April 2026.

1574 Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID […], question 16.

1575 See https://support.google.com/gemini/answer/15734842?hl=en&co=GENIE.Platform%3DAndroid#zippy=%2Cfind-info-in-photos-videos-from-google-photos%2Cfind-photos-videos-albums, Doc ID […], accessed on 21 April 2026.

1576 [Alphabet internal document], Doc ID […], page 4.

1577 [Alphabet internal document], Doc ID […], page 4.

*11.4.6. Current access for third-party services and hardware*

(1323) As explained in Sections 11.4.3.2 and 11.4.3.3 above, Alphabet and OEMs can resort to app restrictions and app exemptions (both direct and indirect) to obtain preferential access to background execution. On the other hand, as explained above, stock Google Android provides several mechanisms for user-downloaded apps to access background execution in limited ways, but all of these have downsides and are not guaranteed to work with OEMs' proprietary power restrictions. Moreover, Alphabet explains that application developers can and do work with OEMs to obtain similar whitelisting to what Alphabet requires for some Alphabet apps via [commercial agreement between Alphabet and OEMs].[1578]

(1324) The Commission considers that these possibilities are not sufficient to provide equally effective interoperability, and access for the purpose of interoperability, with the background execution feature that is available to Alphabet's services. In particular, requiring developers to obtain the same exemptions from Alphabet and OEMs does not amount to effective interoperability.

(1325) Effective interoperability should be provided via stable, reliable, documented OS interfaces that are easily accessible to business users and, if appropriate, are subject to permission and control of the end user. Internal documents confirm that consistency and transparency are critical for developers. Alphabet knows that [broken and fragmented app experiences increase the workload of customer support load and engineering teams].[1579]

(1326) Subjecting the conditions of access to agreements with OEMs is therefore not effective interoperability, as it is likely to lead to conditions that are not transparent, objective, precise, and non-discriminatory. [OEM's] offer to allowlist [third party's] apps, subject to [third party] distributing apps on [OEM's] app store is a case in point. Internal documents show that Alphabet itself offers [confidential information from Alphabet internal documents regarding performance support and optimisations offered by Alphabet to external app partners].[1580]

(1327) Internal documents show that Alphabet [confidential information from Alphabet internal documents regarding the effects of OEM power restrictions]. Internal documents explicitly acknowledge that [smaller apps are disadvantaged compared to popular apps which are more easily automatically whitelisted by major OEMs].[1581] Alphabet employees know that developers need business resources to work with OEMs, that developers have to maintain specific support guide for users to self white-list, and that the situation ultimately leads to poor reviews by the user of these apps.[1582] While Alphabet employees consider that *"OEM app whitelists and force-stops without user consent violate [OEM commercial agreement]*,[1583] Alphabet allows them nevertheless.

(1328) In fact, developers regularly complain about irregular and inconsistent access to resources in their Android apps. As explained in paragraph (1307), some developers

---

1578 GMS Requirements, Document "Annex Q13.2 - DMA.100210 - EC AI RFI #2 of 23 September 2025 (8 October 2025) (Confidential - Contains Business Secrets).pdf", Doc ID […], submitted in Alphabet's reply to RFI AI 2 of September 2025, in particular requirement GMS-8.16-002. Alphabet's reply to RFI PO 6 of 23 February 2026, Doc ID […], question 16.

1579 [Alphabet internal document], Doc ID […], page 4.

1580 [Alphabet internal document], Doc ID […], page 41.

1581 [Alphabet internal document], Doc ID […], pages 164-165.

1582 Alphabet internal document, Presentation [Alphabet internal document], Doc ID […], pages 164-165.

1583 Alphabet internal document, Presentation [Alphabet internal document], Doc ID […], pages 164-165.

created a dedicated website (dontkillmyapp.com) to help users troubleshoot app killer issues. Developers have also reported the issue to Alphabet: a well-known complaint reported to Alphabet on Android's public issue tracker was filed in 2018 and upvoted by over 1'000 developers and users, but the status of the issue is still *"In Progress (Accepted)"*.[1584]

*11.4.7. Alphabet's response to the Preliminary Findings*

(1329) Alphabet considers that the draft measures are unlawful because Alphabet already complies with Article 6(7) DMA.[1585] Alphabet considers that third-party apps already have access to the background execution techniques that Alphabet's services use, and that it is unclear whether the issues described in the PFs refer to discrimination in favour of Alphabet. Alphabet considers that any negative impact of OEM decisions on apps is not in scope of Article 6(7). Alphabet also considers that the PFs' evidence is partial and imbalanced.

(1330) Alphabet considers that the draft measures are disproportionate,[1586] because they are a threat to system health and OEM device design, and risk undermining the functioning of basic device features. According to Alphabet, Android already supports many mechanisms for background execution and provides better support than rival mobile operating systems.

*11.4.8. The Commission's assessment*

11.4.8.1. Concerning the existing level of interoperability and integrity

(1331) **Interoperability gap.** Alphabet argues that the Android OS allows interoperability for first-party and third-party services under equal conditions and that Google's own apps are also subject to OEMs' prioritisation decisions.[1587] In doing so, Alphabet effectively repeats the same arguments that it makes across the rest of its response to the PFs, i.e. that Article 6(7) permits Alphabet to allow the OEMs to treat more favourably certain services over others, including Alphabet's services. These arguments are already rebutted in detail in Sections 4.6 and 6.1 of this Decision. Alphabet's arguments on background execution therefore miss the point: as explained in Section 11.4.6, allowing OEMs to selectively and indiscriminately exempt apps from background execution limitations and to further restrict other apps does not provide effective interoperability under equal conditions. As explained in detail in Section 11.4.3.2, it is Alphabet's CDD and [describes Alphabet's confidential agreements with OEMs] that allow OEMs to take these measures, and it is Alphabet's [describes Alphabet's confidential agreements with OEMs] (as well as Alphabet's close relationship with OEMs) that allows Alphabet to take advantage of this setup.

(1332) As for Alphabet's claims that the PFs rely on partial and anecdotal evidence,[1588] the Commission notes that this Decision provides plenty of evidence from both third parties and Alphabet's internal documents that the discriminatory outcomes are not just a theoretical possibility, but a concrete issue of discrimination that goes (i) to the detriment of third-party app developers and (ii) to Alphabet's advantage. Sections 11.4.3.2 and 11.4.6 discuss in detail the many background execution issues that [third party] and [AI service provider] have encountered, as well as the fact that this are not

1584 See https://issuetracker.google.com/issues/122098785, Doc ID […].
1585 Alphabet's response to the PFs, Doc ID […], paragraph 335.
1586 Alphabet's response to the PFs, Doc ID […], paragraph 335.
1587 Alphabet's response to the PFs, Doc ID […], paragraphs 337-342.
1588 Alphabet's response to the PFs, Doc ID […], paragraph 339.

isolated cases but rather examples of a widespread issue across the Android ecosystem of developers. Alphabet does not contest – let alone substantiate – that the issues experienced by these developers are real. Alphabet implicitly dismisses them by saying that the same conditions apply to Alphabet. This is not true: Section 11.4.5 provides evidence of how OEMs – including Alphabet itself for Pixel devices – have granted exemptions to some services – including Alphabet services – to offer an easy and preferential way out of issues that plague thousands of developers and raise their development costs. Alphabet dismisses this evidence as a *"series of anecdotes about when Google has entered discussions with OEMs to solve specific background execution problems that Google apps were facing"*.[1589] This Decision makes it clear that discrimination does not arise from the fact that Alphabet speaks to OEMs, but rather that Alphabet allows OEMs to take discretionary and discriminatory decisions that advantage Alphabet and penalise others.

(1333) Furthermore, Alphabet considers that the concern of discrimination in access to background execution described above does not relate to preferential interoperability afforded to Google apps specifically.[1590] This is openly contradicted by the evidence referenced in the previous paragraph, such as the internal documents showing that Alphabet asked [OEM] to exempt [Google app] from background restrictions for synchronisation with Alphabet's cloud servers – which is a blatant example of preferential treatment of an Alphabet service that is complementary to the operating system. This highlights a contestability issue.

(1334) In any event, the Commission notes that fairness is also an objective of the DMA and Article 6(7). Article 6(7) does not allow Alphabet to discriminate against some third-party services and favour others by selectively restricting or enhancing interoperability, even if Alphabet would not (yet) use that possibility to favour its own services. This could lead to perverse outcomes under Article 6(7), such as allowing the gatekeeper to charge fees to access "enhanced" forms of interoperability, or to grant selected companies (including other gatekeepers) preferential access to OS features based on the gatekeeper's commercial interests. This type of discrimination would directly go against the objective of fairness, which seeks to address the "superior bargaining power" of gatekeepers, which allows them to "unilaterally set unbalanced conditions" that may also consist in "discriminating against business users".[1591] Instead, Article 6(7) requires that whenever a feature is available to a service of the gatekeeper, the gatekeeper must allow equally effective interoperability for all business users. This does not mean that interoperability must be indiscriminate or unfettered: to the contrary, the measures for background execution allow Alphabet and OEMs to define rules that depend on the type of app and on the circumstances, provided that such rules are transparent, objective, precise, and non-discriminatory.

(1335) **Integrity.** Alphabet considers that the measures would put at risk the integrity of Android devices because developers could overuse device resources like CPU or memory.[1592] Alphabet suggests that the risks could result in battery drain, physical damage of the device, and harm to other developers that use resources more sparingly. The Commission notes that these claims are based on a misrepresentation

---

1589 Alphabet's response to the PFs, Doc ID […], paragraph 339, third bullet.

1590 Alphabet's response to the PFs, Doc ID […], paragraph 341.

1591 Recital 33 DMA. The recital explains that such discrimination can arise if the gatekeepers and business users provide competing services, but the use of "in particular" confirms that this is not a necessary condition to consider some behaviour as discriminatory.

1592 Alphabet's response to the PFs, Doc ID […], paragraphs 340 and 342.

of the measures. The measures do not require that Alphabet or OEMs *"lose control over background execution"*, they do not allow apps to *"misuse foreground services to overuse device resources"* or to continue *"running under all conditions"*, and they do not prevent Alphabet or OEMs from *"making clear to users the likely effect on device performance"* of background execution.[1593] In fact, Alphabet does not even pinpoint the background execution measures that would allegedly cause those adverse outcomes. Alphabet just vaguely refer to the measures as *"a proposed Draft Measure"* [1594] and then misconstrues their content.

(1336) In a nutshell, the measures simply require that background execution (including access to resources) is allocated non-discriminatorily to all services. Alphabet and OEMs retain full freedom to define the rules that govern background execution, provided that they are non-discriminatory. The measures even explicitly allow for mechanisms that support user transparency and user choice, provided the same mechanisms apply to Alphabet's services.

(1337) Importantly, the measures do not require parity between third-party services and internal OS components, as Alphabet appears to wrongly assume.[1595] The measures only require parity between third-party services and Alphabet's services (such as Gemini, Google Photos, Google Play Store, Firebase Cloud Messaging) – which is what Article 6(7) requires. It is acceptable (and generally appropriate) that internal OS components that are not services within the meaning of Article 6(7) enjoy privileged access to background execution. However, such privileged access should not extend to services, either directly or indirectly – see Section 11.4.3.3. The Commission further clarified this aspect in the updated measures in the Annex to this Decision, also in response to comments from Alphabet and OEMs in the public consultation, as discussed next.

(1338) **Role of OEMs.** Alphabet claims that OEMs' freedom over how they manage background processes is a strictly necessary and proportionate measure to address the integrity concerns explained above.[1596] According to Alphabet, this freedom is a technical necessity to balance app needs for background execution with users' interest in system health, battery drain, and excessive network usage. These claims are again premised on a wrong representation of the measures: the measures allow both Alphabet and OEMs to retain freedom in how they manage background processes. Beyond the flexibility that the measures leave for Alphabet, the measures explicitly allow OEMs to customise background execution rules as they please, provided that the rules are non-discriminatory. The measures therefore allow OEMs to *"adjust for […] differences" in "[h]ardware specifications"*.[1597]

(1339) OEMs' responses to the public consultation, including those cited by Alphabet,[1598] either reflect the same misunderstanding of the measure as Alphabet's[1599] or simply

---

1593 Alphabet's response to the PFs, Doc ID […], paragraphs 340, 342, 344.

1594 Alphabet's response to the PFs, Doc ID […], paragraph 340.

1595 See Alphabet's response to the PFs, Doc ID […], paragraph 346, second bullet. Alphabet states that *"it is not clear if services providing basic system functionality like the user interface are included"* as a benchmark for the non-discrimination requirements and that *"this would subject the user interface itself to instability"*. Alphabet here misunderstands the measures: this is not a risk because the OS-level system interface is generally an internal OS component – not a service within the meaning of Article 6(7) DMA – and is therefore not a relevant benchmark for the non-discrimination requirements.

1596 Alphabet's response to the PFs, Doc ID […], paragraphs 343-344.

1597 Alphabet's response to the PFs, Doc ID […], paragraph 343.

1598 Alphabet's response to the PFs, Doc ID […], paragraphs 340 and 346, fifth bullet.

1599 For example, [OEM] appears to understand the measures as requiring *"unlimited access"* (See [OEM] response to public consultation, Doc ID […]. [OEM] appears to understand the measures as requiring to

ask to clarify that the measures do not require to grant unrestricted system-level access – which the Commission clarified in the final measures.[1600] To the extent that Alphabet's or OEMs' statements should instead be interpreted as a request to be free to selectively treat services in a discriminatory way and according to opaque rules or discretion, this would directly go against the requirements of Article 6(7). Alphabet may not allow OEMs to customise the operating system in a way that evades the requirements of Article 6(7) DMA (see Sections 4.6, 6.1, and 12.1).

11.4.8.2. Concerning proportionality

(1340) Alphabet claims that the measures for background execution are disproportionate. However, Alphabet does not substantiate why the measures would be more restrictive than necessary to achieve the objectives of Article 6(7) DMA. In particular, Alphabet fails to propose any less restrictive, yet equally effective means of complying with that provision. Nor does Alphabet explain and substantiate why the feature or a specific measure imposes a disproportionate burden on Alphabet. Instead, its arguments focus on questioning the scope of Article 6(7) and the actual need for interoperability – a line of reasoning which does not meet the legal standard for a valid proportionality claim, as explained in Section 13 below.

(1341) The Commission nevertheless addresses each of Alphabet's arguments in this respect.

(1342) The Commission first notes that, on proportionality too, Alphabet's arguments[1601] do not contain any reference to the specific measures that are allegedly disproportionate (with one exception[1602]). Instead, Alphabet's arguments apply generically and vaguely to the "Draft Measures". The result is that Alphabet's arguments that the measures are disproportionate are premised on imaginary and unspecified measures that are not those proposed in the PFs or specified in this Decision.

(1343) Alphabet appears to interpret the measures on background execution as requiring "special treatment" for AI-related apps compared to other apps/services which have a greater need for special treatment and will be harmed by the draft measures.[1603] This is wrong and nowhere in the measures, which simply require non-discriminatory treatment. Moreover, the Decision explains why background execution is especially important for AI-related services – which was confirmed by several third parties in

---

allow system-level *"background-execution privileges for user-downloaded apps"* (see [OEM] response to public consultation, Doc ID […]). These requirements are nowhere in the measures. In any event the final measures clarify that internal OS components can retain privileged access to background execution.

1600 For example, [OEM] requested that *"the Commission clarify that equal access to hardware resources does not mean unrestricted access"* and that the final measures ensure that OEMs retain the *"ability to manage the basic device experience — including but not limited to performance, power consumption, and thermal management"*, see [OEM's] response to public consultation, Doc ID […]. Similarly, [OEM] recommended to clarify that only *"system components deeply optimized by OEMs be granted unthrottled resource allocation"* (see [OEM's] response to public consultation, Doc ID […]). [OEM] asks to *"preserve OEM authority to apply non-discriminatory, transparent device-management decisions on battery, thermal and integrity grounds"* (see [OEM] response to public consultation, Doc ID […]). The final measures clarify all these aspects.

1601 Alphabet's response to the PFs, Doc ID […], paragraphs 346-347.

1602 Alphabet only refers to the measure requiring that non-discriminatory rules *"may not treat any specific service differently"*, expressing concerns that the measure would extend to internal OS components such as the user interface. This misinterprets the measure, as explained in recital (1337) above.

1603 Alphabet's response to the PFs, Doc ID […], paragraph 346, second bullet.

the public consultation[1604] and which Alphabet does not dispute – but nowhere does it suggest that it is important for those services only.

(1344) As explained above, the evidence used by the Commission are not "anecdotes",[1605] but are instead based on an in-depth review of requirements (and lack thereof) relating to background execution in Alphabet's CDD and [Alphabet's commercial agreements with OEMs] (see Section 11.4.3) – which are unaddressed by Alphabet – as well as publicly available information, minutes of meetings with third parties, and internal documents. Moreover, for the reasons described above, OEMs remain free to *"make specific low-level adjustments"* for their own devices.[1606] Alphabet appears to interpret the "prohibition on selective treatment" as a requirement to apply the exact same limitations to every type of device, app, and use case, which is not what the measures require. The prohibition simply seeks to prevent discretionary and discriminatory application of the rules. Non-discriminatory allocation of resources between the gatekeeper's services and third-party services is, by definition, the least restrictive measure to achieve contestability and fairness.

(1345) Alphabet considers that it would be *"disproportionate and unworkable"* to *"exert control over OEMs"* to ensure that Google Android devices comply with the measures.[1607] This argument is generic and not specific to background execution. As explained in Section 5.1.3, Alphabet already imposes hundreds of obligations on OEMs and approves OEMs' Android builds. The measures require Alphabet to take all actions of legal, contractual and technical nature, that are necessary to ensure that the implementation of Google Android on such devices complies with Article 6(7) – see Section 6.1.1. Alphabet's comment that *"Huawei uses an Android "fork" that is not controlled by Google and does not licence GMS"*[1608] is particularly confused. Huawei used to sell devices with Google Android and licence GMS, and Alphabet does not say Huawei's actions were prohibited or would be now prohibited by the CDD or the GMS Requirements. In fact, Sections 11.4.3 and 11.4.6 argue precisely that such requirements leave room for the type of discrimination explained in relation to Huawei, which Alphabet does not dispute.

(1346) Finally, the fact that Android may provide better support for background execution than rival operating systems such as Apple iOS does not mean that it already provides effective interoperability, and the Commission has never concluded that it does – contrary to what Alphabet suggests.[1609] In fact, the measures for background execution in this Decision are very similar to the ones that apply to Apple in the

---

1604 [AI service provider] explained that *"[f]ormal permission to run an app or model is not enough" since "a third-party AI service may still be non-competitive if it is killed in the background"*, see [AI service provider's] submission to the public consultation, Doc ID […], Section 3. [Third party] submitted that *"[b]ackground execution is not a peripheral technical capability – it is the foundational infrastructure on which virtually every advanced AI assistant feature depends"* and provided examples of AI-based use cases that rely on background execution – see [third party's] submission to the Public Consultation, Doc ID […], Section 13. [AI service provider], [AI service provider], and [third party] also expressed support for the proposed measures on background execution – see [AI service provider's] submission to the public consultation, Doc ID […], page 3; [AI service provider's] submission to the public consultation, Doc ID […], page 4; See [third party's] submission to the public consultation, Doc ID […], page 6.

1605 Alphabet's response to the PFs, Doc ID […], paragraph 346, first bullet.

1606 Alphabet's response to the PFs, Doc ID […], paragraph 346, third and fifth bullets.

1607 Alphabet's response to the PFs, Doc ID […], paragraphs 346, fourth bullet.

1608 Alphabet's response to the PFs, Doc ID […], paragraphs 346, fourth bullet.

1609 Alphabet's response to the PFs, Doc ID […], paragraphs 335-346, last bullet.

Connected Devices decision.[1610] In any case, the Commission does not have an obligation to specify the exact same measures in different Specification Decisions that apply to different gatekeepers, contrary to what Alphabet seems to suggest;[1611] in fact, specification is precisely meant to account for variations of services within a single category of core platform services.[1612]

(1347) The Commission therefore concludes that there is an interoperability gap with respect to the background execution feature of Google Android. As a result, third parties and in particular user-downloaded apps, cannot perform important functionalities that are expected of an AI assistant which places them sub-par in comparison with Google's own AI-powered services. As a result, the Commission considers that it should specify how Alphabet must fill this gap to comply with Article 6(7) DMA.

*11.4.9. Measures that Alphabet should implement*

(1348) To meet the requirements of Article 6(7) DMA, Alphabet should implement the measures set out in Section 4.3 of the Annex to this Decision. The interoperability solution for the background execution feature, as described in Section 11.4.2 of this Decision, must be equally effective as that available to any of Alphabet's own services.

(1349) Alphabet should ensure that third parties have the same access to background execution as Alphabet's services, directly or indirectly. As explained in Section 11.4.3.3, there are several ways in which Alphabet can indirectly favour its services without directly exempting them from restrictions on background execution. To ensure equally effective access, Alphabet should define transparent, objective, precise and non-discriminatory rules that also apply to Alphabet's services, including for use cases that Alphabet does not offer. Alphabet should ensure non-discrimination with respect to any service, not only for AI services. First, Article 6(7) does not provide for any limitation on beneficiaries. Second, as illustrated by the examples in Section 11.4.5, access to background execution by a service that is not an AI assistant (such as Google Photos) can benefit an AI assistant (such as Gemini).[1613]

(1350) The rules may not treat any specific service differently, for example by restricting or exempting a specific service, unless the user has provided consent for the specific service. As explained in Section 11.4.3.2, if an app is selectively restricted by Alphabet or by OEMs without user permission, the app might misbehave for reasons that are not clear or agreed on by the user, and the user might wrongly attribute the malfunctioning to the app rather than to Alphabet or the OEM. Similarly, an exempted service is inherently privileged compared to non-exempted services, which should only be allowed with user permission. This should not apply to internal OS components, which generally need to be privileged to ensure system stability.

---

1610 See Case DMA.100203 (Apple – Operating systems – iOS – Article 6(7) – SP – Features for Connected Physical Devices), Annex, section 4.

1611 To the extent that Alphabet is right in saying that background execution is already more interoperable on Android than on iOS, then Alphabet should be able to implement the measures with more limited efforts, since the measures essentially just require non-discrimination in access to the feature.

1612 Recital 65 DMA.

1613 Alphabet's claims that Gemini is not whitelisted therefore ignore the other Alphabet services and in any event do not address the examples of how Gemini indirectly benefits from privileged background execution in other Alphabet apps. See Alphabet's response to the PFs, Doc ID […], paragraph 338.

(1351) Similarly, Alphabet should ensure that third parties may prompt the user to grant their services access to the same level of background execution that other services – including Alphabet's services – have access to. This is inherently necessary to ensure equally effective access as Alphabet's services. Moreover, while obtaining exemption from the user should not be the primary method to access sufficient background execution, it is critical for some apps[1614] – which is why the exemption mechanism exists in the first place. For this reason, it should be possible for the user to grant such exemption in a simple way.

(1352) Alphabet may allow users to limit the use of background execution by any app. User choice and transparency towards the user are generally positive for interoperability and increase the openness of the ecosystem, provided that the user can take the same action with the same effect for any app, including Alphabet's apps.

(1353) Finally, third parties have expressed that one of the main obstacles to implement always-on hotword detection on connected devices are the limitations on background execution.[1615] Alphabet should therefore ensure that third parties have access to the same degree of background execution as Alphabet to enable this use case, and should publish the necessary documentation to explain how to use the relevant Google Android APIs to implement it.

(1354) For the purpose of ensuring that effective interoperability continues in the future, Alphabet should also provide effective interoperability with any future updates, including new functionalities, of the background execution feature insofar as they are available to Alphabet's own services or hardware. To this end, the general measures in Section 12 of this Decision apply, including in particular the measures concerning future updates and new functionalities set out in Section 12.9 in this Decision.

*11.4.10. Timing for implementation of the measures*

(1355) To meet the requirements of Article 6(7) DMA, and taking into account Alphabet's proposal and the available evidence,[1616] Alphabet should implement the measures in the Annex (Section 4.3) according to the timelines indicated therein.

## 12. MEASURES FOR ALL FEATURES

(1356) According to Article 6(7) DMA, interoperability provided pursuant to this provision needs to be effective. As described in Section 4.5 above, this means that interoperability solutions implemented for the features listed in Sections 8 to 11 of this Decision – and for any future updates of these features, including new functionalities – should be granted in a technically sound and workable manner for third parties, equally effective to the solutions available to Alphabet, under equal conditions, and without any undue obstacles. To achieve such effectiveness, Alphabet should implement certain measures for all the features listed in Sections 8 to 11 of this Decision.

### 12.1. Implementation across the Google Android ecosystem

(1357) Consistent with the language and aim of Article 6(7) DMA, interoperability must be provided to third parties under equal conditions to the interoperability solution available to Alphabet's own services and hardware. Accordingly, Alphabet must

---

1614 See Section 11.4.3.

1615 See [AI service provider's] submission to the public consultation, Doc ID […], Section D, page 4. See [AI service provider's] submission to the public consultation, Doc ID […], Section 1, page 1-2.

1616 Technical meeting Alphabet/Commission of 12 June 2026, Doc ID […], slide deck page 13. Alphabet's reply to RFI PO 21 of 8 June 2026, Doc ID […], question 6, paragraph 17.

provide the interoperability solutions across the Google Android ecosystem, regardless of the OEM, in a manner that ensures a level playing field across Google Android mobile devices,[1617] irrespective of the device model or form factor.

(1358) *First,* Alphabet must take all necessary actions, including of technical, contractual and legal nature, to ensure that Google Android mobile devices effectively receive the interoperability solutions. Alphabet makes available the source code to OEMs on a regular basis and publishes the source code in AOSP twice a year. Before an OEM can distribute their update with the newest Android version, OEMs must certify their update (i.e. Android builds) with Alphabet (see Section 5.1.3). To ensure effective interoperability, Alphabet must take all necessary actions to allow OEMs to certify and distribute their updates to end users as quickly as possible, in principle no later than the timelines set out in the Annex to this Decision. Furthermore, Alphabet must ensure that OEMs distribute the relevant update to end users' devices as soon as Alphabet has certified the update. Effective interoperability also entails the obligation to ensure that users are encouraged to install the relevant update containing the interoperability solutions. As explained at length in Sections 4, 5 and 6 above, Alphabet has the legal duty and the means at its disposal, due to its control over Google Android and the distribution thereof, to take all such necessary actions.

(1359) *Second,* to ensure that interoperability is provided under equal conditions, the interoperability solutions should be available on Google Android mobile devices, both new devices and existing devices ("legacy devices") approved by Alphabet before the implementation deadlines set out in the Annex, on which the features exist and are available to Alphabet, and which will be updated to the relevant Android version specified in the timeline section of each measure in the Annex. Without implementation on legacy devices, Alphabet would retain a substantial advantage in enjoying feature access on the existing installed base of Google Android mobile devices, where interoperability would be unachievable for third parties, for example since feature access would remain tied to system privileges and OEM preinstallation there.

(1360) Updates can reach legacy devices via three main channels: (i) an update to a new Android version (e.g. updating a device from Android 16 to 17); (ii) via component-level updates, where Alphabet delivers specific security and feature fixes (e.g. via Google Play system updates, also called Mainline, or via Google system services updates)[1618] directly to devices, independently of OEMs pushing an update; and (iii) through Jetpack libraries,[1619] which are software libraries that can provide certain features to app developers on older Android devices.

(1361) The Commission noted in its PFs that, in addition to the first channel, the second and third channel would allow Alphabet to ensure the rollout of the interoperability solutions on legacy devices in an efficient manner.[1620]

(1362) In its response to the PFs, Alphabet claimed that implementing the interoperability solutions on all legacy devices via these two channels is not possible, and the

---

1617 Google Android mobile device includes any mobile device that Alphabet certified as a certified Android device or a Google Play supported Android device.

1618 See, respectively, [Alphabet's commercial agreements with OEMs], Section 4, Alphabet internal document, Document [Alphabet internal document], Doc ID […], page 174, submitted in Alphabet's reply to RFI AI 2 of 23 September 2025, and https://support.google.com/product-documentation/answer/11462338, Doc ID […], accessed on 20 April 2026.

1619 Google for Developers (2026), 'AndroidX overview | Jetpack', Android Developers website, 6 March, https://developer.android.com/jetpack/androidx, Doc ID […], accessed on 19 April 2026.

1620 PFs, paragraph 919.

solutions envisaged by the Commission would either not achieve the necessary changes or would create instability and compatibility risks.[1621] Alphabet claimed that the implementation of the measures requires large-scale changes to platform-level APIs, and to fundamental operating system behaviour, such as altering permissions. Alphabet submitted that many (if not all) of the changes required by the measures will need to be implemented via AOSP changes, which are propagated to legacy devices via device upgrades to the newest Android version.[1622] According to Alphabet, the measures involving changes to the CDD policies (e.g., for background execution, Section 11.4 above) will follow the same timeline, as CDD updates accompany the AOSP releases and need to be tested against the respective AOSP release.[1623] Alphabet also argued that it would be impossible to ensure the implementation of the interoperability solutions on legacy devices due to its lack of control over the behaviour of OEMs; and over-the-air software updates would risk breaking the devices' functionality.[1624] Alphabet explained that, while backporting some APIs might be theoretically possible in some cases, and [confidential information concerning backporting by OEMs], [description of Alphabet's experience with backported APIs].[1625]

(1363) The Commission takes note of Alphabet's claims that the type of changes required to implement interoperability solutions for the features covered in this Decision can generally only be delivered via new Android releases. The Commission considers that, to ensure the effectiveness of the measures in practice, Alphabet must, at a minimum, ensure the following:

(a) The interoperability solutions should be included in the relevant Android version specified in each measure in the Annex, within the deadlines specified in the Annex for each measure;

(b) When an OEM sells or updates a device to use a new Android version, that Android version should support all the relevant interoperability solutions pursuant to the previous point; and

(c) If technically feasible and appropriate, after the interoperability solution for a feature has been implemented in the new Android version, Alphabet should backport the interoperability solutions to older Android versions that still have significant adoption.

(1364) *Third,* to ensure effectiveness, Alphabet's interoperability solutions should not put the burden on third parties to build and maintain device-specific implementations. The interoperability solutions must be uniform across all Google Android mobile devices. In particular, all interoperability solutions should be implemented in a way that avoids differences in how third parties must access or integrate interoperability solutions across different Google Android mobile devices, as Alphabet already ensures for other Android SDK APIs. Third parties must be able to use the same interoperability solutions consistently via the same APIs across Google Android versions, devices, and form factors. Alphabet may achieve this, for example, by defining Android SDK APIs and using Android support libraries, such as Android Jetpack, to define a consistent API across different Android versions and potentially

1621 Alphabet's response to the PFs, Doc ID […], paragraph 349; Alphabet's reply to RFI PO 21 of 8 June 2026, Doc ID […], paragraph 10.

1622 Alphabet's response to the PFs, Doc ID […], paragraph 349; Alphabet's reply to RFI PO 21 of 8 June 2026, Doc ID […], paragraphs 10, 12, 19.

1623 Alphabet's reply to RFI PO 21 of 8 June 2026, Doc ID […], paragraph 13.

1624 Alphabet's response to the PFs, Doc ID […], paragraph 349.

1625 Alphabet's reply to RFI PO 22 of 17 June 2026, Doc ID […], paragraphs 14, 15, 18, 20, and 22.

different implementations. If different implementations are required due to hardware differences across OEM devices, Alphabet should take the largest part of the implementation burden.

(1365) The Commission considers that Alphabet has the means to ensure such consistency and compatibility across all Google Android mobile devices. As noted above (see Section 5.1), to obtain the right to distribute Google Android-branded devices, OEMs must enter into and comply with a number of agreements, policies and requirements on how to implement the Android operating system in order to be permitted to use the Android brand and market a device as Android-compatible.[1626] Through these requirements, Alphabet can ensure effective interoperability across Google Android mobile devices.[1627]

(1366) Furthermore, to the extent that Alphabet allows OEMs to apply any customisations that affect the features in Sections 8-11 of this Decision or their implementation, Alphabet should ensure that the custom implementation complies with all the measures in the Annex. The Commission notes that, as explained in Section 6.1 above, the measures attached to this Decision do not limit OEMs' ability to customise, provided the customisation does not interfere with the obligation to allow effective interoperability with the features subject to the measures.

(1367) Alphabet submitted that many of the changes that OEMs can introduce are opaque and difficult for Alphabet to monitor.[1628] Contrary to this claim, Alphabet does have, and already uses, means to verify the compliance of the OEMs' customisations, and, in any event, it must do so to ensure effective interoperability. As explained in Section 6.1 above, the CDD and other licensing agreements already set detailed compliance requirements concerning device hardware, software features, permission model, and the implementation of security or privacy safeguards, which OEMs must accept to obtain Android certification. OEMs must also pass several test suites to demonstrate that their devices comply with Alphabet's requirements.[1629] Where necessary, Alphabet shall ensure that, for instance, the CDD and any other licensing agreements are amended to incorporate the requirement that OEM customisations shall not undermine effective interoperability.[1630] Furthermore, Alphabet must ensure that such contractual changes are also applied in practice. For instance, Alphabet approves OEMs' software builds for each Google Android device. Alphabet therefore already has the appropriate means to verify that OEMs' customisations comply with the measures, and in any event, the need to ensure effective interoperability requires it to put such means into place.

(1368) Therefore, to attain the objectives of the interoperability obligation set out in Article 6(7) DMA, Alphabet shall implement the measures in the Annex in the Section "Implementation across the Google Android ecosystem" (Section 5.1).

---

1626 See https://source.android.com/docs/compatibility/overview, accessed on 23 March 2026, Doc ID […].

1627 For instance, if the hardware of a Google Android mobile device can accommodate multiple concurrent hotwords, the hardware ability should not be artificially or unduly restricted.

1628 Alphabet's response to the PFs, Doc ID […], paragraph 350.

1629 For the CDD, OEMs must pass the CTS, for the [requirements for Alphabet's commercial agreements with OEMs] and in addition there is a VTS (Vendor test Suite) and a BTS (Build Test Suite). See [Alphabet's commercial agreements with OEMs], Document […], Doc ID […], Section 3.6.

1630 In this regard, it is relevant to note that Alphabet does not establish any bounds on the extent to which it limits OEM customisations via the CDD: that OEMs are allowed to customise every feature of AOSP unless expressly limited by the CDD, would theoretically remain correct even if the CDD limits many or all customisations.

## 12.2. User consent

(1369) As explained in Section 4.8.3, Alphabet may give users the ability to consent in an informed manner to grant feature access to third parties. Alphabet may therefore implement technical measures (e.g. system-level prompts) to ensure that features are accessible only with user consent, that users are informed about feature access (for example, through privacy indicators informing the user that an app is currently accessing a feature, or privacy dashboards listing when apps accessed features) and that users can revoke access to the feature for any specific service. As explained in Section 4.8.1, while such measures are not integrity measures by themselves, they may be complemented by integrity measures that ensure that choices made by end users are respected.

(1370) Any requirements for feature access, such as user consent and the use of privacy indicators, and any corresponding technical measures should be based on transparent, objective, precise, and non-discriminatory conditions that also apply to Alphabet's services and hardware. In particular, such requirements should apply equally to user-downloaded apps and preinstalled apps. Third-party contributors to the public consultation have expressed their support for granular, understandable, and revocable user consent as well as clear indicators and audit trails (e.g. dashboards) for feature access,[1631] provided they apply equally to Alphabet's and third parties' services.[1632]

(1371) Alphabet claimed that there is no difference between the privacy indicators applied to pre-installed and downloaded applications, as the privacy indicators depend on the modality of access, and apply equally to all apps using the applicable APIs.[1633] As explained above concerning LPH/LPNH, hotword, and ambient data (see Sections 8.2, 8.3 and 9.4), the delta in showing privacy indicators is closely tied to pre-installation, which does not enable third parties to access the feature under equal conditions (see Section 5.4.1) and in any event is not a strictly necessary and proportionate integrity measure (see Section 6.3). Pre-installed apps can exclusively use dedicated APIs where there is no user consent requested, or no privacy indicators shown. Therefore, while Alphabet's claim that all applications using the applicable APIs display privacy indicators in the same manner is technically correct, Alphabet's own services – by virtue of being pre-installed – can use APIs that enable functionalities for which a privacy indicator is not required, while third parties do not enjoy the same ability for such access. Similarly, Alphabet's reasoning is correct in that measures such as privacy indicators may depend on the modality of access but stops short of recognising that this difference in modality must be defined on objective and non-discriminatory grounds. For example, the display of an indicator can be made conditional on whether microphone input is processed using an isolated process, as long as that pathway is attainable in a non-discriminatory manner for third parties. However, a third party's pathway to such access without indicator via an isolated process depends on OEM preinstallation,[1634] making such a condition currently subjective and discriminatory.

(1372) Alphabet also asserted that it needs to draw a clear distinction concerning user journeys and ease of use between its first-party apps and third parties. According to

---

1631 [Third party's] submission to the public consultation, Doc ID […], page 3. [Third party's] submission to the public consultation, Doc ID […], pages 3-5. [Third party's] submission to the public consultation, Doc ID […], page 2. [OEM's] submission to the public consultation, Doc ID […], pages 3-4.

1632 [AI service provider's] submission to the public consultation, Doc ID […], page 3. [OEM's] submission to the public consultation, Doc ID […], page 4.

1633 Alphabet's response to the PFs, Doc ID […], paragraph 351.

1634 See Section 8.3.3.1 and 8.3.6.5 on hotwords, Section 9.3.3.3 and 9.4.3 on PCC and ambient data.

Alphabet, sharing data with third parties is more dangerous than sharing it with Alphabet's first-party apps or preinstalled apps.[1635] Some third-party contributors to the public consultation (including some contacted by Alphabet) echo that Alphabet's assistant should enjoy privileged access.[1636] Such a privilege would disproportionately interfere with the objectives of the DMA and Article 6(7) thereof. Safeguards that seek to impose higher standards than the gatekeeper requires or accepts in relation to its own services or hardware, or safeguards that are subject to conditions that are subjective, exclusively within the gatekeeper's control, or even unachievable by third parties, such as being the gatekeeper or trusted by it, cannot be considered proportionate.[1637] Contrary to Alphabet's understanding, the measures do not mandate any "forced reduction in transparency".[1638] The measures simply require that the same user journeys and ease of use are applied across first-party and third-party services.

(1373) Therefore, to attain the objectives of the interoperability obligation set out in Article 6(7) DMA, Alphabet shall implement the measures in the Annex in the Section "User consent" (Section 5.2).

### 12.3. Integrity

(1374) Pursuant to Article 6(7) DMA, in the implementation of the specified measures, Alphabet may take strictly necessary and proportionate measures to ensure that interoperability does not compromise the integrity of the operating system, virtual assistant, hardware and software features.

(1375) Pursuant to Article 6(7) DMA, and as explained in Section 4.8.4, any integrity measure should be duly justified and based on transparent, objective, precise, and non-discriminatory conditions that also apply to Alphabet's services and hardware. Pursuant to the second subparagraph of Article 6(7) DMA, Alphabet should only impose conditions and take integrity measures that reflect a genuine integrity risk and do so in a consistent and systemic manner. Similarly, Alphabet should only apply conditions the compliance with which is capable of being independently verified and not exclusively within the gatekeeper's control. An integrity measure cannot be considered strictly necessary and proportionate if it seeks to achieve a higher integrity standard than the one that Alphabet requires or accepts in relation to its own services or hardware. In other words, Alphabet must not impose a higher level of integrity on third parties than it applies to itself.

(1376) It is also important in practice that no integrity measure is implemented in a way that would undermine effective compliance with the measures listed in the Annex or with Alphabet's obligations pursuant to the DMA, including for instance by subverting end users' or business users' autonomy, decision-making, or free choice via the structure, design, function, or manner of operation of a user interface or a part thereof.

(1377) When applying integrity measures, in line with the framework set out in Section 4.8.4, the following principles should apply:

---

1635 Alphabet's response to the PFs, Doc ID […], paragraphs 352, 353.

1636 [Third party's] contribution to the Commission's public consultation, Doc ID […], page 2. [Third party's] contribution to the Commission's public consultation, Doc ID […], page 3. For the contributors who were contacted by Alphabet, see Section 2.4.

1637 See Section 4.8.4 read together with footnote 232.

1638 Alphabet's response to the PFs, Doc ID […], paragraph 353.

(a) The measures should be consistent with the approach that Alphabet already applies in similar or comparable cases. In particular, a measure is not consistent if Alphabet allows the same or similar (or higher) integrity risk to persist in other areas.

(b) The measures should apply equally to the Alphabet's services and to third parties' services. Moreover, measures must apply equally to pre-installed and user-downloaded apps. Preinstalled apps may not enjoy better feature access, e.g. be subject to opt-outs where user-downloaded apps would be subject to opt-ins. Since preinstallation does not per se change the nature of app, a measure that treats these two categories differently would be inherently discriminatory.

(c) The measures should be based on objective and verifiable evidence showing the existence and magnitude of the integrity risk and effectiveness in reducing such risk. In that respect, evidence regarding how other operating systems deal with the same or similar integrity risk can be relevant.

(d) The measures should not impose limits on the beneficiaries, apps, or use cases for feature access (see Section 12.4). This includes not limiting an app to a single purpose. In principle, there should be no need for measures that verify whether the feature is (appropriately) used or required, in particular since this can be addressed through user authorisation.

(e) The measures should not impose any commercial or financial requirements on the beneficiaries (e.g. letters of credit), as these would impose undue burden on third parties where it is on the gatekeeper to implement integrity measures and not directly address the threat to integrity.

(f) The measures should not impose functional requirements beyond those strictly necessary for integrity. In particular, they should not assume or require that a certain technology is used. This would limit the ability to innovate on better technologies for using the feature. For example, feature access cannot be conditioned on the use of artificial intelligence technology.

(g) The measures should be enacted only for as long as is necessary, and adapt to technological evolution, including changes to the operating system. This ensures that the measures remain strictly necessary and proportionate over time and are not superseded by less restrictive measures.

(h) The measures should be grounded in established industry standards and practices.[1639] Such measures are more likely to be transparent, objective, precise, non-discriminatory, and not exclusively under the gatekeeper's control.

(1378) In addition, to ensure, first, a proper monitoring of the effectiveness of the measures specified in a decision pursuant to Article 8(2) DMA and, second, that the Commission is in a position to assess the impact of any potential integrity measures on the effectiveness of the resulting solutions ahead of their implementation, the Commission considers it necessary to put in place a reporting obligation whereby Alphabet shall inform the Commission in writing of any integrity measure that Alphabet intends to take. This is necessary because integrity measures are not always

[1639] For example, the OWASP Mobile Application Security Verification Standard (MASVS), to the extent its guidelines address integrity. See https://mas.owasp.org/MASVS/, accessed on 20 April 2026, Doc ID […]. The absence of a measure under consideration from any such standard may be an indication that the measure is not sufficiently necessary.

easily observable but may have an impact on how third parties can benefit from the interoperability solution. In doing so, and in line with the requirements of Article 6(7) DMA, Alphabet shall provide a relevant justification for the strict necessity and proportionality of the contemplated integrity measures.

(1379) However, there is no need to report integrity measures that are unlikely to affect effective interoperability. Alphabet's claim that the requirement on integrity reporting is unworkable and unduly burdensome,[1640] is therefore overstated, as Commission does not require reporting in relation to each and every single integrity measure. In any event, Alphabet likely already keeps due track of all the integrity measures it has taken and intends to take, alongside the reasons justifying them, therefore making it unlikely that reporting them to the Commission causes an undue additional burden. The measures also provide for flexibility in terms of timing, allowing Alphabet to address "*emerging integrity risks*" without delay, counter to Alphabet's claim that they are "*subject to four weeks of delay and review by the Commission*".[1641]

(1380) Alphabet opposed the measures in Section 5.3 of the Annex, claiming that it should play an active and continuous role in responding to integrity risks, that it cannot be required to demonstrate that every single measure it applies will be effective in addressing risks, that it must be able to impose on beneficiaries to use a specific technology, and that established industry standards do not always exist, especially for developing technology such as AI services.[1642] The Commission notes that Article 6(7) explicitly requires that integrity measures are *"duly justified by the gatekeeper"*. The other points raised by Alphabet (including the involvement of third-party verifiers) are discussed in Sections 4.8 above and 12.4 below.

(1381) Therefore, to attain the objectives of the interoperability obligation set out in Article 6(7) DMA, Alphabet shall implement the measures in the Annex in the Section "Integrity" (Section 5.3).

### 12.4. Eligibility of beneficiaries, applications and use cases

#### *12.4.1. Access to an OS feature should not be limited to specific use cases or beneficiaries*

(1382) Article 6(7) DMA does not provide for any limitations as to the beneficiaries, apps, products and use cases for interoperability insofar as a feature is available to, or used by, Alphabet and intended to be used by a third party that is eligible under Article 1(2) DMA.

(1383) One of the key objectives of the DMA is to foster and promote innovation in the digital sector and remove barriers that could prevent market participants from innovating.[1643] Specifically, Article 6(7) aims to allow third parties to develop and provide innovative services or hardware complementing or supporting the designated operating system offered by a gatekeeper.[1644] Such innovation can only take place if interoperability is not limited to a select group of beneficiaries, apps, products or use cases.

(1384) In particular, it cannot be left to the discretion of the gatekeeper to decide which third parties, apps, products and use cases can benefit from the interoperability mandated by Article 6(7) DMA. Importantly, this implies that the beneficiaries of

1640 Alphabet's response to the PFs, Doc ID […], paragraph 356.
1641 Alphabet's response to the PFs, Doc ID […], paragraph 356.
1642 Alphabet's response to the PFs, Doc ID […], paragraph 355.
1643 See, for example, Recitals 4, 32 and 107 DMA.
1644 See recital 57 DMA.

interoperability cannot be limited to those third parties offering exactly those services and hardware as are offered by the gatekeeper itself. Such a limitation would leave little room for innovation by third parties, as it would always be the gatekeeper deciding which new services or hardware could be launched, effectively capping innovation at the level of the gatekeeper and only from the moment when the gatekeeper decides to launch its first-party product.

(1385) Providing interoperability without restrictions as to the beneficiaries, apps, products or use cases reflects the wording and aim of Article 6(7) DMA and allows third parties to develop novel and innovative products that the gatekeeper (Alphabet in this case) may not yet offer. Conversely, imposing restrictions on which third parties can interoperate with a feature, and for which use, could significantly reduce the commercial incentives of third parties to invest and make use of the interoperability solution.

(1386) In the context of these proceedings, Alphabet argued for the need to restrict the beneficiaries, applications, or use cases when it comes to developing interoperability with features subject to this decision. *First,* as a procedural point, Alphabet argued that the Decision opening the proceedings allegedly placed a boundary on the measures, to AI services.[1645] The Commission first notes that, as a matter of principle, the procedural point where the final scope of the proceedings is set out is the issuance of PFs and not the decision opening the proceedings. In addition, Alphabet seems to confuse the preliminary scope of features in the decision opening the proceedings with the scope of the beneficiaries, which is not limited in any way.[1646] While these proceedings relate to a number of features of Google Android that are relevant for AI services, these proceedings do not aim, as such, to limit the scope of beneficiaries, which is not limited by Article 6(7). Indeed, any such restriction would leave little room for innovation by third parties who may find a use for the feature in novel types of services. In addition, any such limitation would be at odds with the dynamic nature of digital markets where new technologies and business models regularly emerge.

(1387) *Second,* Alphabet claimed that it should be able to restrict use cases and beneficiaries of the features to prevent any "nefarious, illegal, or unethical" activities such as fraud.[1647] This point is addressed in Sections 4.7 and 4.8 of this Decision, where the Commission explains that, as matter of principle, it is not up to the gatekeeper in relation to its operating system to regulate and enforce what business users can and cannot do when they provide a service to Android end users, and that restricting access in an arbitrary way would undermine contestability, fairness, and innovation.

(1388) For the reasons explained in Sections 6.1 and 6.3, the Commission considers that Alphabet should not limit access to features that are in scope of Article 6(7) to apps that are preinstalled and granted system privileges by the OEM.

(1389) However, for the reasons explained below, and in view of the nature of some features in scope of this Decision, the Commission considers that Alphabet may, under exceptional circumstances, restrict access to those features to third parties that meet certain eligibility conditions.

---

1645 Alphabet's response to the PFs, Doc ID […], paragraphs 41-42.

1646 Alphabet's response to the PFs, Doc ID […], paragraphs 41-42.

1647 Alphabet's response to the PFs, Doc ID […], paragraph 153, fifth bullet.

*12.4.2. Restricting feature access to a limited group of beneficiaries via OS-level eligibility conditions*

(1390) During the proceedings, Alphabet submitted that, due to the very sensitive nature of the features subject to these proceedings, it should be entitled to restrict access to only a limited group of third parties that satisfy some eligibility conditions.[1648] These restrictions would be implemented by imposing certain policy-based requirements and prohibitions that would go far beyond the technical protections of the operating system and the generic contractual terms and conditions that apply to all Android developers.[1649] Alphabet would instead impose policy-based conditions in the same way as it imposes policy-based conditions for apps distributed through the Google Play Store. However, in this case, such eligibility conditions would apply at the OS level, i.e. beyond the Google Play Store: a third party would be able to access the restricted features only after Alphabet verified that it meets the eligibility conditions, even if the app is distributed through alternative channels, such as third-party app stores. Alphabet considers that this is justified, given the allegedly higher malware risk for non-Play Store distributed apps.[1650]

(1391) In practice, Alphabet's proposal would allow Alphabet to: (i) define the eligibility conditions; (ii) set up a programme where it is the sole adjudicator of whether the third party meets all eligibility conditions; and (iii) enforce the eligibility requirements at the operating system level, i.e. prevent or revoke access to those features for every app that is not deemed eligible by Alphabet.[1651] Alphabet appears to consider that this is strictly necessary and proportionate to protect against integrity, security, and privacy risks, and is therefore allowed under the integrity clause of Article 6(7) and the integrity and security clauses of Article 6(4).[1652]

(1392) To this end, Alphabet proposed a system of "registered AI virtual assistants" ("RAIVAs") whereby Alphabet would be the certification authority of all beneficiaries based on a number of eligibility criteria ("Proposed Eligibility Conditions") such as functional eligibility, agent functional security, agent privacy and transparency, agent safety, agent app security, model requirements and developer reputability.[1653] To demonstrate that their service meets the Proposed Eligibility Conditions, the third party would have to go through several layers of vetting, including: (i) obtain a number of external certifications;[1654] (ii) provide information to Alphabet to carry out additional verification;[1655] (iii) submit the app to Alphabet

---

1648 See for example: Alphabet's response to the PFs, Doc ID […], paragraph 153, sixth bullet and paragraph 355, last bullet; Alphabet's submission […], Doc ID […]; Alphabet's submission "Draft Measures: Priority issues", Doc ID […], pages 3-4. Alphabet's submission "Google Restricted Features Mark-up" of 15 June 2026, Doc ID […].

1649 Android Software Development Kit License Agreement, see https://developer.android.com/studio/terms, Doc ID […], version of 28 April 2026.

1650 Alphabet's response to the PFs, Doc ID […], paragraph 153, sixth bullet.

1651 Alphabet's submission Doc ID […]; Alphabet's submission "Google Restricted Features Mark-up" of 15 June 2026, Doc ID […], paragraph 9, [description of engagement with the Commission].

1652 Alphabet's response to the PFs, Doc ID […], paragraph 355.

1653 See [Alphabet's submission], of 19 May 2026, Doc ID […].

1654 These include: Application Defence Alliance (ADA) AI Agent Profile, ADA Mobile App Security Assessment (MASA) Level 2 certificate; ISO 27001 for data confidentiality; ISO 42001 for ensuring the safe, secure, and responsible development and use of AI systems; SOC 2, or BSI C5 for EU for cloud security. See [Alphabet's submission] of 19 May 2026, Doc ID […].

1655 See Alphabet's submission "Draft Measures: Priority issues", Doc ID […], page 3 and Alphabet's submission "Google's mark up on the restricted features measures", Doc ID […], paragraphs 8-10.

and allow it to perform "operational tests" to ensure that the app is capable of using the features.[1656]

(1393) As explained in the following sections of this Decision, the Commission considers that Alphabet's proposal to restrict the scope of beneficiaries of interoperability via OS-level eligibility conditions is not a strictly necessary and proportionate measure under the integrity clause of Article 6(7) and the integrity and security clauses of Article 6(4) DMA. However, restricting the scope of beneficiaries may be appropriate under exceptional circumstances where the restriction is also strictly necessary to ensure effectiveness of interoperability, and subject to minimum safeguards to ensure that the objectives of the DMA are not undermined.

*12.4.3. Alphabet's proposal to restrict the beneficiaries via an eligibility programme is not strictly necessary and proportionate to protect integrity and security under Article 6(4) and Article 6(7) DMA*

(1394) *First*, Alphabet's Proposed Eligibility Conditions include several conditions that relate to privacy and security, but do not relate to integrity, as they are not aimed at protecting the OS or the feature against manipulation.[1657] In fact, Alphabet has not substantiated why any of the proposed conditions relates specifically to integrity of the operating system or of the features.[1658] As explained in Sections 4.8 and 6.3.2, the integrity clause in Article 6(7) does not extend to measures that relate to privacy and security – unless they also relate to integrity. Just like Alphabet's response to the PFs (see Section 6.3.2), Alphabet's proposal of eligibility criteria disregards the legal framework set out in the PFs on the scope of integrity measures, and simply postulates that the criteria *"[help] manage risks to integrity and privacy"*.

(1395) *Second,* as concerns Alphabet's argument based on the security clause in Article 6(4), the Commission notes that Article 6(4) only allows the gatekeeper to take strictly necessary and proportionate measures *"enabling users to effectively protect security"* – and is explicitly limited to measures and settings other than default settings.[1659] Alphabet's proposal for eligibility conditions would apply by default, rather than as an optional setting enabling users to effectively protect security. Furthermore, Alphabet may not simply import security justifications into Article 6(7) via Article 6(4). The security measures that Alphabet may implement under Article 6(4) need to address the specific security implications of installing apps from alternative distribution channels, such as the risk of malware – not the security implications of interoperability with specific features. Finally, Article 6(4) does not have a clause allowing the gatekeeper to take privacy measures, unlike other DMA

---

1656 Alphabet's submission "Google Restricted Features Mark-up" of 15 June 2026, Doc ID […], paragraph 4; Alphabet's email with subject "Re: DMA.100220 - Measures on restricted features - without prejudice" of 15 June 2026, Doc ID […].

1657 For example, Alphabet's eligibility conditions include: [conditions to ensure that the candidate AI virtual assistant is secure against AI threats from malicious actors; conditions aimed at ensuring that the candidate assistant handles data in a transparent manner and with consent; conditions aimed at ensuring that the candidate assistant is secure against common mobile threats]. See [Alphabet's submission] of 19 May 2026, Doc ID […]. These conditions clearly relate to security and privacy, not to integrity: they are aimed at constraining how the features are used, not at protecting either the OS or the features against manipulation or impairment.

1658 While Alphabet's proposed criteria include conditions relating to integrity [such as the integrity of input, data, and processing], these are all aspects that relate to the integrity of the third-party service – not of the OS or of the features.

1659 Recital 50 DMA states: *"The gatekeeper should be prevented from implementing such measures as a default setting or as pre-installation."*

obligations such as Article 7(9) on interoperability for number-independent interpersonal communications services.

(1396) *Third,* while Alphabet submitted that malware is collectively more prevalent among apps distributed via channels other than the Google Play Store,[1660] it did not substantiate why Alphabet's eligibility conditions would be necessary and proportionate for all apps distributed via any channel. For example, some app stores only accept open-source apps – whose source code is public and can be reviewed for malware by everyone – and employ technical measures to ensure that the app downloaded by users actually corresponds to the public source code.[1661] Alphabet did not substantiate why eligibility conditions should apply as a blanket measure to all app stores, independently of their security profile and their track record in terms of malicious apps.[1662]

(1397) *Fourth*, nothing in the DMA suggests that the gatekeeper is allowed to restrict the beneficiaries of Article 6(4) and Article 6(7). While this is allowed under other obligations of the DMA,[1663] this is generally not allowed under Article 6(7), as explained in Section 4.7 above. This limit is consistent with the technical approach to security: subjecting feature access to an OS-level eligibility programme would constitute a departure from how most operating systems have been designed for the past 30 years and would have far reaching implications for fairness and contestability. Indeed, this is unprecedented on Google Android for user-downloaded apps: until today, access to Android SDK APIs for user-downloaded apps – including very sensitive ones, such as APIs enabling access to photos, documents, microphone, camera, and location – has been solely subject to technical controls, including for data minimisation and transparency. So far, policy-based access restrictions have been imposed only as part of app store review requirements. In fact, Alphabet's proposed eligibility conditions go even beyond app store review requirements that ensure the absence of malware – which is Alphabet's justification for the need to impose OS-level conditions that apply also outside of the Google Play Store[1664] – and instead impose detailed requirements about how the third party must develop the app (including in terms of data protection and user interface[1665]), which technologies

---

1660 Technical meeting Alphabet/Commission of 6 March 2026, Doc ID […], slide deck page 10.

1661 See https://f-droid.org/en/2025/09/29/google-developer-registration-decree.html, Doc ID […], accessed on 30 June 2026.

1662 According to F-Droid, there has been only a single known case of malware distributed through the F-Droid app store – see https://news.ycombinator.com/item?id=45741132, Doc ID […], accessed on 30 June 2026. By comparison, there have been several documented cases of malware distributed through the Google Play Store at a large scale. For example, in 2025 Zscaler ThreatLabz identified 239 malicious applications hosted on the Google Play Store, which were collectively downloaded 42 million times – see https://ir.zscaler.com/news-releases/news-release-details/zscaler-threatlabz-reveals-67-jump-android-malware-and-40-iot, Doc ID […], accessed on 30 June 2026.

1663 For example, some of the provisions in Article 6(3) explicitly apply only to a "list of main providers" of some services. In fact, Alphabet operates a programme that allows third parties to apply to be part of choice screens. As another example, Article 7(1) limits the beneficiary of interoperability to providers of number-independent interpersonal communications services, and allows the gatekeeper to impose general terms and conditions of interoperability with its number-independent interpersonal communications services, including the necessary details on the level of security and end-to-end encryption that the third-party service must meet.

1664 Alphabet's response to the PFs, Doc ID […], paragraph 153, sixth bullet.

1665 For example, the Proposed Eligibility Conditions include [requirements for the Agent on data minimisation for personal and sensitive data, and the need to explicitly inform the user, during onboarding, and allow it to manage the Agent's pre-established access to any apps.]

it must use (including supply-chain requirements[1666]), and how it must operate the cloud infrastructure that supports the app.[1667]

(1398) *Fifth*, there is a significant overlap between the eligibility conditions proposed by Alphabet and obligations under Union laws.[1668] The gatekeeper should not impose OS-level eligibility restrictions that go beyond what is already mandated by law and constrain how third parties operate their business.[1669] If the gatekeeper was allowed to be a private enforcer of such restrictions, even if they are mandated by Union law, this would negate the aim of the DMA to counteract the power imbalance between gatekeepers and business users.[1670]

(1399) *Sixth*, eligibility conditions are, by definition, conditions of access to the OS: an app using a restricted feature may not access the OS, unless it meets the eligibility conditions. The appropriate type of service where a gatekeeper can impose conditions of access is their app store, not the OS – which is why the PFs and Section 4.8.2 above refer to this possibility and to Article 6(12) DMA,[1671] as acknowledged but contested by Alphabet.[1672] This approach is also consistent with the fact that the DMA places the responsibility to remove and mitigate illegal content on app stores, not on operating systems, and does so by referencing Regulation (EU) 2022/2065 (Digital Services Act).[1673] Indeed, Google Play is designated under the Digital Services Act, but Google Android is not.[1674] Similarly, under Regulation (EU) 2024/2847 (Cyber Resilience Act), only the software application store provider as a distributor, not the operating system provider (who is a manufacturer), bears responsibilities on the (third-party) applications it distributes. This is also in line with the position expressed in the Joint Guidelines on the Interplay between the DMA and the GDPR, whereby *"gatekeepers should refrain from imposing, either technically or contractually, how Article 6(4) DMA beneficiaries comply with the GDPR as*

---

1666 See [Alphabet's submission on a category of Alphabet's proposed eligibility conditions] in the Proposed Eligibility Conditions.

1667 See [Alphabet's submission on a category of Alphabet's proposed eligibility conditions] in the Proposed Eligibility Conditions.

1668 For example, data minimization and transparency are already mandated by the GDPR (e.g. in Article 5(1)(c) and 5(1)(a), respectively), while supply-chain controls for security will be mandated by the CRA (e.g. in Article 13(5) on third-party component due diligence). Alphabet did not substantiate whether any requirement goes beyond what third parties are already or will be required to do by law, and if so why it is justified.

1669 See also Joint EDPB and Commission guidelines on the interplay between DMA and GDPR, version for public consultation, Doc ID […], paragraph 93: *"Gatekeepers should take account of the division of responsibility and liability between themselves and third parties providing apps and app stores […]. Gatekeepers should pay particular attention to any technical or contractual measures they seek to impose which may be intended to prescribe the way a third party, such as an app developer, complies with the GDPR. In this regard, as a separate and independent controller, the third party remains responsible and liable for its own processing and should therefore be free to choose how it ensures that such processing complies with the GDPR."*

1670 Recital 4 DMA.

1671 Indeed, Article 6(12) is specifically designed to ensure that designated app stores apply only conditions of access that are fair, reasonable, and non-discriminatory (FRAND). But Article 6(12) does not apply to operating systems. If gatekeepers were allowed to impose conditions of access at the level of the OS, they could simply circumvent Article 6(12) by moving any non-FRAND condition from the app store to the OS. It would also undermine the objective of Article 6(4), because it would allow the gatekeeper to replicate existing (FRAND or non-FRAND) app store-level access conditions at the level of the OS – thereby preventing other app stores from competing by offering more attractive access conditions.

1672 Alphabet's response to the PFs, Doc ID […], paragraph 153, sixth bullet.

1673 Recital 62 DMA, referencing Regulation (EU) 2022/2065.

1674 Decision C(2023) 2725 final (designating YouTube as a very large online platform), and Decision C(2023) 2738 final (designating Google Play as a very large online platform).

*independent controllers"* and *"should not in any way prescribe whether, how, and when Article 6(4) DMA beneficiaries are to seek consent for the collection of data, restrict the further processing of data that was collected for a specific purpose, nor limit the types of processing which beneficiaries may carry out as separate and independent controllers"*.[1675] Yet, as explained above, this is precisely the type of eligibility requirements that Alphabet intends to impose on Article 6(4) beneficiaries, i.e. developers of apps distributed outside of the Google Play Store.

*12.4.4. Exceptional circumstances that justify restricting access to a feature for effective interoperability*

(1400) In exceptional circumstances where effective interoperability requires ecosystem buy-in that is difficult to achieve if a feature were to be interoperable for all beneficiaries from the start, there might be reasons to restrict interoperability with a feature, at least in the initial stages, to a limited set of beneficiaries that meet certain conditions. This is because many ecosystem participants may distrust some other ecosystem participants, and may therefore avoid using a feature altogether if that is the only way to prevent integration with those distrusted participants. Specifically, these exceptional circumstances may arise when the following cumulative conditions are met:

(a) **Changes to security model.** The gatekeeper's decision to make the feature available to services outside of the operating system (including the gatekeeper's own services) changes the existing security model of the operating system both fundamentally and significantly, and is likely to do so at scale;

(b) **Nascent technology.** The feature or the services using the feature mostly rely on nascent technology and on security practices that are not yet mature; and

(c) **Ecosystem adoption.** The usefulness of the feature for any specific service can be significantly undermined unless other ecosystem participants adopt it at scale.

(1401) While the cumulative conditions above are presented as relating to a feature, it should be clear that the exceptionality criteria – and therefore the possibility to restrict access to a feature – actually apply to specific implementations of the feature. A feature might have more than one implementation, and it is possible that only some of the implementations meet the cumulative conditions above – see for example the case of App Functions discussed in Section 12.4.5 below. In that case, the eligibility restrictions may be applied only to the implementations that meet those conditions.

(1402) Some of (the implementations of) the features subject to these proceedings meet the exceptionality conditions. For instance, the implementation of the screen automation feature via Computer Control (see Section 10.3) meets the three conditions above:

(a) *Changes to security model.* Until today, Android has been based on the principle of app isolation, whereby an app cannot directly act on another app.[1676] Alphabet's decision to use screen automation in Gemini fundamentally and significantly changes the existing Android security model: a single service

---

1675 Joint EDPB and Commission guidelines on the interplay between DMA and GDPR, version for public consultation, Doc ID […], paragraph 101.

1676 Although there have been some exceptions to this (most prominently the Accessibility APIs), such exceptions were not designed to be adopted at scale across apps, users, and use cases – which is the case for agentic AI.

(Gemini) can now operate on the users' behalf and take full control of any other app, including apps that are not provided by Alphabet.

(b) *Nascent technology.* Screen automation is mostly designed to be used via multimodal generative models, including LLMs. Generative models are a nascent technology with important peculiarities, most importantly a *de facto* non-deterministic behaviour[1677] and vulnerability to prompt injection.[1678] This poses novel challenges for integrity, privacy, and security – which is why threat models and technical safeguards are still being actively discussed in academia and in the industry.[1679]

(c) *Ecosystem adoption.* Alphabet already designed and implemented the screen automation feature (via Computer Control in Gemini) following a so-called *app caller agnostic model* – see Section 10.3.3. This means that app developers may only decide if AI assistants (the "callers") can automate (i.e. take control of) their app, but may not decide *which* AI assistants can do so. On one hand, the app caller agnostic model constitutes a good foundation for equally effective interoperability. On the other hand, given the nascent nature of the technology, it may be appropriate to temporarily restrict the scope of beneficiaries to first build ecosystem trust and avoid that app developers simply opt out of screen automation altogether. Interoperability with the screen automation feature would not be effective if, in practice, no apps can be automated.

(1403) Importantly, the mere presence of integrity, security, or privacy implications is not sufficient to justify restricting access to a feature to a group of eligible beneficiaries. As explained in the previous section, many features that are already interoperable today have important integrity, security, or privacy implications. These implications have always been, and continue to be, addressed by Alphabet via OS-level technical controls (such as privacy prompts, privacy indicators, app sandboxing) and app store-level eligibility restrictions, without resorting to OS-level eligibility restrictions. Therefore, OS-level eligibility restrictions are generally not strictly necessary and proportionate to ensure integrity and effectiveness of interoperability, as provided for by Article 6(7). By contrast, OS-level eligibility restrictions may be justified under the exceptional circumstances above because it may be difficult to achieve sufficient ecosystem buy-in from the start unless the eligibility conditions are the same across distribution channels, given that industry standards are still being agreed on.[1680]

---

1677 That is, the models can produce different outputs each time they are given the exact same input or prompt.

1678 Prompt injection is a type of attack where an AI agent reads malicious input (e.g. from the app that it is instrumenting) that may cause it to alter its behaviour. For example, in the absence of safeguards, an app might contain some invisible text that is injected into the AI agent during screen automation and instructing the agent to purchase an item or share data without user's awareness.

1679 See e.g. OWASP Top 10 for Agentic Applications for 2026, https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026/, accessed on 30 June 2026, Doc ID […].

1680 Alphabet, Meta, and Microsoft are actively working on defining such a standard (ADA AI Agent Security Specification) within the App Defence Alliance (see https://github.com/appdefensealliance/ASA-WG/blob/develop/AI%20Profile/AI%20Agent%20Specification.md, Doc ID […], accessed on 30 June 2026. This collaboration akin to well-established industry forums that agree on web standards – such as the W3C, IETF, WHATWG, and TC39 – which include all major web browser vendors and where standardisation of web technologies – including for integrity, privacy, and security – is defined. Importantly, such standards are not horizontally enforced and leave significant room for differentiation, including in terms of integrity, privacy, and security. This flexibility is widely recognised as critical for competition among web browsers and for the health of the web and would be seriously at risk if the OS

(1404) The next section explains which other features in scope of this Decision meet these exceptional conditions and can therefore be subject to restrictions on eligible beneficiaries.

*12.4.5. Scope of restricted features*

(1405) The Commission considers that the following features in scope of this Decision meet the cumulative conditions for the exceptional circumstances in the previous section, and Alphabet may therefore restrict interoperability with them to eligible third parties:

(a) *Screen automation*, as implemented via Computer Control, for the reasons explained in the previous section.

(b) *Structured on-device integration*, as implemented via App Functions (see Section 10.2), meets the cumulative conditions for the exceptional circumstances for the same reasons that apply to screen automation. In particular, the App Functions framework too follows an app caller agnostic model and App Functions are primarily designed to be used with LLMs, which are a nascent technology in terms of maturity of security practices.[1681] If used at scale, this feature would fundamentally and significantly change the Android security model: while Android intents are commonly used, as of today it is uncommon for a single app to expose intents to many other apps. Indeed, general Android intents do not meet the exceptionality standard and are not restricted – nor has Alphabet requested to restrict them.

(c) *System integration.* This feature is included as a restricted feature only because [Alphabet plans concerning the potential implementation via App Functions of the measures imposed in relation to the system integration feature]. This is without prejudice to the fact that another implementation might be possible and might not meet the cumulative conditions for the exceptional circumstances to restrict the feature.

(d) *Centralised access to apps' data stored on-device*, as implemented via AppSearch (see Section 9.2), meets the cumulative conditions for the exceptional circumstances for similar reasons to those that apply to screen automation and structured on-device integration. In particular, the feature captures precisely the scenario where the operating system provides, via an on-device database or data sharing platform, centralised access to apps' data stored on-device that developers choose to donate to a particular role.[1682] In this sense, this feature too follows an app caller agnostic model: the user can select any app as the default assistant and that app will have access to the feature, i.e. the *READ_ASSISTANT_APP_SEARCH_DATA* permission and the associated data donated by every app. If used at scale, this feature would fundamentally and significantly change the Android security model: today, few apps share

---

provider could dictate eligibility conditions for web browsers that can run on that OS. Such industry forums and standards are however not yet mature in the context of AI agents, which is why OS-level eligibility restrictions are justified.

1681 The Commission considers that App Functions may in principle be useful even if the caller app does not use LLMs or similar technologies, for example if the app implements rule-based automation to directly invoke an App Function by name rather than by interpreting the description with an LLM, just like any Android intent. This is an example of why providers of services that are not AI assistants could benefit from access to the restricted features in this Decision.

1682 Under the current implementation of the feature via AppSearch, access to the relevant *READ_ASSISTANT_APP_SEARCH_DATA* permission is reserved only for one service at a time (i.e. the default assistant role holder) – see Section 9.2.

data centrally via on-device mechanisms with other apps. Moreover, while the feature does not per se rely on nascent technologies, prominent use cases for it would include AI assistants and solutions for context-aware intelligence, both of which generally rely on nascent technologies.

(e) *Context-aware intelligence* meets the cumulative conditions for the exceptional circumstances for similar reasons to those that apply to centralised access to apps' data stored on-device. In particular, the measures for context-aware intelligence provide for an app caller agnostic model – see Section 9.3.9. Moreover, Alphabet's use of context-aware intelligence, for example via Magic Cue, fundamentally and significantly changes the Android security model and is likely to do so at scale, with many apps surfacing content originating from other apps.[1683] Finally, context-aware intelligence is most likely to be used via nascent technologies, such as on-device models.

(1406) Alphabet requested that three additional features should be restricted:[1684]

(a) *Ambient data.* Alphabet considers that ambient data is very sensitive and it should therefore be accessible only by eligible apps. Following Alphabet's input, the Commission adapted the measures on ambient data to be clearer in requiring only equivalent access as Alphabet's AI-powered services. Moreover, Alphabet submitted that Gemini accesses ambient data only through APIs that are already public[1685] – which are currently not restricted and therefore do not need to be restricted. In that case, and provided that applies to all of Alphabet services (directly or indirectly), it might not be necessary that Alphabet opens up deeper or different access (e.g. without privacy indicators). If and when deeper access will become necessary to ensure equivalent access to Alphabet's services, Alphabet may submit a request to add Ambient Data to the list of restricted features pursuant to paragraph 121 of the Annex.

(b) *Always-on hotwords.* Alphabet considers that access to this feature raises integrity concerns and should therefore be restricted – see Section 8.3. The Commission considers that Alphabet's request is not warranted at this stage. As explained in Section 8.3, there are different possible implementations of the always-on hotwords feature, each having different implications for integrity and effectiveness. If the implementation that Alphabet will opt for raises exceptional integrity and effectiveness challenges, Google may submit a request to add it to the restricted features pursuant to paragraph 121 of the Annex.

(c) *Long-press Home/Long-press Navigation Handle contextual invocation.* Alphabet considers that access to contextual invocation should be subject to eligibility conditions – although less onerous than those that apply to the other restricted feature[1686] – because contextual invocation has access to sensitive data (i.e. the screen content at the time of invocation) and LPH/LPNH invocation is prone to mistrigger. The Commission considers that this feature does not meet the cumulative conditions for the exceptional circumstances that

---

1683 See https://www.androidauthority.com/google-pixel-10-magic-cue-third-party-apps-preview-3675330/, Doc ID […].

1684 Alphabet's email of 15 June with subject "Re: DMA.100220 - Measures on restricted features - without prejudice", Doc ID […].

1685 Alphabet's response to the PFs, Doc ID […], paragraph 264.

1686 Alphabet's submission "revised mark-up of the draft measures" of 28 May 2026, Doc ID […], paragraph 6(a).

justify restricting the feature via an eligibility mechanism rather than technical controls, such as user prompts and privacy indicators:

(1) *First,* the feature does not affect the existing Android security model, either significantly or fundamentally. The exact same contextual invocation (as implemented via the Assist API) is already available to third parties (including downloaded apps) that are selected by the user as default assistant, and already provides access to the exact same data. The feature can be invoked from the exact same LPH/LPNH access point, if the OEM reserves that access point to the default assistant role.[1687] Similarly, when a user takes a simple screenshot, the screen content is automatically accessible by all apps that have access to photos – which of course do not need to go through an eligibility certification for this, as user permission is sufficient on all major operating systems, including Google Android. Internal documents show that Alphabet's engineers recognise [confidential information from Alphabet internal documents comparing the level of risk concerning contextual invocation and that of an already accessible API] – see recital (483)(a) above.

(2) *Second,* the use of the feature does not rely on nascent technology or on security practices that are not yet mature. The main use case of the feature is to simply share the screen content as a static image to be searched online – something that Google Search and other search engines / AI assistants have supported for 15 years.

(3) *Third,* there is no reason to believe that the usefulness of the feature for any specific service can be significantly undermined by low adoption, given that most apps already allow the user to take a screenshot while in the app and to share the content of the screen via the Assist API.

*12.4.6. Minimum safeguards that should apply when restricting feature access to eligible beneficiaries*

(1407) Restricting feature access to a limited set of beneficiaries that meet some eligibility conditions creates significant risks that some business users would be left out even though their services are legitimate, trustworthy, compliant with applicable law, and would meet end user demand. While restrictions can be justified under the exceptional circumstances described above, it is important to mitigate the risks that could undermine the objectives of Article 6(7) and of the DMA more broadly.

(1408) These risks may include the risk that legitimate business users are excluded simply because they do not meet the eligibility conditions set *ex ante*; the risk that the eligibility conditions go beyond what is necessary to address the exceptional conditions, creating significant costs that might discourage legitimate third parties from applying; the risk that the gatekeeper does not have the incentive to ensure that

1687 Moreover, while Alphabet claims that the chance of mistrigger for LPH/LPNH are high (see Technical meeting Alphabet/Commission of 21 May 2026, Doc ID […], slide deck 18), Alphabet has not provided any data comparing the chances of mistrigger between LPH/LPNH and other access points that are already commonly used with the Assist API, such as long-press on power button. Alphabet has also not provided other benchmark data, for example on how often users inadvertently trigger the use of features that are already interoperable. In any event, restricting the beneficiaries that can access a feature simply because of the possibility that the user can inadvertently trigger it would be neither necessary nor proportionate, as the feature does not meet exceptional conditions and mistrigger is not even an integrity issue. Should Alphabet consider the risk of mistrigger as unacceptable, Alphabet can easily apply technical controls to minimize it, such as implementing an additional step requiring explicit confirmation from the user before the content is shared with the app.

the process to verify eligibility is efficient, fast, and reliable; the risk that the gatekeeper intentionally abuses the process to delay or obstruct access for competitors; the risk that the eligibility checks require the third party to share commercially sensitive information with the gatekeeper. These concerns of forced intermediation are precisely those that underpin the objectives of Article 6(4) DMA, namely to allow app developers to avoid distributing apps under the conditions and review mechanisms of the gatekeeper's app store and to enable competition between app stores with different app review policies and processes.[1688]

(1409) To mitigate the risks of undermining the objectives of Article 6(7) and of the DMA, the Commission considers it necessary and proportionate that the eligibility conditions and the process to certify eligibility are subject to the following safeguards and requirements, which are reflected in the measures in Section 5.4 of the Annex.

(1410) *First*, the eligibility conditions should be strictly necessary to ensure that the third-party service meets appropriate standards that adequately mitigate clearly identified risks to integrity and effectiveness. For example, unlike Alphabet's proposal,[1689] the eligibility conditions shall not prohibit the type of app or the use case for which the feature is used, as the type of app and the use case do not – if legal – inherently determine the level of risk. Alphabet argued that the restricted features in scope of this Decision should be available only to "virtual assistants" because only such apps are capable of using them.[1690] The Commission notes that the concept of virtual assistants in this fast-moving market is continuously evolving and therefore any restriction needs to be future-proof to ensure the effectiveness of the measures over time. The Commission also considers that it is not possible to know in advance all the possible use cases and purposes for which a business user may legitimately want to use a feature, and Alphabet did not substantiate these claims. In fact, Alphabet's internal documents show that Alphabet considered allowing "automation apps" [1691] and [a non-assistant Google app] [1692] – neither of which are virtual assistant apps – to execute App Functions.

(1411) *Second*, to avoid that the risks set out above materialise, eligibility should be assessed within a structured certification programme in which a third party can apply to have their service certified as meeting the eligibility conditions ("Qualified service"). The certification programme should have clear processes, timelines, and mechanisms for appeal. Eligibility conditions should be transparent, objective, precise, and non-discriminatory to ensure predictability and leave little room for discretion to the gatekeeper, whose incentives to allow competitors on the platform might be misaligned with the interest of the potential beneficiaries as well as the

---

1688 [Third party] explained that subjecting feature access to approval from Alphabet is *"sufficient to disqualify an entire category of legitimate beneficiaries"*. [Third party] further explains: *"Google's verification and certification requirements can operate as gatekeeping mechanisms rather than mere quality controls. In practice, access to core platform features becomes tied to participation in Google's commercial ecosystem."* See [third party's] submission to the public consultation, Doc ID […], page 10. [Third party] cites also the Keep Android Open campaign, which opposes a similar mechanism for mandatory OS-level developer registration that Alphabet is planning to impose on all developers of Google Android apps – see https://keepandroidopen.org/open-letter/, Doc ID […], accessed on 30 June 2026.

1689 Alphabet's eligibility conditions explicitly exclude apps that use the features for rules-based automation, which is a legitimate use case. See Alphabet's submission […] of 19 May 2026, Doc ID […], page 3.

1690 See Technical meeting Alphabet/Commission of 12 June 2026, Doc ID […], slide deck page 6.

1691 [Alphabet's internal document] of December 2024, Doc ID […], page 51.

1692 [Alphabet internal document] of December 2024, Doc ID […], page 8.

public interest. This programme should be open to every type of service that uses or may use the restricted features (provided the restrictions are justified for that type of service in the first place), not only to a specific type of service (e.g. AI/virtual assistants). To ensure that these requirements are respected in practice, Alphabet should closely engage with the Commission when defining the eligibility conditions, who may in turn consult third parties.

(1412) Moreover, as AI assistants are the type of services that will most prominently benefit from interoperability with the restricted features subject to these proceedings, the Commission considers that it is important to limit the type of eligibility conditions that Alphabet may impose to certify an AI assistant ("Qualified AI assistant"). These are conditions relating to: functional eligibility; user intent; agent and model functional risks; agent privacy, safety, and transparency; agent app security; developer reputability. These conditions reflect those proposed by Alphabet, but are less detailed to allow Alphabet to further specify them in consultation with the Commission and third parties during the implementation phase, ensuring that they are strictly necessary and proportionate. In any case, Alphabet may propose to the Commission to include additional types of conditions.

(1413) *Third*, while Alphabet may define the eligibility conditions, it should not have the exclusive power to adjudicate and enforce them. Therefore, in parallel, Alphabet should allow trusted third parties ("Trusted Certification Authorities") to determine and certify that a business user's service meets the eligibility conditions. However, in case there is evidence of risk of severe and immediate harm to the user, Alphabet should not be prevented from unilaterally suspending access for a specific beneficiary. Alphabet may define the conditions that a third party should meet to qualify as a Trusted Certification Authority, as well as the process to obtain such a qualification ("Trusted Certification Authorities Programme"). Alphabet may retain the sole responsibility to certify third parties as Trusted Certification Authorities. In any event, Alphabet should certify Trusted Certification Authorities free of charge. Such programme should be in line with similar certification schemes already existing in the market. For example:

(a) Access to some Windows APIs require the app to be signed by Microsoft or a third-party Certification Authority that participates in Microsoft's Trusted Root Program.[1693]

(b) The App Defence Alliance (ADA), an initiative by Google, Meta, and Microsoft, operates a certification programme for mobile applications. ADA has accredited a Certification Body that authorises security test labs, who in turn evaluate e.g. mobile apps. The Certification Body reviews the evaluation and then issues the certification.[1694]

(c) All major browsers (including Google Chrome) support Extended Validation (EV) Certificates issued by selected TLS/SSL certification authorities. These certificates can be consulted by a user when visiting a website, providing a trustworthy confirmation of the identity of the entity that controls the website.

---

[1693] See Microsoft DMA Compliance Report 2026, Doc ID […], Windows Section 2 Annex, paragraph 366 (https://www.microsoft.com/content/dam/microsoft/mscle/documents/presentations/Microsoft_DMA_Compliance_Report%E2%80%93Windows_Section_2_Annex.pdf) and Program Requirements - Microsoft Trusted Root Program. See https://aka.ms/RootCert, Doc ID […], accessed on 30 June 2026.

[1694] App Defense Alliance, Certification Scheme Overview, version 1.2, 26 March 2026, https://drive.google.com/file/d/1pOr_HcAIb9PqeYaYO05PEbYqagKxM2dR/view, Doc ID […], accessed on 30 June 2026.

To obtain an EV certificate, a website owner must prove to the certification authority that it is a legitimately registered business entity.[1695]

(d) Regulation (EU) 2019/881 (Cybersecurity Act) lays down a framework for the establishment of European cybersecurity certification schemes.[1696] Each scheme, prepared by ENISA, sets out cybersecurity requirements for a type or categories of ICT products, ICT services and ICT processes.[1697] National accreditation bodies accredit conformity assessment bodies (e.g. security companies) who in turn independently certify ICT products, ICT services and ICT processes under such a scheme.[1698] Regulation (EU) 2024/2847 (Cyber Resilience Act) and Regulation (EU) 2024/1689 (Artificial Intelligence Act) state that products with digital elements and high-risk AI systems, respectively, certified under a relevant scheme, are presumed to comply with the cybersecurity requirements set out in those Regulations.[1699]

(1414) *Fourth*, in line with existing schemes for privacy and security,[1700] user agency to grant access to features to any app on their device should be preserved, subject to appropriate transparency mechanisms that ensure informed consent. Alphabet should therefore allow users to install an app and grant it permission to access Restricted features, even if the app and developer have not been certified. There are many reasons why a user may want to install and use a service that is not (or not yet/anymore) verified, including for safety and data integrity reasons.[1701] These concerns are exacerbated by the fact that a "kill switch" can be triggered by mistake. In fact, Google Play Protect – Alphabet's mechanism to flag and remove Android apps detected as harmful – has a documented history of falsely flagging legitimate apps, including two popular Samsung apps.[1702] Alphabet indeed allows the user to disable Google Play Protect.

(1415) Alphabet argued that allowing third parties to certify eligible beneficiaries is overly restrictive.[1703] Alphabet argued that, as the *"operator of the Android OS"*, under all circumstances it should have the final word and *"sole responsibility"* to accept

---

1695 See https://cabforum.org/working-groups/server/extended-validation/documents/CA-Browser-Forum-EV-Guidelines-2.0.3.pdf, Doc ID […], version of 6 July 2026.

1696 Article 1(1)(b) of Regulation (EU) 2019/881.

1697 Articles 46(2), 49, and 54(1)(a) of Regulation (EU) 2019/881.

1698 Article 56 and 60 of Regulation (EU) 2019/881.

1699 Article 27(8) of Regulation (EU) 2024/2847 and Article 42(2) of Regulation (EU) 2024/1689.

1700 For example: Google Play Protect can be disabled by the user in the Google Play Store settings; Alphabet explained that it will roll out a new mechanism for third-party app stores to be marked as more trusted by the operating system ("Registered App Stores"), but end users will continue to be able to install apps from any app store and from the web; Alphabet allows Chrome end users – including on Android – to visit any website even though the certificate is absent or invalid, after informing the user of the risks.

1701 For example, the app might be meant for limited distribution only, e.g. within a company, an association, or a group of early testers. In these cases, meeting the eligibility requirements would be disproportionate and too onerous. As another example, the user might use the app in critical settings and must ensure that the app cannot be suspended or made dysfunctional without the user's confirmation. The app might also store important data that the user might be unable to recover if it is unilaterally suspended. None of the circumstances apply only to developers: in principle, they can apply to end users, meaning that Alphabet's request to restrict this possibility to developers would not adequately address the issue (see Alphabet's submission "Draft Measures: Priority issues", Doc ID […], page 4 and Alphabet's submission "Google's mark up on the restricted features measures", Doc ID […], paragraph 16). It may nevertheless be appropriate to gate the relevant user options behind reasonably high friction to confirm that the user is aware of the implications.

1702 See https://9to5google.com/2023/10/23/samsung-messages-wallet-harmful-app-google/, accessed on 30 June 2026, Doc ID […].

1703 Alphabet's response to the PFs, Doc ID […], footnote 27.

beneficiaries of the features subject to these proceedings, as no other party would be well placed to handle the certification process in its entirety.[1704] The Commission notes that these claims are vague and unsubstantiated. As explained above, such a mechanism to restrict access to features is unprecedented on Android – including for very sensitive features such as access to the microphone and access to photos. Moreover, Alphabet's claims fail to account for the scope of integrity in Article 6(7) and for the overall disintermediation objectives of the DMA.[1705] Alphabet's position relies on the assumption that Alphabet should regulate every activity carried out via Google Android, while at the same time exonerating itself from legal responsibility (see Section 4.8.1). Finally, Alphabet's claim that no other party is well placed to verify that a third-party service is eligible according to the criteria set by Alphabet contradicts Alphabet's own argument that OEMs (i.e. third parties) are capable – or even best placed – to determine which apps can safely access the features subject to these proceedings – see Section 6.3.1.

(1416) Therefore, to attain the objectives of the interoperability obligation set out in Article 6(7) DMA, Alphabet shall implement the measures in the Annex in the Section "Eligibility of beneficiaries, applications and use cases" (Section 5.4).

### 12.5. Equal effectiveness

(1417) As described in Section 4.5 above, Article 6(7) DMA requires that an interoperability solution offered to third parties must be equally effective and provided under equal conditions to the interoperability solution available to Alphabet's own services and hardware.

(1418) As also explained in Section 4.3 above, Article 6(7) DMA refers to "*the same features*", and the respective recitals highlight the need for interoperability which is "*equally effective*" to that available to the gatekeeper and offered "*under equal conditions*".

(1419) Therefore, Alphabet should ensure that any interoperability solution implemented for the features listed in this Decision is equally effective and provided under equal conditions to the interoperability solutions available to its own services and hardware.

(1420) Alphabet should apply such equal effectiveness and equal conditions across all dimensions, including, but not limited to, the end user journey, ease of use for end users, device and software setup, data transmission speed, default settings, and energy consumption. A difference in any such dimension, e.g. higher energy consumption by lack of access to low-power processors, could negatively affect the attractiveness of third-party services.

(1421) Alphabet claimed that these requirements are "rigid" and "formalistic".[1706] The Commission considers this a mischaracterisation. Indeed, these requirements recall the underlying principles of the interoperability obligation under Article 6(7). They do not prescribe a particular interoperability solution, thereby leaving room for Alphabet to implement different solutions for its own services than for third-party services, subject to those solutions ensuring effective interoperability and being non-discriminatory. Indeed, the requirements ensure that user-facing and non-user-facing

---

1704 See Alphabet's submission "Draft Measures: Priority issues", Doc ID […], pages 3-4. See also Alphabet's submission "Google's mark up on the restricted features measures", Doc ID […], paragraphs 8-10.

1705 See recitals 4 and 50 DMA.

1706 Alphabet's response to the PFs, Doc ID […], paragraph 357.

parameters governing access to the feature – which could affect the effectiveness of the interoperability solutions – are applied on equal terms to both Alphabet's interoperability solution and any interoperability solutions implemented by Alphabet for third parties. If the interoperability solutions for the features listed in this Decision are not as effective or are provided on less favourable conditions than for Alphabet's services or hardware—both for user-facing and non-user-facing aspects—third parties will be disadvantaged. In particular, there is a risk that the gatekeeper's services appear more attractive to users, not because of their intrinsic merits but rather due to arbitrary limitations on the effectiveness of third-party services (e.g. lower resource allocation leading to slower performance or high-friction user journeys).

(1422) For instance, as regards the end user journey and the ease of use for end users, the effectiveness of interoperability solutions could be undermined by introducing unnecessary friction when an end user uses third-party services or hardware (Section 4.5.2). Friction has an impact on the ease, convenience, and speed of using the service or hardware from the end user perspective. Therefore, to ensure that the access for third parties to the features listed in this Decision is equally effective to the solution available to Alphabet's services and hardware in terms of user experience, it is appropriate and necessary that Alphabet does not decrease the ease, convenience and speed of using third-party services from the end user perspective, as compared to Alphabet's services.

(1423) Preventing unnecessary friction is particularly important at those moments where consent or permission is asked from the end user, e.g. to grant a given third party access to a given feature. It is important that the user has agency to authorise such feature access (Section 4.8.3), and the gatekeeper may give the user this ability (Section 12.2). However, given that this authorisation step is under the control of the gatekeeper, as e.g. relevant prompts are its operating system, there is a risk that the gatekeeper engages in behaviour that would undermine the effectiveness of Article 6(7), concretely by discouraging the user from granting permission for a third party to access a feature potentially in a subtle manner or by engaging in other design practices or behavioural techniques described in Article 13 DMA. Otherwise, this would result in the gatekeeper's service being able to offer the user a better and richer service, by virtue of broader feature access, thereby limiting the attractiveness of third parties' services for end users, contrary to the objectives of contestability and fairness in the DMA.

(1424) Section 4.5.2 above provides why preventing unnecessary friction is important to preserve effective interoperability. The Commission sets out below further details as to how friction may undermine effective interoperability.[1707]

(1425) *First,* Alphabet should refrain from adding friction by offering choices to, or requesting permission from, the end user in a non-neutral or leading manner. Such leading choices or misrepresented risks may insinuate that the third-party service should not be entrusted with feature access, making it more likely that the end user will refuse such access.

(1426) *Second,* Alphabet should refrain from adding friction by preventing the third party from explaining to end users in their own language the relevance of any system prompts shown, immediately before the prompt is shown or within the prompt. This

[1707] Alphabet at numerous instances tried to strike down these measures. See for instance, Alphabet's first draft measures redline of 20 May 2026, Doc ID […]; Alphabet's second draft measures redline of 28 May 2026, Doc ID […].

ensures that third parties can clearly inform users about the purpose of asking for permission to access specific features (e.g. explain why they request access to screen content or location). It also ensures that third parties have full autonomy over how they communicate with their end users and are not put at disadvantage by how Alphabet may choose to frame the respective information on their behalf.

(1427) *Third,* Alphabet should refrain from adding friction by showing unnecessary recurring prompts or notifications that the end user cannot easily and permanently disable in the same prompt or notification. Such recurring and unnecessary prompts or notifications may cause user fatigue and may lead to the end user deactivating access to features for that respective third party, just to avoid being prompted again, when there is no less restrictive option to dismiss the prompts instead. Recurring and unnecessary prompts or notifications may also risk being shown out of context (e.g. while the end user is in another app entirely) and convey a false sense of danger (e.g. insinuating that the app accesses a feature more often than needed). For instance, a weather app may regularly access the user's location to ensure the user has up-to-date information, and the user may want to dismiss recurring prompts that the app has location access, if they trust that service.

(1428) *Fourth,* Alphabet should refrain from adding friction by preventing the third party from triggering a permission prompt again in the future, unless the end user has so decided. This ensures that third parties have more than one opportunity to be granted the user's trust to access a feature, even if the end user may have refused to grant authorisation earlier on; while the user should retain agency to deny it permanently. This is particularly relevant if the service may request several bundled permissions when the end user first opens the app, at which point the user may deny the permission, but the service later wants to request the permission again in a more dedicated and appropriate context, where they can better explain to the user why the service needs the permission (e.g. asking for microphone access when the user activates a speech function). This is also relevant if a user may have inadvertently dismissed the permission prompt.

(1429) *Fifth,* Alphabet should refrain from adding friction by preventing third parties from obtaining user consent for any required permissions (i) from inside the application itself or with a simple system prompt or (ii) by indicating the user to follow a link directly pointing to and highlighting the relevant item in the system settings (so-called "deep-linking"). This ensures that effective interoperability is not undermined by artificially increasing the steps for granting permissions to third parties, including by requiring end users to switch apps and contexts, which may risk end user drop-off.

(1430) *Sixth,* Alphabet should refrain from adding friction by requiring end users to process multiple successive permission prompts that could be presented in a single prompt. This ensures that effective interoperability is not undermined by artificially created user fatigue, which may lead the end user to drop in the middle of the process and not grant the permission required by the third party.

(1431) The end-user journey for third parties' and Alphabet's services should not differ when it comes to obtaining user consent and informing users about feature access. Alphabet merely states that "*this is not a results-based obligation that ensures equal effectiveness*", without any substantiation.[1708] As explained above, consent, permission, and information moments are crucial to the effectiveness of Article 6(7) DMA, as they determine feature access for a third party. There is no reason why the

[1708] Alphabet's response to the PFs, Doc ID […], paragraph 358.

end user should not be granted the same agency on feature access when it comes to the gatekeeper's services. A differentiated user consent experience cannot be justified by the mere fact that third parties are not the gatekeeper, and therefore they cannot be trusted, as this is a subjective assessment exclusively within the gatekeeper's control.[1709] This ensures that third parties have a fair chance to compete with Alphabet's services, and that Alphabet's services are not treated more favourably, by mere virtue of being preinstalled or by Alphabet having control over system prompts. For example, the Google app receives many pre-granted permissions, granting access to sensitive resources, such as the calendar, microphone, or camera (see Section 5.3). Third-party services would have to request each permission independently, adding friction. Similarly, there should not be a risk that end users falsely associate a lack of information screens for Alphabet's services with a safer experience offered by Alphabet's services.

(1432) Interoperability solutions should be granted in a manner that is technically sound, stable, and workable in practice for third parties without unnecessary hurdles. Shortcomings in the interoperability solutions, such as bugs or undefined behaviour, would prevent these solutions from being deployed in practice and working as intended for third parties, and being properly integrated with the systems they are intended for. This lack of effectiveness would negatively affect the user experience and discourage the user from using the third-party service.

(1433) Access to a feature may be subject to quantitative limitations – such as quotas and rate limits – only if these limitations are based on transparent, objective, precise, and non-discriminatory conditions that also apply to Alphabet's services and hardware, including for use cases that Alphabet does not offer. Alphabet "*agrees that any quota limits should be applied in a proportionate and non-discriminatory manner*" but claims that there might be *"legitimate and proportionate reasons for differences"* in quotas and rate limits, giving as example that core system functionality should receive higher priority.[1710] The Commission notes in this respect that conditions should be non-discriminatory *between* Alphabet's services and third-party services, not between the OS and services. The OS may receive higher priority for its core system functionality. This is relevant in particular for on-device model implementation, where the measures explicitly recognise this differentiated priority (see Section 11.3.8).

(1434) Alphabet should design interoperability solutions so that all apps can use features in a concurrent and a non-discriminatory way and without a requirement for an app to hold a default role, including the default assistant role. Alphabet confirmed that certain features are currently reserved to the default assistant role holder only.[1711] However, those restrictions would prevent effective multi-homing and favour a "winner-take-all" scenario, to the detriment of contestability.[1712]

(1435) Alphabet claimed that the requirement to enable the concurrent use of the features by all apps in a non-discriminatory way and without a requirement for an app to hold a

---

1709 See Section 4.8.4 read together with footnote 232.

1710 Alphabet's response to the PFs, Doc ID […], paragraph 360.

1711 Alphabet's reply to RFI PO 11 of 9 March 2026, Doc ID […], question 23, paragraph 60 and Table 1. See also Technical meeting Alphabet/Commission of 5 March 2026, Doc ID […], slide deck page 19. This is also due to the granting of certain permission to the assistant role. See Alphabet internal document, Presentation […] of August 2025, Doc ID […], page 227.

1712 Recitals 13, 27, and 59 DMA. See [Academic]'s submission to the Public Consultation, Doc ID […], Section I: "*The search cases found, on the basis of evidence Alphabet itself produced at trial, that this kind of default-driven exclusion is sufficient to maintain a monopoly*."

default role is unjustified and technically unreasonable. Alphabet asserted that exclusive roles bring important security and system health benefits, that many functions of the device are designed to be exclusive, that users expect and require the concept of default functionality on their mobile devices, and that unlimited concurrency threatens integrity.[1713]

(1436) *First,* the Commission notes that nothing in this Decision or the Annex undermines the concept of defaults on Android devices. Under the DMA, the concept of default refers to a pre-configured option that is automatically activated when a user does not make an active choice as to what service or app should respond to its action. Specific DMA obligations, such as Article 6(3), recognise the existence of defaults, from a choice architecture perspective and with the aim to ensure that user inertia is addressed by giving users a more prominent and easier choice of alternative services. Nothing in this Decision or the Annex precludes users from sticking with one single specific service for all the tasks that they would otherwise perform by using the app holding the default (assistant) role. In addition, the measures specifically allow Alphabet to allow OEMs to reserve one access point, other than LPH/LPNH, to the default assistant role.

(1437) The change brought by the measures in this decision would strictly concern what interoperability capabilities a default (assistant) role would grant. As explained in Section 5.2.4 above, Alphabet uses defaults to limit access to certain features of Google Android for third-party services. This is at odds with Alphabet's claimed respect for user autonomy.[1714] The measures simply aim to ensure that users can easily use their service of choice for different tasks on Android devices, without needing to go through the process of switching defaults each time. The fact that default switching is "easy",[1715] as Alphabet claims,[1716] does not remove the need to untie feature access from the requirement to hold a default role. In particular, if the default role grants exclusive access to certain features – as Alphabet wishes – changing the default could unduly break other functionality in the former default service which cannot enjoy deep access to Google Android features once it does not hold the default role anymore.

(1438) *Second,* the Commission considers that Alphabet did not substantiate why defaults preserve security. Alphabet claimed that defaults require the user to make an explicit decision to replace the default holder and that they limit the attack and failure surfaces to which a user is exposed.[1717] The Commission notes that untying the access to the features from the default role does not mean accessing the features without explicit user authorisation. The Commission further notes that changing the default assistant currently only triggers one consent moment, with multiple permissions being bundled and granted at once, and with the end user only seeing a short description of access that the new default assistant obtains, without a concrete list of features involved. If these permissions were not bundled under the default assistant role, but could be granted to multiple services, as the measures foresee, end users would have more granular control over which specific permissions to enable for each service, and will be able to make an explicit decision concerning such access. The Commission also notes that, as explained in Section 4.8.1 above, opening interoperability to third parties would only increase the attack surface insofar as it

1713 Alphabet's response to the PFs, Doc ID […], paragraph 359.
1714 Alphabet's response to the PFs, Doc ID […], paragraph 359, third bullet.
1715 In any event, Article 6(3) DMA requires that default switching is easy.
1716 Alphabet's response to the PFs, Doc ID […], paragraph 359, fifth bullet.
1717 Alphabet's response to the PFs, Doc ID […], paragraph 359, first bullet.

could lead to unauthorised access. This will not be the case following the implementation of the measures, as Alphabet can require user authorisation to grant access to the respective feature.

(1439) *Third,* Alphabet's assertion that exclusive roles are important for system health is not supported by evidence either. Regarding the purported threats to system health or integrity due to concurrency, the measures allow Alphabet to take technical capabilities into account when enabling concurrency, including system health (e.g. see Sections 1.2 and 2.2 in the Annex). There is no requirement to support "unlimited" concurrency, as Alphabet claims.[1718] However, a limitation of concurrency to one (or any fixed value), without considering the actual system capabilities, would be arbitrary and unduly restrictive.

(1440) *Fourth,* the Commission does not contest that certain functions of the device are designed to be accessed exclusively by one service at a time. Nothing in the Decision or the Annex prohibits exclusive access to the camera when in use,[1719] if there are duly justified technical reasons. Alphabet already dedicates efforts to enabling further concurrent feature use, for example for concurrent microphone access,[1720] which should not be unduly impeded by arbitrary limitations on concurrency. The Commission further notes that, contrary to Alphabet's claim, processors can be accessed concurrently (via multiple cores and multithreading), and, in any case, such example is not relevant to the features in the Decision or the measures in the Annex.

(1441) One important use of AI services lies in their ability to support use cases within other applications, by centrally exposing assets. For example, an email app may use an on-device model from a third-party AI service to provide to end users email drafting support or grammar check. Alphabet already offers its Gemini Nano model centrally via the OS, for all apps to use (Section 11.2). Such use cases increase the usefulness of the respective AI service across the apps' ecosystem and are therefore important for allowing third parties to compete on a level playing field with Alphabet. It is therefore important that third-party apps can centrally expose their functionalities to other apps, and that these apps are able to access these functionalities in a way that is equally effective to what is available to Alphabet's services and hardware.[1721]

(1442) Central exposure needs to occur in an efficient manner, with a seamless experience for end users. Third-party apps should therefore be able to dynamically provide and retrieve assets at runtime, including from other apps, in a way that is equally effective to what is available to Alphabet's services and hardware.[1722] If the app providing the assets is not installed when another app requests those assets, it should be possible for the user to install the first app via a low-friction in-app mechanism, in a way that is equally effective to what is available to Alphabet's services and hardware.[1723] Otherwise, the user would be required to complete a more complicated user journey (e.g. navigating to a software application store) to retrieve the asset themselves, with a risk that the user abandons such a journey. This makes the use of third-party assets such as on-device models less attractive to developers of other

---

1718 Alphabet's response to the PFs, Doc ID […], paragraph 359, fourth bullet.

1719 Alphabet's response to the PFs, Doc ID […], paragraph 359, first bullet.

1720 https://developer.android.com/media/platform/sharing-audio-input, Doc ID […], accessed on 15 April 2026.

1721 For example, Google Play Services APIs are exposed centrally to other apps.

1722 For example, Gemini Nano is downloaded via AICore when the user takes an action that triggers the download, e.g. by using a feature in Alphabet's apps that requires Gemini Nano.

1723 For example, in Google Play, Alphabet implements the Play Auto Install and Google Play Inline Installs mechanisms.

apps, since there is a risk that the assets will not be available, breaking the app experience. In turn, that makes it less attractive for third parties to offer such assets, whereas Alphabet can leverage its position as OS provider, centrally distribute its assets, and therefore encourage adoption of e.g. its own on-device models.

(1443) Therefore, to attain the objectives of the interoperability obligation set out in Article 6(7) DMA, Alphabet shall implement the measures in the Annex in the Section "Equal effectiveness" (Section 5.5).

### 12.6. Free of charge

(1444) Pursuant to Article 6(7) DMA, interoperability with and/or access for the purposes of interoperability to the features within the scope of Article 6(7) DMA must be provided free of charge. Alphabet, or third-party OEMs integrating Google Android, shall not charge any fees, directly or indirectly, to beneficiaries for effective interoperability with the features within the scope of Article 6(7) DMA.

(1445) As described in Section 4.5.1 above, Article 6(7) DMA is a results-based obligation with the goal of enabling effective interoperability. All implementation steps to enable that the desired interoperability result can be achieved in practice fall within the scope of Article 6(7) DMA and must therefore be provided free of charge.

(1446) Therefore, to attain the objectives of the interoperability obligation set out in Article 6(7) DMA, Alphabet shall implement the measures in the Annex in the Section "Free of charge" (Section 5.6).

### 12.7. Documentation and APIs

(1447) As described in Section 4.5 above, Article 6(7) DMA requires an interoperability solution to be effective. This means that the interoperability solution must work in practice and be "fit for purpose" and accessible to third parties.

(1448) Apps integrate with the operating system through well-defined APIs.[1724] Alphabet generally enables interoperability with Google Android features through APIs in the Android SDK[1725] and in Jetpack libraries. These are accompanied by proper documentation so that they can be used by developers effectively, fulfilling the same necessary purpose as user manuals for consumer products. Such documentation is critical for third parties to build apps in practice, as it contains all the information required to make use of the APIs in third-party apps, with details about the classes, methods, return types, arguments and more, often supported by tutorials and examples. Making available and maintaining complete, accurate, and well-documented APIs is necessary to enable developers to understand and effectively use the interoperability solutions and is an unavoidable corollary of allowing effective interoperability pursuant to Article 6(7) DMA.

(1449) Effective interoperability requires that third-party developers can rely on the interoperability solution over time. Third parties develop their services and hardware based on the interoperability solution and make significant investments for this purpose. The documentation and APIs for the features listed in Sections 8 to 11 of this Decision should therefore be properly maintained over time, and in line with Alphabet's internal processes and policies, in particular such that those features are not treated unfavourably, compared to other features or interoperability solutions, because they are subject to this Decision.

---

1724 See footnote 106 above.

1725 See footnote 107 above.

(1450) Therefore, to attain the objectives of the interoperability obligation set out in Article 6(7) DMA, Alphabet shall implement the measures in the Annex in the Section "Documentation and APIs" (Section 5.7).

### 12.8. Assistance and testing

(1451) Assistance and testing are meant to support effective and timely interoperability for third parties.

(1452) For interoperability solutions to be effective and workable in practice, third parties must be able to receive adequate technical assistance for the purpose of implementing effective interoperability in relation to the features listed in this Decision. While Alphabet is not required to provide unlimited technical support to third parties, it shall provide the support and testing that is required so that third-party services and hardware can interoperate with the features listed in this Decision as effectively as Alphabet's services and hardware can. This follows from two related principles of Article 6(7) DMA. First, the obligation is results-oriented, and third parties need to be able to effectively interoperate with the respective feature. Second, the interoperability solution needs to be equally effective to that available to Alphabet.

(1453) Furthermore, Alphabet should enable assistance and testing in a way that prevents any delays beyond the implementation timelines set out in the Annex. This is particularly important to avoid perpetuating the existing lack of contestability, considering that Alphabet enjoys a head start in adapting its services to the respective interoperability solution, due to its existing seamless access to the Google Android features captured under this Decision. Any engagement with, and any communication to, third parties concerning the implementation of the measures in the Annex shall therefore be carried out efficiently, promptly and without undue delay.

(1454) In the same vein, to achieve effective interoperability, all interoperability solutions implemented to address the measures specified in the Annex must be sufficiently tested, in line with Alphabet's usual testing cycles and practices, including beta testing. Alphabet shall make it possible for third parties to integrate and test the interoperability solution, including in how the solution will be used by their service or hardware, before the solution's general release in a publicly available Android update. Alphabet already makes several beta versions of new Android releases available to all developers,[1726] and such versions should include the interoperability solutions for the features covered by this Decision.

(1455) Alphabet's claim that developers can and do develop new interoperable products using already-released products fails to consider the need for interoperability to be effective.[1727] Pre-release testing is crucial for third parties to develop a well-functioning service or product where such service or product is reliant on Alphabet's interoperability solution. By being able to test pre-release, third parties would gain valuable time to adapt their services before these interoperability solutions become widely available to users. In addition, third parties will be able to familiarise themselves with that interoperability solution and identify its scope, possibilities, and importantly its shortcomings and issues, which they may also report to Alphabet in order to improve the implementation of the interoperability solution in Android.

---

1726 See https://www.google.com/android/beta, Doc ID […], accessed on 10 July 2026.

1727 See Alphabet's response to the PFs, Doc ID […], paragraph 361.

(1456) Alphabet also claimed that pre-release testing would slow down the development of Android and [claim of impact on Alphabet's competitive position].[1728] Alphabet did not substantiate this claim further. In any case, as explained before, Alphabet already allows third parties to test Google Android features before these are released, via beta testing, which is well-established and embedded into Android's development model. Moreover, this claim appears a generic remark on testing, whereas the testing requirement in the measures only applies to the features subject to this Decision (which already exist), narrowly limiting its scope and making a significant impact on the development of Android at large unlikely.

(1457) Pre-release testing should be available to all third parties. Alphabet asserted that it is unclear how it could select third parties for pre-release testing without discrimination.[1729] Alphabet does not substantiate why it would have to discriminate and select only some third parties for testing, especially since the measures require it to ensure that third parties can test the solutions, and impose no restrictions on which parties may do so (in line with Article 6(7) DMA as a whole, see Section 12.4.1). Android's existing beta programme also does not restrict eligible developers.[1730]

(1458) Alphabet shall not impose any undue restrictions on the ability for third parties to develop and test services that use the interoperability solutions implemented to address the measures specified in the Annex to this Decision. In particular, this includes not preventing the ability for a third party to develop and test services that access Restricted features, without that service having been certified yet as a Qualified service. Otherwise, the third party could not evaluate or test their ability to integrate with these features before going through the certification programme. Moreover, this certification depends in part on feature-specific conditions, which would be impossible to guarantee without the possibility to test the features upfront.

(1459) Therefore, to attain the objectives of the interoperability obligation set out in Article 6(7) DMA, Alphabet shall implement the measures in the Annex in the Section "Assistance and testing" (Section 5.8).

### 12.9. Future updates and new functionalities

(1460) Article 6(7) DMA aims to ensure that third parties are able to provide their services and hardware on a level playing field with the gatekeeper's services and hardware. This is why Alphabet must provide third parties with interoperability solutions that are equally effective to those available to Alphabet itself.

(1461) To achieve a level playing field, it is crucial that third parties obtain effective access to any updates of features or functionalities thereof, and to any new functionalities of features subject to this Decision, at the same time and under the same conditions as Alphabet. In practice, this means that third parties need to be informed of any such planned updates or new feature functionalities and be able to test them and obtain access to them at the same time as they are available to Alphabet's own services or products.

(1462) A deprecation or discontinuation of an interoperability solution could significantly impact the functioning of third-party services and hardware and therefore the developer's investments, and ultimately the incentive to start and continue innovating based on Google Android interoperability solutions. Third parties cannot be obliged to adapt their implementation at short notice or lose interoperability in response to

[1728] Alphabet's response to the PFs, Doc ID […], paragraph 362.
[1729] Alphabet's response to the PFs, Doc ID […], paragraph 361.
[1730] See https://www.google.com/android/beta, Doc ID […], accessed on 10 July 2026.

deprecation, as they depend on these interoperability solutions for their services and hardware.

(1463) Deprecation of interoperability solutions for any of the features listed in this Decision would therefore cause substantial harm to third parties' ability to provide existing and new services and hardware. The risk of deprecation in the future would also diminish third parties' incentives to make the substantial investments that may be necessary for innovation and contestability of Alphabet's services and hardware.

(1464) Notwithstanding the above and contrary to Alphabet's claims that the Commission does not consider the need to deprecate features over time,[1731] the Commission acknowledged (including in the PFs), that, over time, certain software and hardware capabilities may become obsolete, go out of fashion, or be superseded by better alternatives, including in response to improvements in integrity standards or the discovery of vulnerabilities. In such cases, the gatekeeper may decide, subject to the requirements of Article 6(7) DMA, to adjust or deprecate (parts of) an interoperability solution that it provides to third parties. When deciding on the deprecation of interoperability solutions, the gatekeeper should refrain from taking deprecation decisions that could constitute a circumvention of Article 6(7) in the meaning of Article 13 DMA, such as degrading the conditions or quality of Google Android as provided to third parties who avail themselves of the rights in Article 6(7).

(1465) Deprecation of features subject to this Decision is likely to be exceptional, as those features are important building blocks necessary for effectively operating an AI service (Section 7), and likely of high value to Alphabet's own services as well. Due to this, and given the narrow scope of the features in this Decision, the Commission considers it proportionate that deprecation of an interoperability solution or parts of it, is subject to a reasoned request in accordance with the procedure described in Section 12.11, in order to ensure that the deprecation does not result in the loss of effective interoperability or the reinstatement of an interoperability gap.

(1466) Therefore, to attain the objectives of the interoperability obligation set out in Article 6(7) DMA, Alphabet shall implement the measures in the Annex in the Section "Future updates and new functionalities" (Section 5.9).

### 12.10. Reporting

(1467) A timely and effective implementation of the measures in the Annex is key to meet the objectives of Article 6(7) DMA.

(1468) Pursuant to Article 26 DMA, the Commission shall monitor gatekeepers' compliance with a decision adopted pursuant to Article 8(2) DMA. To this end, in the context of the present case, the Commission considers that reporting measures are necessary and proportionate to ensure the timely and effective implementation of the measures. Such reporting is also in line with the gatekeepers' duties under Article 8(1) DMA.

(1469) In particular, the Commission must be able to assess whether the solutions to be implemented by Alphabet achieve effective interoperability, such that any issues can be promptly raised with Alphabet and remediated. Otherwise, there is a risk that the interoperability solution released to third parties is not effective, causing further undue delay to third parties' right to interoperability under Article 6(7) DMA. The Commission must be able to assess the solutions' design, to monitor that it does not impede effective interoperability; their development, to monitor any barriers to

[1731] Alphabet's response to the PFs, Doc ID […], paragraph 363.

timely implementation; their implementation, to monitor that the final solution provides effective interoperability; and their release, to monitor that the final solution meets the required deadlines, complies in practice with all the requirements set out in the Annex, the Decision, and reflects the engagement of Alphabet with the Commission and third parties across the previous stages. To make this possible, Alphabet must provide five types of reports.

(1470) The Commission describes each of these reports and explains why they are necessary to enable the Commission to properly exercise its monitoring powers below.

(1471) The scope, timeframe and the frequency at which each of the five reports must be produced are aligned with the specific monitoring needs of the corresponding stage (i.e. design, development, implementation or release stage). Alphabet may comply with these reporting measures through existing reporting processes and materials (e.g. reports for internal stakeholders), as long as these comprise the required information, in line with Section 5.10 of the Annex. The Commission does not prescribe any formatting requirements for any of the required reports.

(1472) *First,* Alphabet must provide an initial implementation report ("Initial Implementation Report"), shortly, and, in any event, within maximum two months of the date of the notification of this Decision. This report must describe the interoperability solutions that Alphabet intends to make available for each measure, and how these solutions address all the measures in the Annex. The Initial Implementation Report must also contain a detailed timeline for the design, development, implementation and release of each of the interoperability solutions.

(1473) This information is necessary to enable the Commission to assess early on whether the envisaged solutions are prima facie complete, workable, and satisfy the requirements set out in this Decision and the Annex. Such information lays the foundation for a constructive and effective engagement with Alphabet throughout the implementation phase and enables early-stage technical engagement, thereby minimising the risk of rollout delays). The Initial Implementation Report must also contain a description of any integrity measures that Alphabet intends to apply. Such information will allow the Commission to verify that any potential integrity measure would comply with the provisions of Article 6(7) DMA concerning integrity, as detailed in Section 4.8 of this Decision and in Section 5.3 of the Annex.

(1474) *Second,* following the Initial Implementation Report, Alphabet must provide a monthly implementation report ("Monthly Implementation Report") for each measure until the expiry of the implementation deadlines for all features. This report must contain the envisaged interoperability solutions, the status of the design, development and release of each of the interoperability solutions, and whether progress is on schedule and, if not, why not. If necessary, Alphabet must provide an updated timeline for the design, development, implementation and release of each solution. Alphabet must also report any potential issue that it may have encountered during the design, development, implementation or release of each of the envisaged interoperability solutions. Alphabet must also describe its engagement with third parties concerning each interoperability solution. The Monthly Implementation Report must also contain any relevant changes that Alphabet has made or intends to make to policies, requirements and agreements (including the CDD, the GMS, placement agreements etc.), as these may affect the effectiveness of the interoperability solution, particularly if they resulted in preferential access to Google Android features for Alphabet's own services. Finally, since non-Pixel devices account for the majority of Google Android devices, the Monthly Implementation

Report must provide a description of whether and how OEMs have customised or plan to customise the interoperability solutions, including any relevant rules (e.g. rules on background execution), to monitor compliance with the measures described in Section 5.1 of the Annex.

(1475) Such information will enable the Commission to effectively sustain its engagement with Alphabet, track the implementation progress and assess the effectiveness of the interoperability solutions in practice. The required information will also allow the Commission to properly assess whether any potential delays are justified, and to swiftly engage with Alphabet to address any potential obstacles which may otherwise delay compliance. The information concerning Alphabet's engagement with third parties will allow the Commission to understand third parties' needs and the potential obstacles they may encounter on the path towards obtaining effective interoperability with Google Android. It also ensures Alphabet's accountability and encourages the gatekeeper to engage meaningfully and timely with third parties.

(1476) *Third,* prior to any public release (e.g. be it beta testing or the final solution) of an interoperability solution, Alphabet must provide a pre-release feature implementation report ("Pre-release Feature Implementation Report"), communicating in advance any public release plans to the Commission.

(1477) This report is essential to ensure that the solution planned for public release by Alphabet reflects the one discussed during the design, development and implementation stages, and that Alphabet will allow effective testing for any interested third party.

(1478) *Fourth,* Alphabet must provide a final feature implementation report ("Final Feature Implementation Report") upon the expiry of the implementation deadline for each feature. In this report, Alphabet must explain all the measures it has taken to comply with this Decision and the Annex and confirm the date of the public release of the interoperability solutions. The Final Feature Implementation Report must describe the interoperability solutions made available to third parties, including all the technical details and potential APIs, as well as any potential integrity measures.

(1479) Such information allows the Commission to verify whether Alphabet is meeting the prescribed deadlines, and a clear mapping of the released technical solutions and of any potential integrity measures allows the Commission to monitor the effectiveness and continuity of the interoperability solutions.

(1480) *Fifth,* Alphabet must provide a semi-annual implementation report ("Semi-annual Implementation Report") every six months, for two years after the expiry of the implementation deadlines for all features, containing the same information as the Monthly Implementation Report.

(1481) This report allows the Commission to monitor that Alphabet properly maintains effective interoperability solutions, and does not raise, after the release of the solution, any undue obstacles of any nature, including of a technical or contractual nature. Such information also allows the Commission to monitor the effectiveness of the solutions in practice, and to assess whether Alphabet engages promptly and meaningfully with third parties.

(1482) Alphabet must also include, in particular, in each of the five types of reports mentioned above any changes that Alphabet plans to make or it has made in relation to the features listed in the Annex (including any new functionality it plans to add or it has added to the feature or any updates it plans to make or it has made to such

feature or its functionalities). This enables the Commission to monitor compliance with the measures in Section 5.9 of the Annex.

(1483) A proper assessment of the interoperability solutions might require the Commission to consult with third parties, as the beneficiaries who will ultimately adopt these solutions having relevant interest and expertise to assess the effectiveness of the solutions. This will also provide third parties with transparency and predictability on the interoperability solutions and their stage, allowing them to allocate the necessary resources to adopt and test the interoperability solutions, which is essential for ensuring effective interoperability. To enable effective third-party consultation and participation, Alphabet may therefore also be required to provide the Commission with non-confidential versions of the reports, in which business secrets and other confidential information of any natural or legal person are redacted in a clear and intelligible manner.

(1484) Alphabet claimed that the reporting obligations, in particular, the monthly reporting, are disproportionate and unnecessary, and would divert attention and resources from the actual implementation of the measures.[1732] Except for the reporting on OEM customisations, which is addressed below, Alphabet merely reproduces the text of the measures without any further comment or claim. In Alphabet's markups to the draft final measures, exchanged in the context of post-PFs engagement, Alphabet simply stroke the measures relative to three out of the five types of reports, and tripled the deadline for the Initial Implementation Report.[1733]

(1485) In any event, the Commission notes that, under Articles 6(1) and 8(1) DMA, Alphabet must ensure and demonstrate compliance with this Decision, by effectively developing and implementing the measures in the Annex. In this context, and as explained above, the Commission considers it essential to engage with Alphabet on a regular basis, and from the earliest stages when devising the interoperability solutions. This is necessary to ensure that Alphabet observes the principles set out in this Decision and the measures set out in the Annex and, more generally, that Alphabet's interoperability solutions align with the DMA's objectives of fairness and contestability. Such regular engagement with the Commission may in fact speed up Alphabet's compliance and save resources, as it can allow recalibrations while Alphabet is devising the solutions, avoiding more costly later-stage changes.

(1486) Alphabet also asserted that describing how OEMs have implemented customisations will mostly be outside of Alphabet's knowledge.[1734] The Commission notes that it falls under its powers under Article 8(2) DMA to specify that Alphabet is required to report on the customisations implemented by OEMs. This obligation has the sole purpose of ensuring effective monitoring of Alphabet's compliance with the final measures in the Annex.

(1487) Therefore, to attain the objectives of the interoperability obligation set out in Article 6(7) DMA, Alphabet shall implement the measures in the Annex in the "Reporting" (Section 5.10).

### 12.11. Waiver

(1488) Given the potential technical complexity associated with the implementation of some of the measures set out in the Annex to this Decision, it cannot be excluded that, in

---

1732 Alphabet's response to the PFs, Doc ID […], paragraphs 364 and 365.

1733 Alphabet's first draft measures redline of 20 May 2026, Doc ID […], paragraphs 152-156. Alphabet's second draft measures redline of 28 May 2026, Doc ID […], paragraphs 152-156.

1734 Alphabet's response to the PFs, Doc ID […], paragraph 364.

exceptional circumstances, particularly those relating to the technical implementation, Alphabet may not be in a position to implement one or more of the measures, or a part of them.

(1489) In such cases, the Commission may, in response to a reasoned request from Alphabet showing good cause, modify or substitute one or more of the measures or a part of them. To ensure the effectiveness and timely implementation of the measures, it is important that the request does not have the effect of suspending the implementation of the measures and, in particular, of suspending the expiry of any time period in which the measure has to be complied with.

## 13. PROPORTIONALITY

(1490) Article 6(7) DMA is an obligation susceptible of being specified. A specification decision pursuant to Article 8(2) DMA sets out the measures that the gatekeeper is required to implement in order to comply with Article 6(7) DMA. Pursuant to Article 8(7) DMA, in specifying the measures that the gatekeeper concerned is to implement in order to comply with its obligations, the Commission shall ensure that the measures are effective in achieving the objectives of the DMA and the relevant obligation and that they are proportionate in the specific circumstances of the gatekeeper and the relevant service. Subject to the principle of proportionality, limitations on the exercise of the rights and freedoms recognised by the Charter may be made only if they are necessary and genuinely meet objectives of general interest recognised by the Union or the need to protect the rights and freedoms of others.[1735]

(1491) In that regard, Article 6(7) DMA pursues the legitimate objectives of contestability and fairness in digital markets by requiring effective interoperability between the operating system or virtual assistant and third-party services. It is an appropriate and necessary means to ensure contestability and fairness. In the absence of an effective interoperability obligation, third parties wishing to offer a service running on a gatekeeper's operating system would depend on the goodwill of that gatekeeper to do so. Ensuring contestability and fairness in digital markets is a cornerstone of fair competition, itself a core principle of Union law.

(1492) The Commission also notes that the second subparagraph of Article 6(7) DMA allows the gatekeeper to take strictly necessary and proportionate measures to ensure that interoperability does not compromise the integrity of the operating system. The possibility to take justified measures to protect the integrity of the operating system allows for a proportionate application of Article 6(7) DMA. Among several equally effective ways to protect integrity, the least restrictive way should be selected.[1736]

(1493) Furthermore, the principle of proportionality requires that the measures specified by the Commission are appropriate and no more restrictive than necessary in the specific case, namely to achieve effective compliance with Article 6(7) DMA, taking due account of the legislative decision for a clear and self-executing[1737] obligation enshrined in Article 6(7) DMA and the objectives pursued by the DMA.[1738] In particular, regard must be had to the legislative decision to confer on the Commission regulatory powers to ensure that gatekeepers effectively comply with

---

1735 Article 52(1) of the Charter.

1736 See recital (200).

1737 Recital 5 DMA shows that the legislator sought to adopt rules that can be enforced without an "extensive investigation of often very complex facts on a case by case basis".

1738 Joined cases C-241/91 P and C-242/91 P, Magill, ECLI:EU:C:1995:98, paragraph 93; Case T-201/04, Microsoft v Commission, ECLI:EU:T:2007:289, paragraph 1276.

the obligations in Articles 5, 6 and 7 DMA and achieve its objectives and the objectives of the relevant obligation by means of implementing acts pursuant to Article 8(2) DMA.[1739]

(1494) In the context of its dialogue with the Commission, the gatekeeper should present less restrictive, but equally effective ways of complying with Article 6(7) DMA or substantiate that compliance with that Article is inappropriate in a specific case. The gatekeeper, as the best-placed entity to do so, should provide clear, convincing evidence in that regard.

(1495) As regards claims based on fundamental rights protected by the Charter, it is upon the gatekeeper to make and substantiate such a claim. The assessment must take into account that the legislator has, in principle, already considered and weighed the fundamental rights which are intrinsically linked to the interoperability provision.[1740]

(1496) Alphabet claims with respect to all features covered in this Decision that all or some of the measures adopted in this Decision are disproportionate and/or infringe its fundamental rights.

(1497) Given the clear decision by the legislator to require interoperability for the operating system-controlled features available to gatekeepers, the legitimate objectives pursued by Article 6(7) DMA, and the importance of interoperability for business and end users, it is insufficient to submit generic or speculative disproportionality arguments. Equally, any alleged infringements to fundamental rights must be substantiated.

(1498) The Commission addresses Alphabet's claims relating to the lack of proportionality and breach of fundamental rights for each feature in Sections 8 to 11.[1741] As is clear from these feature-specific discussions, in many cases Alphabet's claims of lack of proportionality aim at contesting the scope of Article 6(7) DMA instead of arguing for a less restrictive, but equally effective means to comply with that Article or substantiating to the requisite standard that application of the provision in the specific case puts a disproportionate burden on Alphabet. Such claims are insufficient to overrule the legislative decision enshrined in Article 6(7) DMA.

(1499) In particular, Alphabet contends that the measures annexed to this Decision have no regard to (i) privacy and security interests of end users and third parties, (ii) the commercial freedom and compliance burden of OEMs and (iii) Alphabet's fundamental rights to do business.[1742] Alphabet's claims are discussed in each of the dedicated sections of this Decision under Sections 8 to 11, and the Commission makes the following remarks.

(1500) *First,* as regards Alphabet's claim that the measures annexed to this Decision do not properly take account of the privacy and security interests of end users and third parties, it should first be pointed out that not all privacy and security interests fall within the concept of integrity set out in Article 6(7) DMA as explained in Section 4.8 of this Decision. Nevertheless, even from the point of view of privacy and security interests, the measures specified in this Decision duly take account of users' privacy and security interests, including by implementing relevant user consent mechanisms. Similarly, as explained in Section 12.4, this Decision allows Alphabet to limit access to some of the features covered by this Decision to third parties

1739 See recital (99).

1740 Case C-201/11 P, UEFA v Commission, ECLI:EU:C:2013:519, paragraphs 10, 20, 21, 101-102.

1741 See in particular Sections 8.2.8.2, 8.3.7.4, 9.2.8.4, 9.3.8.2, 9.4.8.2, 10.2.8.2, 10.3.8.2, 10.4.8.2, 11.2.7.2, 11.3.7.2, and 11.4.8.2.

1742 Alphabet's response to the PFs, Doc ID […], paragraphs 175-178.

meeting criteria including those relating to privacy and security. Sections 4.8 and 6.3 explain in detail why Alphabet's current permission model is not an appropriate integrity measure. Alphabet does not properly substantiate its generic claim that a proportionate measure must take into account the impact on end users regardless of whether the impacts fall within the integrity framework. In particular, Alphabet does not substantiate that any fundamental rights of users would be disproportionately interfered with by the measures annexed to this Decision, particularly in view of the fact that all features covered by this Decision are already used by Alphabet's services. In addition, as further explained in Section 4.8.2, Alphabet and third parties alike remain under the obligation to comply with the regulatory framework they are subject to, including the Cyber Resilience Act and the GDPR. Finally, as further detailed in Section 4.8.4 above, a gatekeeper cannot justify an integrity measure (or for that matter, a privacy or security measure, given that equally effective interoperability must be provided under equal conditions) implemented in relation to third parties' services or hardware on the basis of a general distrust towards third parties or an assumption that control by the gatekeeper must always be ensured. Nor can a measure be considered disproportionate simply based on the assumption that third parties might not abide by the legal framework they are bound by.

(1501) *Second,* as regards Alphabet's claim relating to the commercial freedom and compliance burden of OEM, as explained in Section 6.1.2 above, this Decision does not have a disproportionate impact on third-party OEMs. In particular, OEMs remain free to design the out-of-the-box experience as they see fit, including through (i) pre-installing certain apps, setting default services, offering innovative complementary software and user interface layouts, as well as on the basis of hardware features (e.g. camera, battery, memory space, etc) or (ii) providing attractive experiences for users and maintaining branding and product feel.[1743] OEMs will also continue to be able to effectively monetise their investments to end users (e.g. via device sales) and app developers (e.g., for pre-installation and default-setting, including with respect to apps which can generate continuous revenues for OEMs such as app stores).[1744]

(1502) *Third,* Alphabet's claim that some of the measures annexed to this Decision disproportionately interfere with its fundamental rights to conduct business and to property is generic and unsubstantiated. It is clear from Article 52(1) of the Charter that restrictions can be imposed on the exercise of the right to property, provided that the restrictions genuinely meet objectives of general interest and do not constitute, in relation to the aim pursued, a disproportionate and intolerable interference impairing the very substance of the right guaranteed. Alphabet neither substantiates why the measures specified in this Decision would infringe its intellectual property rights (and in particular does not refer to any specific intellectual property right which would be infringed), nor, *a fortiori*, does it explain why this alleged infringement would be disproportionate with regards to the objectives pursued by Article 6(7) DMA.[1745]

## 14. CONCLUSION

(1503) For the reasons set out above, the Commission hereby specifies measures pursuant to Article 8(2) DMA that Alphabet has to implement within the deadlines specified in the Annex to this Decision to effectively comply with the obligations under Article

[1743] Alphabet submission of 17 March 2026, Doc ID […], paragraphs 20-21, 25-28.
[1744] Alphabet submission of 17 March 2026, Doc ID […], paragraph 22.
[1745] See Sections 8.3.7 (always-on hotword detection feature) and 11.2.7 (system-wide ODM feature).

6(7) DMA in relation to certain Google Android features used by or available to Alphabet's services or hardware.

HAS ADOPTED THIS DECISION:

*Article 1*

Alphabet shall implement the measures as specified in the Annex to this Decision within the deadlines specified therein pursuant to Article 8(2) of Regulation (EU) 2022/1925.

*Article 2*

This Decision is addressed to Alphabet Inc., 251 Little Falls Drive, Wilmington, DE, 19808, United States, and Google Ireland Ltd, Google Building Gordon House, 4 Barrow St, Grand Canal Dock, Dublin 4, D04 V4X7, Ireland.

Done at Brussels, 16.7.2026

*For the Commission*

*(Signed)*

Teresa RIBERA
*Executive Vice-President*

# DMA.100220 – Alphabet - OS - Google Android - Art. 6(7) – SP – AI

## Decision of 16 July 2026 – Final Measures

## Non-Confidential Version

*[This version has been adapted for publication. Only the adopted decision is legally binding]*

## TABLE OF CONTENTS


1. Measures on features for invocation ..... 1
1.1. Measures for Long-press home/Long-press navigation handle contextual invocation ..... 1
1.2. Measures for always-on hotword detection ..... 2
2. Measures for context ..... 3
2.1. Measures for centralised access to apps' data stored on-device ..... 3
2.2. Measures for context-aware intelligence ..... 5
2.3. Measures for access to ambient data ..... 7
3. Measures on features for actions on apps and the OS ..... 8
3.1. Measures for structured on-device integration ..... 8
3.2. Measures for screen automation ..... 10
3.3. Measures for system integration ..... 12
4. Measures on features for access to resources ..... 13
4.1. Measures for system-level on-device models ..... 13
4.2. Measures for on-device model implementation ..... 14
4.3. Measures for background execution ..... 16
5. Measures for all features ..... 18
5.1. Implementation across the Google Android ecosystem ..... 18
5.2. User consent ..... 18
5.3. Integrity ..... 18
5.4. Eligibility of beneficiaries, applications and use cases ..... 20
5.5. Equal effectiveness ..... 24
5.6. Free of charge ..... 25
5.7. Documentation and APIs ..... 25
5.8. Assistance and testing ..... 25
5.9. Future updates and new functionalities ..... 26
5.10. Reporting ..... 26
5.11. Waiver ..... 28

# ANNEX - MEASURES

## 1. MEASURES ON FEATURES FOR INVOCATION

### 1.1. Measures for Long-press home/Long-press navigation handle contextual invocation

(1) Alphabet shall provide effective interoperability with the *Long-press home (LPH)/Long-press navigation handle (LPNH) contextual invocation* feature.

(2) The LPH/LPNH contextual invocation feature is an access point feature described in Section 8.2.2 of the Decision. Access points enable Alphabet's services to be invoked by users at any time while using the device. The LPH/LPNH access point, depending on the configuration of the device, invokes Google Search's Circle to Search solution to overlay on top of the screen and offer a service based on contextual data, such as screen content.

(3) Alphabet shall implement an interoperability solution that provides third parties[1] with access to the LPH/LPNH contextual invocation feature and its functionalities, as available to Alphabet's services, such as Google Search via the Circle to Search solution, and in a way that is equally effective as the solution available to Alphabet's services.

(4) These functionalities are:

(a) Ability to be invoked system-wide through the LPH/LPNH access point and any other access points available to Alphabet's services, such as Google Search (via the Circle to Search solution);

(b) Upon invocation, ability to receive contextual data, including but not limited to a current screenshot, information about the open apps, information provided by apps in the foreground, such as a URL, picture, or additional data as available to Alphabet's services, such as Google Search (via the Circle to Search solution); and

(c) Upon invocation, the ability to overlay content.

(5) Alphabet shall grant third parties access to additional functionalities available to Alphabet's services if necessary to enable effective interoperability with LPH/LPNH contextual invocation feature.

(6) To provide third parties with an interoperability solution for the LPH/LPNH contextual invocation feature and functionalities that is equally effective as that available to Alphabet's own services, such as Google Search, the following measures shall apply:

(a) Alphabet shall make available to third parties the means for invocation through LPH/LPNH and any other access point. Alphabet shall ensure that third-parties can receive upon invocation through an access point the same contextual data as available to Alphabet's services, such as the data made available to Google Search when invoked via Circle to Search;

[1] The terms "third parties", and "third-party services" across this Annex include all apps, including user-installed apps, pre-installed apps without system privileges, and apps of original equipment manufacturers (OEMs). User-installed apps are apps that are only installed following a decision by the user. Apps that are pre-installed on the Google Android mobile device without the user making such decision (pre-installed apps) are not user-installed apps.

(b) Alphabet shall ensure that the user can customise the LPH/LPNH access point and any other access point, subject to the possibility of OEMs to reserve one access point other than the LPH/LPNH to the default assistant role;

(c) Subject to point (b) above, Alphabet shall not reserve, including through technical or contractual means, any access point on Google Android mobile devices for its services; and

(d) Alphabet shall ensure that users can easily select which app is invoked through any access point. The mechanism allowing the user to exercise this choice must be non-discriminatory, including in terms of user interface.

(7) Alphabet shall provide effective interoperability with any future updates, including new functionalities, of the LPH/LPNH contextual invocation feature insofar as they are available to Alphabet's own services.

(8) Alphabet shall implement the measures above in compliance with the measures for all features in Section 5 of this Annex.

(9) Alphabet shall implement the measures for the LPH/LPNH contextual invocation feature in the next major Android release, i.e. Android 18, and by 1 August 2027 at the latest.

**1.2. Measures for always-on hotword detection**

(10) Alphabet shall provide effective interoperability with the *always-on hotword detection (AOHD)* feature.

(11) The AOHD feature is described in Section 8.3.2 of the Decision. The feature enables Alphabet services, such as Gemini, to be invoked when the user utters a hotword (also known as "wake word"). Hotwords allow users to invoke and interact, hands free, with Alphabet's services, such as Gemini. The feature is implemented in an energy-saving manner that allows AOHD to run continuously, even when the phone screen is turned off and when the phone is in battery saver mode. The feature is currently implemented based on a digital signal processor (DSP) which depends on the characteristics of the device, as decided by OEMs.

(12) Alphabet shall implement an interoperability solution that provides third-party services with access to the AOHD feature and its functionalities, as available to Alphabet's services and in a way that is equally effective as the solution available to Alphabet's services, such as Gemini.

(13) These functionalities are:

(a) **Sound model enrolment.** The ability to enrol a sound model in the Android operating system to persistently store and retrieve the sound model upon request. An enrolled sound model can be loaded on the DSP when the app starts AOHD;

(b) **First-stage detection on the DSP.** The ability to load and execute its enrolled sound model on the DSP, and to ensure that the DSP performs the first stage hotword detection on the app's behalf;

(c) **Second-stage validation.** The ability to run a second stage validation after the DSP sound model has recognised the hotword in the first-stage detection, to confirm that the hotword was detected correctly. The second stage may run in a secure and isolated process, disallowing to extract audio data before the hotword was validated;

(d) **Invocation of the app.** The ability of the app to be invoked and to handle the request of the user; and

(e) **Continued access to record audio.** The ability, after invocation, to receive the audio of the hotword detection and to continue to record the audio until the user decides to finish its request or conversation.

(14) Alphabet shall grant third parties access to additional functionalities available to Alphabet's services if necessary to enable effective interoperability with the AOHD feature.

(15) To provide third parties with an interoperability solution for the AOHD feature that is equally effective as that available to Alphabet's services, such as Gemini, the following measures shall apply:

(a) Alphabet shall make available to developers testing tools enabling developers to verify, amongst others, that their AOHD implementation correctly recognises the hotword phrase;

(b) Alphabet shall ensure that, upon invocation, the third party receives access to the same contextual data that is available to Alphabet's services, such as Gemini;

(c) Alphabet shall ensure that Google Android users have control over the apps using AOHD on their device through mechanisms such as central settings. The mechanism allowing users to enable or disable AOHD shall be non-discriminatory, including in terms of user interface;

(d) Alphabet shall ensure that AOHD for third parties can continue to run while the screen is locked or the device is in battery saver mode;

(e) Alphabet may apply technical measures such that audio data for the first-stage detection and second-stage validation is processed in accordance with principles of process isolation, and that continuous access to audio data is only granted after the hotword has been detected and validated; and

(f) Alphabet shall ensure that AOHD from multiple services, including services belonging to third parties and Alphabet, are able to run concurrently, within the technical capabilities of the device.

(16) Alphabet shall provide effective interoperability with any future updates or implementations, including new functionalities, of the AOHD feature insofar as they are available to Alphabet's services, such as Gemini, or hardware.

(17) Alphabet shall implement the measures above in compliance with the measures for all features in Section 5 of this Annex.

(18) Alphabet shall implement the measures (15)(a) to 15(e) for the always-on hotword detection feature in the next major Android release, i.e. Android 18, and by 1 August 2027 at the latest. Alphabet shall implement the measure 15(f) in the following major Android release, i.e. Android 19, and by 1 August 2028 at the latest.

## 2. MEASURES FOR CONTEXT

### 2.1. Measures for centralised access to apps' data stored on-device

(19) Alphabet shall provide effective interoperability with the *centralised access to apps' data stored on-device* feature.

(20) The centralised access to apps' data stored on-device feature is described in Section 9.2.2 of the Decision. The feature enables access to on-device-stored apps' data that apps choose to share in a centralised manner through on-device databases or data sharing platforms, thereby allowing efficient cross-app data access, search and retrieval. The feature is currently implemented by Alphabet as AppSearch.

(21) As described in Section 12.4.5 of the Decision, the centralised access to apps' data stored on-device feature, as implemented in AppSearch, meets the requirements of a restricted feature. Consequently, access to the functionalities referenced below in this section may be limited to Qualified Services.

(22) Alphabet shall implement an interoperability solution that provides third parties with access to the centralised access to apps' data stored on-device feature and its functionalities, as available to Alphabet's AI-powered services, such as Gemini, and in a way that is equally effective as the solution available to Alphabet's AI-powered services, such as Gemini.

(23) These functionalities are:

(a) **Ability to search** the on-device data that apps choose to share on an opt-in role-based basis; and

(b) **Ability to retrieve** the on-device data that apps choose to share on an opt-in role-based basis.

(24) Alphabet shall grant third parties access to additional functionalities available to Alphabet's services if necessary to enable effective interoperability with the centralised access to apps' data stored on-device feature.

(25) To provide third parties with an interoperability solution for the centralised access to apps' data stored on-device feature that is equally effective as that available to Alphabet's AI-powered services, such as Gemini, the following measures shall apply:

(a) Alphabet shall ensure access by third-party services to on-device data that apps choose to share on an opt-in role-based basis, in a centralised manner;[2]

(b) Alphabet shall ensure that data that is stored on device by Alphabet's apps and that is shared with Alphabet's AI-powered services, such as Gemini, for the purposes of answering a user query is accessible by third-party services under the same conditions; and

(c) Alphabet shall enable concurrent use of the centralised access to apps' data stored on-device feature.

(26) Alphabet shall provide effective interoperability with any future updates or implementations, including new functionalities, of the centralised access to apps' data stored on-device feature insofar and as soon as they are available to Alphabet's AI-powered services, such as Gemini.

(27) Alphabet shall implement the measures above in compliance with the measures for all features in Section 5 of this Annex.

---

[2] With AppSearch, this may be enabled by allowing third-party services to access the data stored with READ_ASSISTANT_APP_SEARCH_DATA visibility. However, the measure applies regardless of the technical implementation of the centralised access to apps' data stored on-device feature.

(28) Alphabet shall implement the measures for the centralised access to apps' data stored on-device feature in the next major Android release, i.e. Android 18, and by 1 August 2027 at the latest.

### 2.2. Measures for context-aware intelligence

(29) Alphabet shall provide effective interoperability with the *context-aware intelligence* feature.

(30) The context-aware intelligence feature is described in Section 9.3.2 of the Decision. Context-aware intelligence allows providing proactive suggestions and "intelligent experiences", i.e. experiences that are personalised to the user's context and can be continuously active to enhance the user experience. This relies, for example, on accessing inputs from device sensors and processing those inputs to enable proactive actions.

(31) As described in Section 12.4.5 of the Decision, the context-aware intelligence feature meets the requirements of a restricted feature. Consequently, access to the functionalities referenced below in this section may be limited to Qualified Services.

(32) Alphabet shall implement an interoperability solution that provides third parties with access to the context-aware intelligence feature and its functionalities, as available to Alphabet's services, and in a way that is equally effective as the solution available to Alphabet's services.

(33) These functionalities are:

(a) **Access to inputs**:

(1) Access to the user's physical context, as used by or available to Alphabet's services. This includes ambient data, such as data from the camera, microphone, accelerometer, proximity sensor, GPS, and any other location signals (see Section 2.3 of the Annex);

(2) Access to the user's digital context, as used by or available to Alphabet's services. This includes, but is not limited to, device audio, screen contents and context, screenshots, relevant information on the foreground app, notifications, app launches, contacts, shortcuts, SMS, and app package details;

(3) The ability to retrieve data provided ad hoc by an app, service, or the OS to the proactive suggestion or intelligent experience, as used by or available to Alphabet's services; and

(4) Global access to apps' data and actions stored on-device that apps choose to donate, including through an on-device database or data sharing platforms allowing for centralised data access and sharing, as used by or available to Alphabet's services (see Section 2.1 of the Annex).

(b) **Access to processing resources**:

(1) Ability to use on-device models to process the input data and generate the outputs, both those already present on the device (see Section 4.1 of the Annex) and those implemented by third parties (see Section 4.2 of the Annex).

(2) Ability to use communication channels to send and receive data to the solution's cloud-based models to process the input data and generate the

outputs, in accordance with paragraph (35)(i) of the Annex, and as used by or available to Alphabet's services;

(3) Ability to use protected execution environments allowing to process particularly sensitive data, such as ambient data, in isolation, in support of paragraph (35)(i) of the Annex, and as used by or available to Alphabet's services.

(c) **Access to output channels and display surfaces**:

(1) Ability to present proactive suggestions or intelligent experiences on a surface visible and/or audible to the user in the user interface, mediated by the OS, as used by or available to Alphabet's services;

(2) Ability to transmit data from a proactive suggestion or intelligent experience to an app, service, or the OS, as used by or available to Alphabet's services; and

(3) Ability to trigger integrations and interactions with apps and the OS from the proactive suggestion or intelligent experience, such as redirecting to another app, opening an app in an overlay, or deep linking to parts of the OS such as settings.

(34) Alphabet shall grant third parties access to additional functionalities available to Alphabet's services if necessary to enable effective interoperability with the context-aware intelligence feature.

(35) To provide third parties with an interoperability solution for the context-aware intelligence feature that is equally effective as that available to any of Alphabet's services, the following measures shall apply:

(a) Alphabet shall enable access to device sensors, outputs, and data as specified under paragraph (33)(a) of the Annex on a continuous basis and in the background, as are used by or available to Alphabet's services;

(b) Alphabet shall grant access to device sensors, outputs, and data as specified under paragraph (33)(a) of the Annex with the same degree of granularity, frequency, and recency, as are used by or available to Alphabet's services;

(c) Alphabet shall allow third parties offering proactive suggestions and intelligent experiences to tailor the surfacing experience, when accessing display surfaces as specified under paragraph (33)(c) of the Annex, for instance in terms of including the branding, icon, or link for the apps the surfaced information has been sourced from, or in terms of contributing to surfacing decisions such as which surface to use, and as used by or available to Alphabet's services;

(d) Alphabet shall enable access to digital context, as specified under paragraph (33)(a)(2) of the Annex, data provided ad hoc, as specified under paragraph (33)(a)(3) of the Annex, apps' data and actions stored on-device, as specified under paragraph (33)(a)(4) of the Annex, of Alphabet's first-party apps, as used by or available to Alphabet's services;

(e) Alphabet shall enable access to surfaces in/on Alphabet's first-party apps, as specified under paragraph (33)(c)(1) of the Annex, on a non-discriminatory basis between Alphabet and third parties offering proactive suggestions and intelligent experiences;

(f) Alphabet may enable apps to opt out of the ability for components offering proactive suggestions and intelligent experiences to access that app's digital

context and/or to appear in a surface in/on that app, either for the whole app or for specific parts of the app, where that opt-out applies to all such components on a non-discriminatory basis between Alphabet's own services and third-party services.

(g) Alphabet shall enable concurrent use of the context-aware intelligence feature, within the technical capabilities of the inputs, resources, and outputs. Where such concurrent use is not possible or is limited, Alphabet shall enable the use of the context-aware intelligence feature on a non-discriminatory basis between Alphabet's own services and third-party services;

(h) Alphabet shall not be required to enable the replacement of existing intelligence components, such as Android System Intelligence; and

(i) Alphabet may implement measures requiring that, when accessing inputs as specified under paragraph (33)(a) of the Annex, processing resources as specified under paragraph (33)(b) of the Annex, and output channels and display surfaces as specified under paragraph (33)(c) of the Annex, third parties must do so in line with principles of process isolation, encryption, anonymous and aggregated telemetry, and user control and awareness, insofar as this is based on transparent, objective, precise, and non-discriminatory conditions that also apply to Alphabet's services.

(36) Alphabet shall provide effective interoperability with any future updates, including new functionalities, of the context-aware intelligence feature insofar as they are available to Alphabet's services.

(37) Alphabet shall implement the measures above in compliance with the measures for all features in Section 5 of this Annex.

(38) Alphabet shall implement the measures for the context-aware intelligence feature in the next major Android release, i.e. Android 18, and by 1 August 2027 at the latest.

### 2.3. Measures for access to ambient data

(39) Alphabet shall provide effective interoperability with the *access to ambient data* feature.

(40) The access to ambient data feature is described in Section 9.4.2 of the Decision. Access to ambient data consists of the ability to access data produced or collected via a device's core sensors and data presented via the device's main output channels. The access enables AI services to deliver personalised, context-aware experiences such as sound detection, live screen guidance, or object recognition.

(41) Alphabet shall implement an interoperability solution that provides third parties with access to the access to ambient data feature and its functionalities, as available to Alphabet's AI-powered services and in a way that is equally effective as the solution available to Alphabet's AI-powered services.

(42) These functionalities are:

(a) the ability to access the microphone input,

(b) the ability to access speaker output (system audio),

(c) the ability to access the camera,

(d) the ability to access the screen contents,

(e) the ability to access location data, and

(f) the ability to access environmental sensors such as the accelerometer or proximity sensor.

(43) Alphabet shall grant third parties access to additional functionalities available to Alphabet's AI-powered services if necessary to enable effective interoperability with the access to ambient data feature.

(44) To provide third parties with an interoperability solution for the access to ambient data feature that is equally effective as that available to any of Alphabet's AI-powered services, the following measures shall apply:

(a) Alphabet shall provide access to ambient data under equal conditions as those that apply to Alphabet's AI-powered services, including in terms of continuous availability, quality, frequency, latency, and availability in the background, and any other technical characteristics necessary to enable equivalence in collection, transmission, access to, and processing of such data;

(b) Alphabet shall provide access under equivalent consent flows and indicators that apply to Alphabet's AI-powered services, including in terms of frequency of prompts, and in terms of the level of continuous availability, availability in the background, and the mechanism of processing;

(c) Alphabet shall allow third parties to access ambient data related to Alphabet's applications (such as screen content), whenever such functionality is enabled for Alphabet's AI-powered services;

(d) Where Alphabet's AI-powered services rely on dedicated apps, running on runtime environments for low-power processors, to access ambient data, Alphabet shall provide third parties with access to those dedicated apps; and

(e) Alphabet may implement measures requiring that, when accessing ambient data, third parties must do so in line with principles of process isolation, encryption, anonymous and aggregated telemetry, and user control and awareness, insofar as this is based on transparent, objective, precise, and non-discriminatory conditions which also apply to Alphabet's AI-powered services.

(45) Alphabet shall also provide effective interoperability with any future updates, including new functionalities, of the access to ambient data insofar as they are available to Alphabet's AI-powered services or hardware.

(46) Alphabet shall implement the measures above in compliance with the measures for all features in Section 5 of this Annex.

(47) Alphabet shall implement the measures for the access to ambient data feature in the next major Android release, i.e. Android 18, and by 1 August 2027 at the latest.

## 3. MEASURES ON FEATURES FOR ACTIONS ON APPS AND THE OS

### 3.1. Measures for structured on-device integration

(48) Alphabet shall provide effective interoperability with the *structured on-device integration* feature.

(49) The structured on-device integration feature is described in Section 10.2.2 of the Decision. This feature enables, in response to a user request, a service to take actions on other apps on the device. This feature is currently implemented through on-device integration channels, such as App Actions and App Functions.

(50) As described in Section 12.4.5 of the Decision, the structured on-device integration feature includes an implementation, namely App Functions, that meets the requirements of a restricted feature. Consequently, insofar as the functionalities referenced below in this section pertain to this implementation, access to the functionalities may be limited to Qualified Services.

(51) Alphabet shall implement an interoperability solution that provides third parties with access to the structured on-device integration feature and its functionalities, as available to Alphabet's services and in a way that is equally effective as the solution available to Alphabet's services.

(52) These functionalities are:

(a) The ability to discover all available structured on-device integrations, including metadata necessary to determine which actions to trigger;

(b) The ability to execute structured on-device integrations;

(c) The ability to let the user be aware of and observe that a structured on-device integration is being executed; and

(d) The ability to execute the structured on-device integration within the app, i.e. without app switching (unless that is the intent of the structured on-device integration).

(53) Alphabet shall grant third parties access to additional functionalities available to Alphabet's services, if necessary to enable effective interoperability with the structured on-device integration feature.

(54) To provide third parties with an interoperability solution for the structured on-device integration feature that is equally effective as that available to any of Alphabet's services, the following measures shall apply:

(a) Alphabet shall make available to third parties all channels of structured on-device integration, including App Actions and App Functions, in a manner that is equally effective as that available to Alphabet's own services;

(b) Alphabet shall enable third parties to discover all structured on-device integrations exposed by apps that are installed on the user's device, including a mechanism that ensures third parties can retrieve all metadata necessary to determine which actions to trigger; and

(c) Alphabet shall enable third parties to execute all structured on-device integrations exposed by other apps, including third parties, system apps, and Alphabet apps.

(55) Through the structured on-device integration feature, Alphabet shall enable Qualified Services to execute specific functionalities in the relevant Alphabet's first-party apps. The relevant Alphabet apps and functionalities are as follows:

(1) Gmail: Retrieve and return relevant emails and draft, edit, and reply to emails;

(2) Google Calendar: The ability to retrieve information about events and to create and manage events;

(3) Google Drive: The ability to retrieve relevant information from files and return information about their content;

(4) Google Docs: The ability to retrieve relevant information from files and return information about their content;

(5) Google Maps: The ability to trigger navigation to a general public place in Google Maps or to the end user's designated home or work address, or to other labelled places, and the ability to control the navigation;

(6) YouTube: The ability to play a video in YouTube, the ability to control the video playback, and the ability to answer user queries regarding recent watch history activity;

(7) Messages: The ability to retrieve relevant information from SMS, MMS, and RCS messages in the user's Android app and write SMS, MMS, and RCS messages; and

(8) Phone: The ability to initiate phone calls.

(56) For the functionalities in the previous paragraph, Alphabet shall ensure that third parties can use the functionalities as effectively as Alphabet's services, including in terms of scope of the data and in terms of supported use cases, such as semantic search and similar use cases that enable contextual interaction between the user and the service using natural language.

(57) Alphabet shall provide effective interoperability with any future updates, including new functionalities, of the structured on-device integration feature insofar as they are available to Alphabet's own services or hardware.

(58) Alphabet shall implement the measures above in compliance with the measures for all features in Section 5 of this Annex.

(59) Alphabet shall implement the measures for the structured on-device integration feature in the next major Android release, i.e. Android 18, and by 1 August 2027 at the latest.

### 3.2. Measures for screen automation

(60) Alphabet shall provide effective interoperability with the *screen automation* feature.

(61) The screen automation feature is described in Section 10.3.2 of the Decision. The feature enables Gemini to take control of other apps on the Android mobile device to automate multi-step tasks on behalf of the user. Gemini can imitate user behaviour and observe the controlled app's screen to automate the user interface and to perform most tasks a user would be able to do. Gemini controls the app in a separate virtual window, allowing the assistant to complete the task in the background, while the user can do something else.

(62) As described in Section 12.4.5 of the Decision, the screen automation feature, as implemented in Computer Control, meets the requirements of a restricted feature. Consequently, access to the functionalities referenced below in this section may be limited to Qualified Services.

(63) Alphabet shall implement an interoperability solution that provides third parties with access to the screen automation feature and its functionalities, as available to Alphabet's services and in a way that is equally effective as the solution available to Alphabet's services, such as Gemini.

(64) These functionalities are:

(a) **Taking control over apps.** Ability for an app (controlling app) to take over control of other apps (controlled apps) including those of third parties, the OEM, and Alphabet. The controlling app can perform interactions with the controlled apps by imitating user interactions, such as taps, clicks, and typing;

(b) **Accessing the on-screen content.** Ability for the controlling app to access the screen content of the controlled app to understand its user interface;

(c) **Discovering apps.** Ability for the controlling app to determine which apps are installed on the device and retrieve relevant information to decide which app to automate;

(d) **Controlling apps in background.** Ability for a controlling app to fully control of other apps in background. The process of controlling apps is not visible to the user. The controlled app will be run on a virtual display to enable background execution. The controlling app can complete the task while the user uses the phone otherwise or decides not to use their phone;

(e) **User observation.** Ability for the user to surface the controlled app and observe the controlling app interacting with it. In addition, the user can stop the controlling app from interacting with the controlled app;

(f) **Seamless transition.** Ability to transition seamlessly between background and foreground when the user chooses to observe the controlled app;

(g) **OS indicators.** Ability for OS indicators to demonstrate to the user a controlling app is currently interacting with another app;

(h) **Concurrency management.** Ability for Google Android to manage concurrent access, including scheduling or preventing multiple accesses. This can include user-facing mechanisms, such as greying out the app icon of the controlled app so that the user cannot use the same app at the same time as the controlling app; and

(i) **Blocking sensitive areas.** Ability for controlled apps to block sensitive views from access of the controlling app. This functionality allows apps to protect sensitive user data, critical settings, or purchase decisions from being automated by the controlling app.

(65) Alphabet shall grant third parties access to additional functionalities available to Alphabet's services if necessary to enable effective interoperability with the screen automation feature.

(66) To provide third parties with an interoperability solution for the screen automation feature that is equally effective as that available to Alphabet's services, such as Gemini, the following measures shall apply:

(a) Alphabet shall enable to discover all apps that are installed on the user's device. Alphabet shall ensure that third parties can retrieve relevant information to decide which app to automate;

(b) Alphabet shall enable multiple apps to have the capability to take control over other apps. Alphabet shall not limit this capability to one single app such as a default app; and

(c) Alphabet may enable any third party to restrict automation of their app for certain parts of the app, insofar as it applies across all services that use the screen automation feature.

(67) Alphabet shall provide effective interoperability with any future updates, including new functionalities, of the screen automation feature insofar as they are available to Alphabet's own services or hardware.

(68) Alphabet shall implement the measures above in compliance with the measures for all features in Section 5 of this Annex.

(69) Alphabet shall implement the measures for the screen automation feature in the next major Android release, i.e. Android 18, and by 1 August 2027 at the latest.

### 3.3. Measures for system integration

(70) Alphabet shall provide effective interoperability with the *system integration* feature.

(71) The system integration feature is described in Section 10.4.2 of the Decision. The feature enables Alphabet's services, such as Gemini, to interact and integrate with the operating system on behalf of the user. The user may request the assistant amongst other things to change a specific setting, control media playback, or open an app.

(72) As described in Section 12.4.5 of the Decision, certain functionalities of the system integration feature, if implemented via App Functions, would meet the requirements of a restricted feature. Consequently, insofar as the functionalities referenced below in this section are made available through this implementation, access to the functionalities may be limited to Qualified Services.

(73) Alphabet shall implement an interoperability solution that provides third parties with access to and interoperability with the system integration feature and its functionalities, as available to Alphabet's services and in a way that is equally effective as the solution available to Alphabet's services, such as Gemini.

(74) These functionalities are:

(a) **Ability to interact with Google Android OS settings.** These settings include turning on/off Bluetooth, turning on/off battery saver mode, turning on/off do not disturb, reading the current battery level;

(b) **Ability to interact with ongoing activities on the Google Android mobile device.** These activities include controlling playing media and volume;

(c) **Ability to interact with Google Android OS-level functionalities.** These functionalities include restarting the device, turning the device off, taking screenshots, and integrate with notifications; and

(d) **Ability to integrate with common functionalities of Google Android.** These functionalities include setting timers, managing alarms, integrate with text messaging functionalities.

(75) Alphabet shall grant third parties access to additional functionalities available to Alphabet's services if necessary to enable effective interoperability with the system integration feature.

(76) To provide third parties with an interoperability solution for the system integration feature that is equally effective as that available to Alphabet's services, such as Gemini, the following measures shall apply:

(a) Alphabet shall enable the discovery of all apps that are installed on the user's device and to understand user queries that demand to open specific apps;

(b) Alphabet shall make available the information necessary to open specific apps as are available to Alphabet services, such as Gemini; and

(c) Alphabet shall ensure that all system integrations are made available in the same manner as they are available to Alphabet's services, such as Gemini. In particular, the user should be able to execute the integrated functionalities

directly inside the third-party service, rather than be required to switch to the settings app or another app, to the extent this capability is available to Alphabet's services, such as Gemini.

(77) Alphabet shall provide effective interoperability with any future updates, including new functionalities, of the system integration feature insofar as they are available to Alphabet's own services or hardware.

(78) Alphabet shall implement the measures above in compliance with the measures for all features in Section 5 of this Annex.

(79) Alphabet shall implement the measures for the system integration feature in the next major Android release, i.e. Android 18, and by 1 August 2027 at the latest.

## 4. MEASURES ON FEATURES FOR ACCESS TO RESOURCES

### 4.1. Measures for system-level on-device models

(80) Alphabet shall provide effective interoperability with the *system-level on-device models (ODM)* feature.

(81) The system-level ODM feature is described in Section 11.2.2 of the Decision. The system-level ODM feature covers the Gemini Nano model family, including its variants and any successor models performing a similar role (including Gemma ODMs), and any other ODMs implemented via AICore or a successor or software performing a similar role to AICore on Google Android mobile devices ("system-level ODMs"). The feature allows Alphabet to discover, access, use and customise the system-level ODMs.

(82) Alphabet shall implement an interoperability solution that provides third parties with access to the system-level ODM feature and its functionalities, as available to Alphabet's services and in a way that is equally effective as the solution available to Alphabet's services.

(83) These functionalities are:

(a) **Discovery of system-level ODMs.** The ability to discover system-level ODMs;

(b) **Access to system-level ODMs.** The ability to access and use system-level ODMs and all their functionalities, including specialized APIs for specific tasks; and

(c) **Ability to customise functionality.** The ability to customise the system-level ODMs, its functionalities, and output.

(84) Alphabet shall grant third parties access to additional functionalities available to Alphabet's services if necessary to enable effective interoperability with the system-level ODM feature.

(85) To provide third parties with an interoperability solution for the system-level ODM feature that is equally effective as that available to any of Alphabet's services or hardware, the following measures shall apply:

(a) **Access subject to the same restrictions.** Alphabet shall allow third-party services or hardware to use the system-level ODMs without restrictions, except for those permissible under paragraph (86) and those that equally apply to Alphabet's services and hardware using the system-level ODM feature;

(b) **Ability to use functionality for all use cases.** Alphabet shall allow third-party services and hardware to use the system-level ODMs for all use cases, including those that Alphabet supports at the system level; and

(c) **Access conditions.** Alphabet shall ensure access to the system-level ODM feature for third-party services according to transparent, objective, precise, and non-discriminatory rules that also apply to Alphabet's services, including for use cases that Alphabet does not offer.

(86) Alphabet shall not be required to make accessible to third parties functionalities that are available only to internal operating system components.

(87) Alphabet shall provide effective interoperability with any future updates, including new functionalities, of the system-level ODM feature insofar as they are available to Alphabet's own services or hardware. For the avoidance of doubt, such future updates of the system-level ODM feature include any updates concerning the:

(a) The implementation of new system-level on-device models in AICore or a successor or similar Alphabet solution to AICore, such as those related to image or video creation; and

(b) Updates to existing system-level ODMs such as Gemini Nano ODMs on-device models.

(88) Alphabet shall implement the measures above in compliance with the measures for all features in Section 5 of this Annex.

(89) Alphabet shall implement the measures for the system-level ODM feature in the next major Android release, i.e. Android 18, and by 1 August 2027 at the latest.

### 4.2. Measures for on-device model implementation

(90) Alphabet shall provide effective interoperability with the *on-device model (ODM) implementation* feature.

(91) The ODM implementation feature is described in Section 11.3.2 of the Decision. The feature allows Alphabet to install, run and use AICore and ODMs implemented via AICore (including Gemini Nano and Gemma ODMs), as well as any future ODMs implemented via AICore or a successor or software performing a similar role to AICore on Google Android mobile devices ("AICore ODMs and related software"). The feature relies on access to hardware resources such as the CPU, GPU, NPU and RAM (including RAM residency), background execution capabilities, and the ability to make functionality of Alphabet's ODMs available to all or some apps running on the device to enable broad adoption.

(92) Alphabet shall implement an interoperability solution that provides third parties with access to the ODM implementation feature and its functionalities, as available to AICore ODMs and related software and in a way that is equally effective as the solution available to AICore ODMs and related software.

(93) These functionalities are:

(a) **Operation of on-device models.** The ability to install, run and use on-device models and software to implement such models;

(b) **Hardware resources.** The ability to timely and reliably access hardware resources to run on-device models or software to implement such models. This includes access to specialised chips to run on-device model tasks, such as the

TPU/APU/NPU, access to the Graphics Processing Unit (GPU), the CPU, and to RAM (incl. RAM residency) and other memory;

(c) **Centralised availability.** The ability for an app to centrally expose on-device models and software that implements such models to another app, and the ability for the other app to use such models and software. This includes the ability to exchange information between the service implementing the on-device model and the service making use of the on-device model; and

(d) **Background execution.** The ability to provide and use on-device models or software to implement such models in the background, i.e. when the app hosting the model and the app using the model are not in the foreground.

(94) Alphabet shall grant third parties access to additional functionalities available to AICore ODMs and related software if necessary to enable effective interoperability with the ODM implementation feature.

(95) To provide third parties with an interoperability solution for the on-device model implementation feature that is equally effective as that available to AICore ODMs and related software, Alphabet shall implement the following measures shall apply:

(a) Alphabet shall allocate hardware resources to AICore ODMs and related software and third-party on-device models and software to implement such models according to transparent, objective, precise, and non-discriminatory rules. Alphabet may do this by, for example, implementing an appropriate system-level scheduler that complies with these requirements. Such rules may distinguish between the use of hardware resources for critical system-level functionality and other functionality where appropriate;

(b) Alphabet shall apply any restrictions, time windows, and resource limitations (e.g. on CPU, GPU or NPU/APU/NPU execution) on background execution according to transparent, objective, precise, and non-discriminatory rules that also apply to AICore ODMs and related software, including for use cases that AI Core ODMs and related software do not support. Such rules may take into account aspects of system stability and device health, such as battery and thermal management. The rules may also take into account the need of system components – both Alphabet's and OEM's – to access background execution to preserve the internal functioning of the operating system, such as system-level hardware scheduling;

(c) Any limitation or choice on the background execution capabilities of, or access to hardware resources for third-party on-device models or software to implement such models as a result of a user action shall only be permissible if the user is allowed to take the same action with the same limiting effect regarding AICore ODMs and related software;

(d) If Alphabet allows OEMs to customise or otherwise change the level of access to hardware resources or background execution for AICore ODMs and related software, Alphabet shall ensure that such customisation complies with all the measures specified in this Annex. In particular, any such customisations or changes granted to AICore ODMs and related software must also be granted to third-party on-device models or software to implement such models; and

(e) Alphabet shall allow third-party on-device models and software to implement such models access to the same functionalities to install, run and use them as available to AICore ODMs and related software, with the exception of functionalities that are available only to internal operating system components.

(96) Alphabet shall provide effective interoperability with any future updates, including new functionalities, of the ODM implementation feature insofar as they are available to AICore ODMs and related software.

(97) Alphabet shall implement the measures above in compliance with the measures for all features in Section 5 of this Annex.

(98) Alphabet shall implement the measures for the ODM implementation feature in the next major Android release, i.e. Android 18, and by 1 August 2027 at the latest.

### 4.3. Measures for background execution

(99) Alphabet shall provide effective interoperability with the *background execution* feature.

(100) The background execution feature is described in Section 11.4.2 of the Decision. The feature allows to execute tasks in a timely manner, including when the relevant app is not in the foreground. This ability relies on the OS providing adequate interfaces to start and resume tasks, and allocating sufficient system resources (including CPU, GPU, NPU, and RAM) to the app or service.

(101) Alphabet shall implement an interoperability solution that provides third parties with access to the background execution feature and its functionalities, as available to Alphabet's services and in a way that is equally effective as the solution available to Alphabet's services.

(102) These functionalities are:

(a) **Access to hardware resources.** This consists in the ability to access hardware resources – including CPU, GPU, NPU, and RAM – to execute tasks;

(b) **Requesting resource allocation.** This consists in the ability to indicate to the OS that the service requests allocation of certain amount of resources, e.g. for a certain period of time or while a certain process is active;

(c) **Starting or resuming a task while the app is in the background.** This can depend on external conditions (e.g. an instruction received by another app; time-based conditions; a user approaching a certain location, such as their home; the user taking a picture) or internal conditions (e.g. the app concluding a task, such as the summarisation of a long audio recording). The task can be executed by the app itself (e.g. an AI assistant displaying a reminder notification at a given time) or by another app (e.g. an AI assistant sending a message via another messaging app at a given time); and

(d) **Device configurations.** This consists in the ability of an app or service to be treated preferably by the OS, directly or indirectly, in terms of access to resources because of configuration settings, including settings configurable by the end user.

(103) Alphabet shall grant third parties access to additional functionalities available to Alphabet's services if necessary to enable effective interoperability with the background execution feature.

(104) To provide third parties with an interoperability solution for the background execution feature that is equally effective as that available to Alphabet's services, such as Gemini, the following measures shall apply:

(a) Alphabet shall ensure that third parties have the same access to background execution as Alphabet's services, directly or indirectly. To this end, Alphabet

shall define transparent, objective, precise and non-discriminatory rules that also apply to Alphabet's services, including for use cases that Alphabet does not offer. To that end, Alphabet shall ensure that, in particular, the following conditions are met:

(1) The rules may take into account aspects of system stability and device health, such as battery and thermal management. The rules may also take into account the need of system components – both Alphabet's and OEM's – to access background execution to preserve the internal functioning of the operating system, such as system-level hardware scheduling; and

(2) The rules may not treat any specific service differently, for example by restricting or exempting a specific service, unless the user has provided consent for the specific service (e.g. by exempting the corresponding app from power restriction).

(b) Alphabet shall ensure that third parties can offer always-on hotword detection on connected devices to invoke an app or perform other operations on the Google Android mobile device, using the same access to background execution that is available to Alphabet's services for the same purpose. Alphabet shall publish developer documentation explaining how third parties can use the relevant Google Android APIs to implement this use case;

(c) If Alphabet allows OEMs to customise background execution rules, Alphabet shall ensure that such rules comply with all measures pertaining to the background execution feature specified in this Annex. In particular, such rules shall be transparent, objective, precise and non-discriminatory and equally apply to pre-installed services;

(d) Alphabet shall ensure that third parties may prompt the user to grant their services access to the same level of background execution that other services – including Alphabet's services – have access to. In particular, granting this access shall provide the app the same level of background execution that Alphabet requires OEMs to grant to its own services (also with respect to proprietary OEM restrictions) or similar requirements. The user shall be able to grant access with a single low-friction option, without having to navigate to or change multiple settings; and

(e) If Alphabet allows any limitation or choice on the background execution capabilities of user-downloaded apps as a result of a user action, Alphabet shall ensure that the user can take the same action with the same limiting effect regarding other apps, including pre-installed services. This includes the action of a user terminating an app in the app switching menu ("force-quitting").

(105) Alphabet shall provide effective interoperability with any future updates, including new functionalities, of the background execution feature insofar as they are available to Alphabet's services.

(106) Alphabet shall implement the measures above in compliance with the measures for all features in Section 5 of this Annex.

(107) Alphabet shall implement the measures for the background execution feature in the next major Android release, i.e. Android 18, and by 1 August 2027 at the latest.

## 5. MEASURES FOR ALL FEATURES

(108) The following measures shall apply to all features referred to in Sections 1 to 4 of this Annex.

### 5.1. Implementation across the Google Android ecosystem

(109) Alphabet shall take all necessary actions, including of technical, contractual, and legal nature, to ensure that all measures in this Annex are complied with on Google Android mobile devices,[3] including on devices supplied by third-party OEMs, provided the mobile devices meet the following conditions:

(a) The relevant feature exists on the device, and

(b) The device still receives Android AOSP, Google Play system, or Google system services updates.

(110) Alphabet shall implement all interoperability solutions in a way that avoids differences within the Android ecosystem, as Alphabet does for other Android SDK APIs. In particular, Alphabet shall ensure that third parties can use the interoperability solutions consistently via the same APIs across Google Android versions, devices, and form factors. This requirement applies regardless of how the solution is implemented on the specific Google Android mobile device or Android version. Alphabet may achieve this, for example, by defining Android SDK APIs and using Android support libraries, such as Android Jetpack, to define a consistent API across different Android versions and potentially different implementations.

(111) If Alphabet allows OEMs to apply any customisations that affect the features in Sections 1 to 4 of this Annex or their effective implementation, Alphabet shall take all actions necessary to ensure that the custom implementation complies with all the measures specified in this Annex.

### 5.2. User consent

(112) Alphabet shall not be prevented from implementing technical measures (e.g. system-level prompts) to: (i) ensure that features are accessible subject to user consent; (ii) inform users about feature access (such as dashboards or indicators); and (iii) allow the user to revoke access to the feature for any specific service.

(113) Any measures subjecting feature access to any requirements, such as user consent and the use of privacy indicators, as well as any corresponding technical measures shall be based on transparent, objective, precise, and non-discriminatory conditions that equally apply to Alphabet's services and hardware. They shall also comply with the other measures in this Annex, particularly the measures in Section 5.5 on equal effectiveness.

### 5.3. Integrity

(114) Alphabet may take strictly necessary and proportionate measures to ensure that interoperability does not compromise the integrity of the operating system, hardware and software features.

(115) Any integrity measure shall be duly justified and based on transparent, objective, precise, and non-discriminatory conditions that also apply to Alphabet's services and hardware. Alphabet can only impose conditions and take integrity measures that reflect a genuine integrity risk and do so in a consistent and systemic manner.

[3] Google Android mobile device in this Annex includes any mobile device that Alphabet certified as a certified Android device or a Google Play supported device.

Similarly, Alphabet shall only apply such conditions with which compliance is capable of being independently verified and not exclusively within the gatekeeper's control. An integrity measure cannot be considered strictly necessary and proportionate if it seeks to achieve a higher integrity standard than the one that Alphabet requires or accepts in relation to its own services or hardware. Alphabet shall not impose a higher level of integrity on third parties than it applies to itself.

(116) Any integrity measure shall be implemented in a way that does not undermine effective compliance with these measures and Alphabet's obligations with Regulation 2022/1925 including for instance by subverting end users' or third parties' autonomy, decision-making, or free choice via the structure, design, function or manner of operation of a user interface or apart thereof.

(117) Alphabet shall ensure that the following conditions apply to any integrity measures:

(a) The measures must be consistent with the approach that Alphabet already applies in similar or comparable cases;

(b) The measures shall apply equally to Alphabet's services and to third parties;

(c) The measures shall be based on objective and verifiable evidence showing the existence and magnitude of the integrity risk and effectiveness in reducing such risk. Alphabet shall retain such evidence;

(d) The measures shall not impose limits on the purpose, beneficiaries, apps, technologies used or use cases for feature access, unless specified otherwise in this Annex;

(e) The measures shall not impose any commercial or financial requirements on the beneficiaries;

(f) The measures shall not impose functional requirements beyond those strictly necessary to preserve integrity. In particular, they shall not assume or require that a certain technology is used, unless specified otherwise in this Annex;

(g) The measures shall be enacted only for as long as is necessary, and be adapted to technological evolution, including changes to the operating system; and

(h) The measures shall be grounded in established industry standards and practices.

(118) Alphabet shall inform the Commission in writing of any integrity measure that Alphabet intends to take in relation to the features covered by this Annex, providing a justification of their strict necessity and proportionality at least 4 weeks in advance of their implementation, or without undue delay in case of urgency, unless the measure meets each of the following cumulative conditions:

(a) it is not user-facing,

(b) it is exclusively of a technical nature,

(c) it is implemented for Alphabet and third parties in precisely the same way, and

(d) Alphabet has determined that the change will have no or only insignificant impact of any nature on third parties, including technical or commercial impact. Alphabet shall retain written documentation on how any such determination was made.

### 5.4. Eligibility of beneficiaries, applications and use cases

(119) Unless a feature constitutes a Restricted feature within the meaning of paragraph (120) below, Alphabet shall comply with the following measures:

(a) Alphabet shall make the features listed in Sections 1 to 4 of this Annex, and interoperability solutions to implement these features pursuant to the measures in this Annex, accessible to all third parties, including user-installed apps, pre-installed apps without system privileges, and apps of original equipment manufacturers (OEMs).

(b) Alphabet shall make available the interoperability solutions and measures implemented in compliance with these measures to all third parties without undue delay, to the extent they indicate, including through the use of APIs, an interest in making use of any or all of the features listed in Sections 1 to 4 of this Annex, including for use cases that Alphabet does not offer.

(c) Alphabet shall not impose any restrictions on the type or use case of the software application that can access or make use of the features listed in Sections 1 to 4 of this Annex.

*5.4.1. Restricted features*

(120) Prior to enabling access to restricted features (**"Restricted feature"**), Alphabet may require the developer to demonstrate that its service meets certain eligibility conditions. The Restricted features are:

(a) Centralised access to apps' data stored on-device;

(b) Context-aware intelligence;

(c) Structured on-device integration;

(d) Screen automation; and

(e) System integration.

If doing so, Alphabet shall comply with the measures set out below in this section.

(121) **Request to expand the list of Restricted features.** The Commission may, in response to a reasoned request from Alphabet showing good cause, include another feature (or part thereof) among the ones in scope of this Decision to the group of Restricted features.

(122) Alphabet shall ensure that the eligibility conditions to access Restricted features apply to and treat equally all services, including Alphabet's services, and third-party services.

(123) **Qualified AI Assistant Programme.** Alphabet shall create a Qualified AI Assistant Programme to certify AI assistants as meeting the eligibility conditions (**"Qualified AI assistants"**) to access Restricted features. Alphabet shall grant Qualified AI assistants access to all Restricted features listed in paragraph (120).

(124) Alphabet shall define the terms that govern the Qualified AI Assistant Programme – which shall consist of (i) eligibility conditions subject to the requirements set out in paragraphs (125)-(126) and (ii) processes subject to the requirements set out in paragraphs (127)-(135) below.

(125) **Eligibility conditions to be certified as Qualified AI assistant.** Alphabet shall define a single set of transparent, objective, precise and non-discriminatory eligibility conditions that apply to AI assistants (including Alphabet's AI assistants) and are

sufficient for them to be certified as Qualified AI assistants. The eligibility conditions for AI assistants may only include the following requirements:

(a) **functional eligibility conditions**, requiring the candidate AI assistant to have (i) advanced reasoning capabilities, such as through the use of Large Language Models ("LLMs"), that enable it to interpret and act on user intent autonomously; (ii) capability to seek information, nuanced understanding and the reliable execution of complex, multi-step instructions; and (iii) capability to perform a range of tasks including tasks relating to other apps installed on the device;

(b) **user intent conditions**, requiring that the candidate AI assistant's agentic capabilities (i) honour user intent; (ii) are only triggered with the user's intent; and (iii) must reconfirm user intent before executing sensitive, irreversible actions;

(c) **agent and model functional risks conditions**, requiring that the candidate AI assistant, and the advanced reasoning used, are adequately hardened against key agentic risks that would negate user intent, such as input, supply chain, integration, model integrity, and infrastructure risks;

(d) **agent privacy, safety, and transparency conditions**, requiring that the candidate AI assistant (i) accesses and processes data only to the extent necessary for the user's intent, (ii) minimises inadvertent data disclosure, and (iii) enables informed user consent including through user transparency and control mechanism;

(e) **agent app security conditions**, requiring that the candidate AI assistant meets baseline mobile application security requirements, showing that the agent app guards against key vulnerabilities; and

(f) **developer reputability conditions**, requiring that the developer of the candidate AI assistant (i) implements relevant technical and organisational measures, policies, and processes to preserve agent and model safety, privacy, transparency, and security, (ii) remains abreast of emerging AI-based vulnerabilities, and (iii) implements policies and processes to monitor and assess the resilience of its AI assistant in light of these developments.

(126) Alphabet may further specify the eligibility conditions, including by specifying feature-specific conditions, but shall not introduce additional conditions without consulting the Commission. Alphabet shall ensure that the feature-specific conditions are strictly necessary and proportionate to the specific feature. Alphabet shall not require the AI assistant to support all Restricted features to be certified as Qualified AI assistant but may subject interoperability with a Restricted feature to the feature-specific conditions.

(127) **Trusted Certification Authorities Programme.** Alphabet shall create a Trusted Certification Authorities (**"TCA"**) Programme and define the terms of the programme, which shall include: (i) the eligibility criteria and (ii) the process for a third party to be approved or revoked as TCA by Alphabet, free of charge. The terms of the programme must be reasonable, non-discriminatory, and in line with existing schemes for certification authorities.

(128) **Obtaining certification from Trusted Certification Authorities.** Alphabet shall entrust the selected TCAs to certify services as Qualified AI Assistants, i.e. to approve them as beneficiaries of the interoperability measures set out in this Annex. To this end, Alphabet shall ensure that AI assistants meeting the eligibility conditions

can obtain certification as Qualified AI Assistants from TCAs. Alphabet shall accept certifications issued by TCAs without imposing further requirements. Alphabet shall not revoke certifications issued by TCAs.

(129) **Obtaining certification from Alphabet.** In addition to the Trusted Certification Authorities Programme, Alphabet shall allow third-party services to be certified as Qualified AI assistants by Alphabet, free of charge.

(130) **Revocation.** If Alphabet intends to revoke a certification as a Qualified AI Assistant issued by Alphabet, Alphabet shall inform the provider of the certified Qualified AI assistant sufficiently in advance to allow the provider to address Alphabet's concerns, and in any case at least one month before the revocation. Alphabet shall communicate to the provider a complete description of the reasons for the intended revocation and the steps that the provider should take to avoid the revocation.

(131) **Unilateral suspension by Alphabet.** Alphabet may temporarily suspend a Qualified AI assistant from accessing Restricted features on end users' devices, subject to Alphabet being in possession of a consistent body of evidence that a Qualified AI assistant has been engaging in practices that can cause severe and immediate harm to users. When applying such a suspension, Alphabet shall take the following steps:

(a) No later than at the time of the suspension, Alphabet shall inform the provider of the Qualified AI assistant and indicate the steps that the third party has to take for Alphabet to revert the suspension or to avoid the suspension if it has not yet entered into effect. Alphabet shall provide the third party with a direct and continuous channel of communication for a prompt resolution of the issue; and

(b) Alphabet shall inform the Commission and the relevant TCAs without undue delay.

(132) When informing the third party, the TCAs, and the Commission pursuant to the previous paragraph, Alphabet shall transfer the body of evidence that led to the suspension for reasons of severe and immediate harm to users.

(133) Alphabet shall reinstate the Qualified AI Assistant's access to Restricted features as soon as the grounds justifying the suspension are no longer present.

(134) **Appeal.** Alphabet shall provide third parties with a mechanism to appeal the following decisions by Alphabet:

(a) A decision by Alphabet not to approve the third party as a TCA;

(b) A decision by Alphabet to revoke the approval of the third party as a TCA;

(c) A decision by Alphabet not to certify a service as Qualified AI assistant;

(d) A decision by Alphabet to revoke a certification as Qualified AI assistant issued by Alphabet; and

(e) A decision by Alphabet to unilaterally suspend a Qualified AI assistant's access to Restricted feature, per paragraph (131) above.

Alphabet shall issue a decision on the appeal within one month.

(135) Alphabet shall allow end users to give informed consent to opt out of the certification requirement for their device, on a per-service basis, in line with existing opt-out mechanisms for advanced users. Alphabet may put in place measures that support transparency to the user before granting consent, such as information screens and short delays. Alphabet shall not require the end user to enable developer mode or

other options that can interfere with the functioning of the Restricted features or any other feature.

(136) **Timeline**.

(a) By 1 February 2027, Alphabet shall publish the draft terms of the Qualified AI Assistants Programme and of the Trusted Certification Authorities Programme for consultation by third parties and by the Commission. In the final terms, Alphabet shall take utmost account of the comments received;

(b) By 1 May 2027, Alphabet shall publish the final terms;

(c) Alphabet shall accept applications for certification as a Qualified AI assistant as of 1 May 2027 and shall conclude the assessment within four weeks upon receipt of the application, except where delays are outside of Alphabet's control. If Alphabet received an unusually high number of requests within a single time period, Alphabet shall conclude the assessment within 8 weeks; and

(d) Alphabet shall accept applications for certification as a Trusted Certification Authority as of 1 February 2027 and shall conclude the assessment within four weeks upon receipt of the application.

(137) **Qualified services that are not AI assistants.** Alphabet shall ensure that services (**"Qualified Service"**) that are not Qualified AI assistants can access Restricted features under transparent, objective, precise, non-discriminatory eligibility conditions that apply also to Alphabet's own services. Alphabet shall not impose any restrictions on the type or use case of the software application that can access Restricted features under such conditions. Alphabet shall ensure that the eligibility conditions are strictly necessary and proportionate to the specific feature and the specific type of service. Alphabet shall apply the same processes set out in (127)-(135) above. Alphabet shall allow the same Trusted Certification Authorities to certify both AI assistants and services that are not AI assistants.

(138) **Updates to the terms.** Alphabet shall keep the programmes specified in this section under regular review to ensure that its criteria and their application are suitable to enable access to future generations of services – including AI Assistants – and are in line with developments in the relevant technology. If Alphabet plans to define or update the terms for Qualified AI assistants or any other Qualified Service, or if Alphabet plans to update the terms of the Trusted Certification Authority Programme, Alphabet shall consult the Commission at least 2 months prior to updating the terms.

*5.4.2. Other restrictions and requirements*

(139) Alphabet shall not impose any undue restrictions, including by requiring third parties to use other Alphabet products or services unless required for the functioning of the feature, or by requesting third parties to make choices in situations where such choice is not justified (for example, choosing between using the interoperability solution and continuing to use the same application ID), requiring third parties to use other Alphabet products or services in order to access and use the feature unless required for the functioning of the feature, or preventing third parties from benefitting from access to other features, including using the feature in combination with other features within the scope of Article 6(7) of Regulation 2022/1925.

(140) Alphabet shall not undermine effective interoperability with the features listed in Sections 1 to 4 of this Annex by behaviour of a technical, commercial, contractual or any other nature. In particular, Alphabet shall:

(a) enable third parties to make use of the interoperability solution in their existing apps via an automatic update of such apps, without being in any way disadvantaged for doing so, such as by having to reacquire or migrate existing users to a separate app; and

(b) not degrade, remove, disable, or otherwise make ineffective the interoperability solution, or prevent or impede updates, including security updates, for the end user as long as the end user is eligible to benefit from the functionalities allowed by these interoperability solutions; before taking any such measure in this respect, Alphabet shall notify the end user explaining how the measures Alphabet intends to take will affect the interoperability solution relied on by third parties on the end user's Google Android mobile device.

**5.5. Equal effectiveness**

(141) Alphabet shall ensure that any interoperability solution implemented for the measures listed in Sections 1 to 4 of this Annex is equally effective to the solution available to Alphabet's services and hardware. Alphabet shall apply such equal effectiveness across all dimensions, including, but not limited to, the end user journey, ease of use for end users, device and software setup, data transmission speed, and energy consumption.

(142) Alphabet shall make all interoperability solutions accessible in a manner that is technically sound, stable, and workable in practice for third parties without unnecessary hurdles.

(143) Alphabet shall design interoperability solutions so that all apps can use features in a concurrent and a non-discriminatory way. Alphabet shall not subject access to features to the app holding a default role, including the default assistant role.

(144) To ensure access is equally effective to the solution available to Alphabet's own services and hardware in terms of end user journey, Alphabet shall not decrease the ease, convenience and speed of using third-party services and hardware from the end user perspective. In particular, Alphabet shall refrain from adding friction by:

(a) Offering choices to, or requesting permission from, the end user in a non-neutral or leading manner, including by using design patterns, dark patterns, or misrepresenting or exaggerating any risks of using the third-party service or granting a permission;

(b) Preventing the third party from explaining to end users in their own language the relevance of any system prompts shown, immediately before the prompt is shown or within the prompt;

(c) Showing unnecessary recurring prompts or notifications that the end user cannot easily and permanently disable in the same prompt or notification;

(d) Preventing the third party from triggering a permission prompt again in the future, unless the end user has so decided;

(e) Preventing third parties from obtaining user consent for any required permissions (i) from inside the application itself or with a simple system prompt or (ii) by indicating the user to follow a link directly pointing to and highlighting the relevant item in the system settings (so-called "deep-linking"); and

(f) Requiring end users to process multiple successive permission prompts that could be presented in a single prompt.

(145) In particular, Alphabet shall apply the same end user journey and ease of use for end users in order to obtain the necessary consent for its own and third-party services and hardware, including:

(a) showing the same user consent prompts (in terms of, inter alia, number, content, format and design) as Alphabet;

(b) showing the same information screens (in terms of, inter alia, number, content, format and design) as shown to users by Alphabet; and

(c) limit the necessary user engagement to the same level as adopted by Alphabet, including the number of prompts and information screens.

(146) Alphabet shall base any quantitative limitations (e.g., quotas and rate limits) on third-party access to a feature on transparent, objective, precise, and non-discriminatory conditions that also apply to Alphabet's services and hardware, including for use cases that Alphabet does not offer.

(147) Alphabet shall enable third parties to centrally expose their functionalities to other apps and shall enable third parties to access such functionalities – in a way that is equally effective to what is available to Alphabet's services and hardware.[4] Alphabet shall also enable third parties to dynamically provide and retrieve assets – including but not limited to executable modules – at runtime, including from other apps, in a way that is equally effective to what is available to Alphabet's services and hardware.[5] If the app providing the assets is not installed when another app requests those assets, Alphabet shall enable the user to install the app via a low-friction in-app mechanism, in a way that is equally effective to what is available to Alphabet's services and hardware.[6]

### 5.6. Free of charge

(148) Alphabet shall provide the interoperability solutions and measures implemented to allow for effective interoperability with the features listed in this Annex free of charge, irrespective of their beneficiary, app, service, product, and use case. Alphabet shall also not charge any fees indirectly for any of the measures set out in this Annex.

### 5.7. Documentation and APIs

(149) Alphabet shall make available complete documentation of all interoperability solutions it will be making available in the context of the implementation of the measures set out in this Annex. This includes Alphabet making available complete, accurate, and well-documented APIs to the extent that access to such APIs is relevant for the implementation of the measures set out in this Annex.

(150) Alphabet shall maintain this documentation and these APIs in line with Alphabet's internal processes and policies, so that effective access for the purpose of ensuring interoperability is guaranteed over time.

### 5.8. Assistance and testing

(151) Alphabet shall provide reasonable technical assistance, free of charge, to third parties to implement and achieve effective interoperability with the features in this Annex.

---

4 For example, Google Play Services APIs are exposed centrally to other apps.

5 For example, Gemini Nano is downloaded via AICore when the user takes an action that triggers the download, e.g. by using a feature in Alphabet's apps that requires Gemini Nano.

6 For example, in Google Play, Alphabet implements the Play Auto Install and Google Play Inline Installs mechanisms.

(152) Alphabet shall ensure that all interoperability solutions implemented to address the measures specified in this Annex are subject to Alphabet's usual practices, including beta testing.

(153) Alphabet shall ensure that any engagement with, and communication to, third parties for the purpose of implementing the measures specified in this Annex is carried out effectively, promptly, and without undue delay.

(154) Alphabet shall not impose any undue restrictions on the ability for third parties to develop and test services that use the interoperability solutions implemented to address the measures specified in this Annex. This includes not preventing the ability to develop and test services that access Restricted features without having been certified yet as a Qualified Service.

**5.9. Future updates and new functionalities**

(155) Should Alphabet make changes to a feature listed in in this Annex, including adding new functionalities to the feature or making updates, Alphabet shall:

(a) Develop such new functionalities or updated features in a way that they are interoperable with third-party services or hardware;

(b) Include the interoperability solutions at an appropriate time in the beta version of the new functionalities or updated features; and

(c) Make accessible the updated interoperability solution on a Google Android mobile device and documentation for the relevant feature no later than at the time the new functionality or updated feature becomes accessible to any Alphabet service or hardware on the same Google Android mobile device.

(156) Alphabet shall maintain the interoperability solution over time such that the solution and its documentation continue being available, functional, usable, and effective for all third parties without interruption. If, in exceptional circumstances, Alphabet wishes to deprecate an interoperability solution or parts of it, Alphabet shall submit a reasoned request in accordance with the procedure described in paragraph (165) of this Annex.

**5.10. Reporting**

(157) Alphabet shall communicate to the Commission shortly after the notification of the Decision pursuant to Article 8(2) of Regulation (EU) 2022/1925 in the present proceedings, and in any event within two months of the date of notification of such Decision, all measures that it intends to take to comply with this Annex and the Decision in sufficient detail to enable the Commission to assess whether the measures are *prima facie* suitable to address the requirements of the Decision ("Initial Implementation Report"). The Initial Implementation Report shall, for each specified measure, include, in particular, the following information:

(a) a detailed description of the interoperability solutions that Alphabet intends to make available;

(b) a description of how these solutions address all of the measures in this Annex and how it will provide third parties with effective interoperability under equal conditions to those available to Alphabet's services;

(c) if applicable, a description of any integrity measures that Alphabet intends to apply; and

(d) a detailed timeline for the design, development, implementation and release of each of the effective interoperability solutions.

(158) Following the Initial Implementation Report and until the expiry of the implementation deadlines for all features, Alphabet shall provide, on a monthly basis, a report ("Monthly Implementation Report") providing, in detail, for each specified measure, the following information:

(a) the envisaged interoperability solutions that Alphabet intends to make available to address each of the specified measures;

(b) status on the design, development, implementation and release of each of the interoperability solutions, including whether progress is on schedule and, if not, why not;

(c) if necessary, an updated detailed timeline for the design, development, implementation and release of each of the interoperability solutions, including, for instance, beta testing;

(d) any potential issue that Alphabet may have encountered during the design, development, implementation or release of each of the envisaged interoperability solutions;

(e) a description (including, where necessary, the relevant supporting documents) of Alphabet's engagement with third parties for the purposes, and in the context of, the design, development, implementation and release of each of the interoperability solutions. This shall include:

(1) a description of the issues raised by third parties that relate to the features and an assessment of whether such issues will require a modification of the interoperability solution, including any relevant rules (e.g. rules on background execution);[7]

(2) any information and feedback provided to Alphabet by third parties regarding the betas of Alphabet's interoperability solutions; and

(3) all interoperability requests made by third parties with regards to all features referred to in Sections 1 to 4 of this Annex.

(f) any relevant changes that Alphabet has made or intends to make to policies, requirements and agreements, including the CDD, the GMS requirements, placement agreements, etc.; and

(g) a description of how OEMs will apply or have applied customisations, if any, to the interoperability solutions, including any relevant rules (e.g. rules on background execution).

(159) At least 10 working days prior to any public release in relation to a feature or functionality within the scope of these Proceedings, be it beta testing or the final solution, Alphabet shall communicate its plans to the Commission ("Pre-release Feature Implementation Report").

(160) Upon expiry of the implementation deadline for each feature, Alphabet shall provide a report ("Final Feature Implementation Report") to the Commission explaining all the measures that it has taken to comply with this Annex and with the Decision and confirming the date of the public release. Under this obligation, Alphabet shall

[7] Modifications of the rules may also include changes to existing APIs and the introduction of new APIs that relate to background execution.

describe the interoperability solution made available to third parties, including all technical details and potential APIs, as well as any potential integrity measures.

(161) For two years after expiry of the implementation deadlines for all features, Alphabet shall provide to the Commission, every six months, a report ("Semi-annual Implementation Report") containing the same information as the Monthly Implementation Report in relation to any change that Alphabet has made or intends to make to the interoperability solutions, OEM customisation, and engagement with third parties.

(162) Alphabet shall include in particular in each of the five types of reports (Initial Implementation Report, Monthly Implementation Report, Pre-release Feature Implementation Report, Semi-annual Implementation Report, and Final Feature Implementation Report) any changes it plans to make or has made, respectively, in relation to any feature listed in the Decision, including any new functionality it plans to add or has added to the feature or any updates it plans to make or has made to such feature or functionalities thereof.

(163) Alphabet shall provide the Commission with a confidential and a non-confidential version of the Final Feature Implementation Report when each report is due. Upon the Commission's request, Alphabet shall provide the Commission with a non-confidential version of any of the other reports (Initial Implementation Report, Monthly Implementation Report, Pre-release Feature Implementation Report, and Semi-annual Implementation Report).

(164) Alphabet shall cooperate with the Commission to test specific technical solutions and report on the results of the tests with sufficient frequency and granularity to allow the Commission to properly assess the specific technical solution and the test results.

### 5.11. Waiver

(165) The Commission may, in response to a reasoned request from Alphabet showing good cause, modify or substitute one or more of the measures or a part of them. The request shall not have the effect of suspending the implementation of the measures and, in particular, of suspending the expiry of any time period in which the measure has to be complied with.