ETSI TS 103 426 V1.1.1 (2016-11)

Publicly Available Specification (PAS);
Smart Machine-to-Machine communications (SmartM2M)
Home Gateway Initiative
RD048-HG Requirements For HGI Open Platform 2.1

CAUTION

The present document has been submitted to ETSI as a PAS produced by HGI and approved by the ETSI Technical Committee Smart Machine-to-Machine communications (SmartM2M).

Reference
DTS/SMARTM2M-103426

Keywords
GATEWAY, Home Gateway, intelligent homes & buildings

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

List of figures

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Smart Machine-to-Machine communications (SmartM2M), as result of the PAS process for document HGI-RD048 developed by the Home Gateway Initiative.

The Home Gateway Initiative, a non-profit organization closed on June 2016, produced guidelines, requirements documents, white papers, vision papers, test plans and other documents concerning broadband equipment and services which are deployed in the home.

HGI worked on Specifications for home connectivity and Services enablement, in particular to encompass a delivery framework for Smart Home services. The defined architecture includes support for a standard, general purpose software execution environment in the HG (for third party applications), API definitions, device abstraction, and interfacing with Cloud based platforms.

The HGI's methodology ensured that projects undertaken reflected items of strong interest to the Broadband Service Providers (BSPs), as well as brought in opportunities at every stage for vendor input, suggestions and participation.

NOTE: Silicon Labs®, Java™, ProSyst®, Makewave® are suitable products available commercially. This information is given for the convenience of users of the present document and does not constitute an endorsement by ETSI of this these product).

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

1 Scope and purpose of the present document

1.1 Abstract

The present document, originally developed as HGI-RD048v2, contains requirements regarding modular software deployments on the home gateway. These requirements form the HGI's "Open Platform 2.1".

Under the Open Platform 2.1 requirements, modular software applications have to run in a dedicated virtual execution environment to avoid conflicts and interferences with the natively installed software. The present document reflects generic requirements valid for any modular execution platform, as well as technology-specific requirements for OSGi technology. Other technology-specific requirements could be developed in the same way as the OSGi requirements, in conjunction with the generic requirements.

The present document, HGI_RD048v2, is an update of, and supersedes, HGI's already-published HGI-RD48v1 which specified the requirements for Open Platform 2.0. HGI-RD048v2 specifies Open Platform 2.1 that primarily updates external references and makes various corrections.

HGI-RD48v1 itself was an updated version of HGI-RD008 [i.20], HG Requirements for a Software Execution Environment, which was known in the HGI community as "SWEX". Besides updates on referred standards, HGI-RD048v1 added basic requirements to support USB based hardware extendibility for Smart Home services, details for usage for OSGi technology and system clock management.

1.2 Scope and purposes

Service delivery to residential customers beyond triple play requires the integration of home devices and appliances with cloud infrastructures. Such integration often requires new software in the home network, but the variety of available technologies makes this difficult. In order to achieve the next level of service integration, there is a need for software flexibility on the main operator-controlled device in the home, the home gateway.

New software can be added to the HG by doing complete firmware upgrades; however doing this may cause significant problems. Each new version need to be fully tested, and different versions are required for different application areas. Maintaining different versions of firmware for several HG models would further complicate configuration management. There is also the considerable overhead of upgrading large numbers of HGs.

The solution discussed in the present document integrates a software execution platform, called by HGI Open Platform 2.1, into the firmware, allowing the installing, updating, uninstalling, starting and stopping of additional software modules, while the underlying firmware image remains untouched.

The present document contains a home gateway software modularity architecture specification based on function blocks, a role and entity model, and derives requirements for the home gateway. Requirements are specified not only for a software execution platform, but also for an API that allows software modules to access the core home gateway functions.

The requirements section is divided into two areas: generic requirements, which apply to any technology used as software execution platform, and specific requirements for selected technologies. Specific requirements are only given for OSGi technology in the present document.

1.3 What is new compared to HGI-RD008 and to RD-048v1?

1.3.1 RD-048v1 compared with RD-008 [i.20]

- Addition of Smarthome related low-level requirements, especially in terms of support of USB based hardware extendibility.
- Detailed requirements for system clock synchronization.
- Detailed OSGi requirements: Many requirements have been detailed for clarification and precision reasons.
- Reference to Broadband Forum TR-181 [i.4] rather than the outdated TR-098 [i.5].

- Reference to OSGi R4 version 4.2 ([i.9] and [i.10]) and higher rather than 4.0 and higher.
- Collation of Java™ related requirements into HGI/Minimum-1.0 Java™ Profile.

1.3.2 RD-048v2 compared with RD-048v1

- Streamlined text.
- Modified figures to reflect text consistency.
- Streamlined wording of requirements.
- Updated references to OSGi and Java™ JRE version.
- Updated Java™ related requirements to Java™ 8 compact1 profile.
- Requirements that in RD-048v1 referred to OSGi Service Platform Specification Release 4.2 (Core [i.9] and Compendium [i.10]) now refer to Release 4.3 (Core [i.11], and Residential [i.12]), 5 (Core [i.13]) and 6 (Core [i.14], and Residential [i.15]).
- Re-inserted references to TR-098 [i.5] along with TR-181 [i.4] since TR-098 continues to be deployed.

1.4 Relationship with HGI residential profile

The present document contains requirements over and above those in the HGI Residential Profile [i.1], to support the software module execution environment. All requirements of the HGI Residential Profile still apply.

Some of the original functionalities defined in the HGI Residential Profile [i.1] may be suitable for implementation as software modules, in particular those that have some of the following characteristics:

- Control functions (rather than data plane).
- Functions which can be run on different HG models.
- Functions with different versions.

The following list gives examples of functionalities from the HGI Residential Profile, which may be amenable to implementation as software modules (see definitions in [i.1]):

DHCP Server - A DHCP server is a fairly standalone application, handing out local IP addresses and managing their lifetime. The information gathered by the DHCP server can easily be given to other (authorized) modules (MAC addresses, DHCP options).

UPnP IGD - A UPnP IGD [i.2] service for example would be easy to implement on top of an OSGi service platform, because OSGi provides a standardized UPnP stack [i.2], given that there is an interface to manage the lower level functions of the HG.

Local Management Remote User Interface - An operator might see some advantage in having the same implementation of the LM Remote UI for different models of HGs, to facilitate a consistent corporate design, and having only one code base to maintain.

DynDNS Client - A DynDNS client is also a fairly stand-alone application (registering a public IP address at a certain domain server). Having it as a module has the advantage that several registration protocols for different dynamic DNS services can be supported.

NTP Client - Needs access to system date and time management of the HG.

However, some functions defined in the HGI Residential Profile are not recommended to be done as modules. These include the networking features (e.g. routing, bridging, NAT), the basic HG remote management and the firmware upgrade from the RMS (Remote Management System). The common feature of these is that these functions are normally preserved in the case of an execution environment fault. Further, these functions are generally already available in native software, and gain a performance advantage from direct interaction with the hardware.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at https://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

Not applicable.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] HGI-RD001-R2.01: "Home Gateway Technical Requirements: Residential Profile V1.01".

[i.2] UPnP IGD 2.0: "Internet Gateway Device (IGD) v2.0".

NOTE: See http://upnp.org/specs/gw/igd2/.

[i.3] ETSI TS 103 424: "Publicly Available Specification (PAS); Smart Machine-to-Machine communications (SmartM2M) Home Gateway Initiative RD036-Smart Home architecture and system requirements".

[i.4] BBF TR-181: "Device Data Model for TR-069".

NOTE: See https://www.broadband-forum.org/technical/trlist.php?key=1.
At the time of writing two *issues* of TR-181 are available. *Issues* in BBF terminology stand for non-backward-compatible updates, so in fact TR-181i1 and TR-181i2 describe independent data models and requirements are intended to specify compliance at least with one *issue*. *Amendments* indicate backward compatible changes (except for DEPRECATED and OBSOLETED features) so it would be enough to check compliance with the most recent *Amendment* for each issue. *Corrigenda* indicate minor revisions.

[i.5] BBF TR-098: "Internet Gateway Device Data Model for TR-069".

NOTE: See https://www.broadband-forum.org/technical/trlist.php?key=1.

[i.6] BBF TR-069 Amendment 5: "CPE WAN Management Protocol".

NOTE: See https://www.broadband-forum.org/technical/trlist.php?key=1.

[i.7] BBF TR-104: "DSLHome Provisioning Parameters for VoIP CPE".

NOTE: See https://www.broadband-forum.org/technical/trlist.php?key=1.
At the time of writing two *issues* of TR-104 are available. *Issues* in BBF terminology stand for non-backward-compatible updates, so in fact TR-104i1 and TR-104i2 describe independent data models and requirements are intended to specify compliance at least with one *issue*. *Amendments* indicate backward compatible changes (except for DEPRECATED and OBSOLETED features) so it would be enough to check compliance with the most recent *Amendment* for each issue. *Corrigenda* indicate minor revisions.

[i.8] BBF TR-157 Amendment 10: "Component Objects for CWMP".

NOTE: See https://www.broadband-forum.org/technical/trlist.php?key=1.

[i.9] "OSGi Service Platform Core Specification Release 4.2".

NOTE: See https://osgi.org/download/r4v42/r4.core.pdf.

[i.10] "OSGi Service Platform Compendium Specification Release 4.2".

NOTE: See https://osgi.org/download/r4v42/r4.cmpn.pdf.

[i.11] "OSGi Service Platform Core Specification Release 4.3".

NOTE: See https://osgi.org/download/r4v43/osgi.core-4.3.0.pdf.

[i.12] "OSGi Service Platform Residential Specification Release 4.3".

NOTE: See https://osgi.org/download/r4v43/osgi.residential-4.3.0.pdf.

[i.13] "OSGi Service Platform Core Specification Release 5".

NOTE: See https://osgi.org/download/r5/osgi.core-5.0.0.pdf.

[i.14] "OSGi Service Platform Core Specification Release 6".

NOTE: See https://osgi.org/download/r6/osgi.core-6.0.0.pdf.

[i.15] "OSGi Service Platform Residential Specification Release 6".

NOTE: See https://osgi.org/download/r6/osgi.residential-6.0.0.pdf.

[i.16] "Java 8 For Embedded, compact1 profile".

NOTE: See https://docs.oracle.com/javase/8/docs/technotes/guides/compactprofiles/compactprofiles.html.

[i.17] JSR 80: "Java™ USB API".

NOTE: See https://jcp.org/ja/jsr/detail?id=80.

[i.18] JSR 180: "SIP API for J2ME™".

NOTE: See https://jcp.org/ja/jsr/detail?id=180.

[i.19] "Java SE Embedded 8 vs. Java ME CDC Comparison".

NOTE: See http://www.oracle.com/technetwork/java/embedded/resources/tech/java-se-emb-vs-java-me-cdc-2157146.html.

[i.20] HGI-RD008: " Requirements for Software Modularity on the Home Gateway".

NOTE: See http://www.homegatewayinitiative.org/userfiles/file/downloads/RD-008-R3.pdf.

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

Gateway Operator (or just operator): Primary responsibility of the Gateway Operator is to control who is allowed to deploy services to the Service Platform in question i.e. control which Service Deployment Managers are allowed to manage the particular Service Platform. In addition to this, the Gateway Operator can also manage other functions related to a specific Service Platform instance.

Home Gateway (HG): hosts one or more Service Platforms

(Home Gateway) Service Platform (HG_SP): Primary function of the Home Gateway Service Platform is to manage the execution lifecycle of Service Modules. The Service Platform is capable of dynamically loading, activating, deactivating, updating, and unloading the Service Modules. The HG_SP is part of the HG_EE as defined in clause 4.2.

Network Provider: Provides and manages wide area network connectivity between the Service Platform and other parties, which include the Gateway Operator and other Service Providers. In the case where the Service Platform is connected via the Internet, the Network Provider also supplies the Internet Service Provider (ISP) functionality.

Service Application: Set of modules and configuration that collectively implement a specific function or set of functions, possibly across several Service Platforms. The Service Application concept is only relevant to the Service Deployment Manager.

Service Customer: Subscribes to services and pays the charges that are incurred using those services.

Service Deployment Manager: Acts on behalf of the Service Provider and deploys Service Modules on the Service Platform. The Service Deployment Manager manages all aspects related to the life cycle of Service Applications that are external to the Service Platform.

Service Module (or just module): downloadable, packaged collection of resources and/or code needed to provide a specific function

Service Provider: Is a business entity. The Service Provider supplies the necessary means to provide the business related support of a specific Service Application. The Service Provider is also responsible for delegating the task of service deployment to the Service Deployment Manager.

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ACS	Auto Configuration Server
CWMP	CPE Wan Management Protocol
RMS	Remote Management System

4 Architecture

4.1 Roles and entities

This clause defines operational roles and entities for an execution environment on the HG.

The HGI roles of "Broadband Service Provider" (BSP) and "Application Service Provider" (ASP) as defined in [i.1] may apply to several of the following roles, but both certainly apply to the "operator" role. For example, a BSP might take the role of the Service Deployment Manager, or chose to delegate this function to another company. It is up each BSP to decide which roles to take and which to delegate.

Figure 1 depicts roles (depicted as round-cornered rectangles) and entities (regular rectangles). Roles are individual persons or legal entities (e.g. companies), while entities are machines/software.

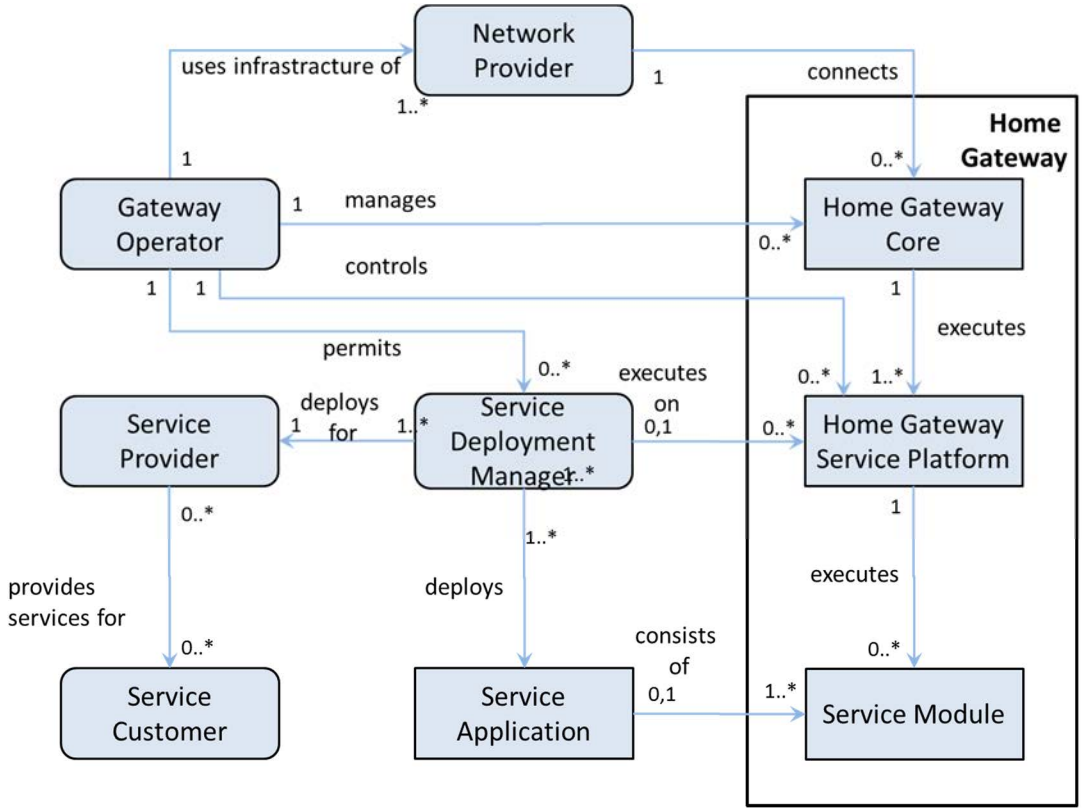

Figure 1: Entities, Roles and Relations between them

The various roles and entities are defined in clause 4.2.

4.2 Entities in the open platform 2.1 - enabled home gateway

Entities in the Open Platform 2.1 enabled Home Gateway are depicted in Figure 2. This is the reference scheme for the Home Gateway Software Architecture [i.3]. The HG consists of the core functions (HG_Core) plus one or more HG Execution Environments (HG_EE), each of them including a Service Platform (HG_SP) and software modules, all entities as defined below:

- **HG Core (HG_Core)** - The HG_Core uses an operating system (OS). On this operating system runs the HG native software that provides some or all of the home gateway functions as defined in the HGI Residential Profile [i.1]. Native drivers give direct access to hardware modules. The combination of the Home Gateway hardware, operating system, native software and drivers constitutes the HG_Core.

- **HG Execution Environment (HG_EE**) - The HG_EE is the combination of the Service Platform (HG_SP) and the Service Modules. The Service Platform (as defined in clause 4.1) runs on the HG_Core, and is used to provide flexible application software module execution and management. The platform allows life cycle control (install, start, update, stop and uninstall) of provider-managed software modules.. There may be more than one HG_EE running on top of the HG_Core .

- **HG Application Programming Interface (HG_API)** - The HG_API is provided to modules running in the HG Execution Environment (HG_EE), giving standardized access to residential gateway functions that are defined in the HGI Residential Profile. The minimum function set of the HG_API shall provide access to the applicable parameters defined in TR-181 [i.4] or TR-098 [i.5], profiled for the actual HG features. The HG_API can be implemented as part of the HG_Core and/or as modules running in an HG_EE; typically it is implemented as one or more modules. Access to the HG_API shall be explicitly granted by the operator for each module on a per function basis.

- **Management Agent (MA)** - The MA is a piece of software running on the HG and using the CWMP protocol [i.6] to handle commands from a RMS (Remote Management System) to the HG. Via CWMP and the MA, the RMS can set on and get parameters from the HG or execute HG functions. Moreover, by means of CWMP the MA is able to send notifications from the HG back to the RMS. No assumption is made in the present document about the internal HG implementation of the MA. It could be part of the HG_Core, or one or more modules running in the HG_EE, or a hybrid implementation.

4.3 Home gateway software architecture

This clause describes the software architecture for the integration of a Service Platform into an HG. The architecture is generic in the sense that applies to any chosen Service Platform technology.

4.3.1 Essentials

- An HG_API shall be defined for any specific Service Platform technology. All HGs using that specific technology shall implement the HG_API defined for that technology.
- There are no restrictions on how modules are implemented other than those defined by the chosen Service Platform technology.
- Any modules intended to be portable across different HG products shall not access any HG internal software interface outside of the HG_EE.
- Implementation of the HG_API is vendor-specific and not part of the present document.

4.3.2 Details

The HG_EE runs on top of the HG_Core. Modules run on top of the Service Platform and are able to communicate with each other to use each other's functions, if they are authorized to do so.

Using functions other than those provided by the Service Platform is allowed (e.g. to provide direct access to the HG_Core for vendor-specific modules), although doing so would probably limit the portability of a module.

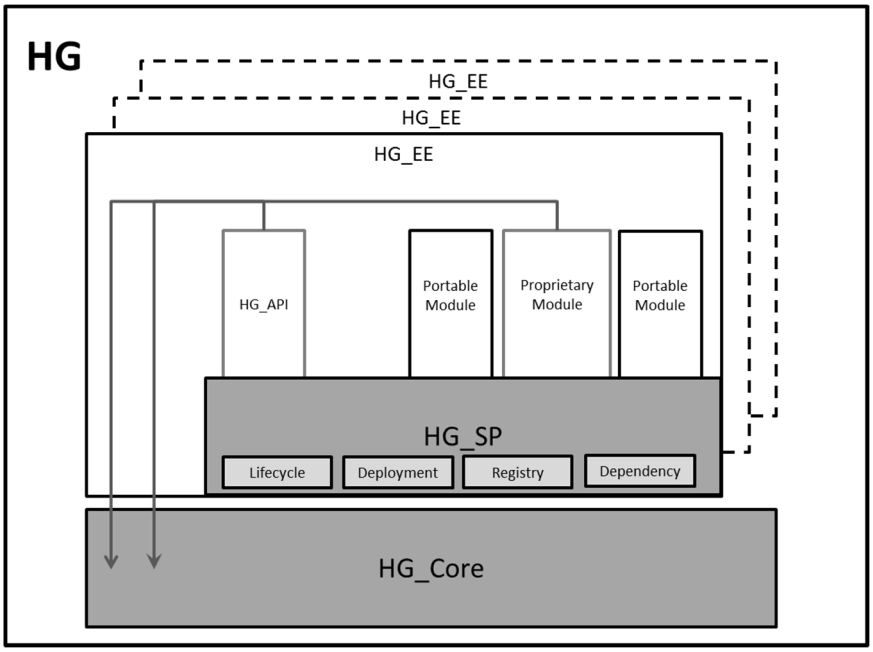

Figure 2: Home Gateway Architecture Model

The HG_SP has a number of functional blocks:

- **Lifecycle Management** - The Service Platform executes Service Modules and controls their execution. The lifecycle management function of the Service Platform provides APIs to start and stop these modules.

- **Deployment Management** - Modules are packaged in a format defined by the Service Platform technology and can be downloaded from remote repositories accessible through URLs. According to the URL, the proper protocol handler shall be made available (e.g. HTTP, HTTPS or FTP). Modules can then be installed which may imply unpacking on local storage or other specific one- time tasks.

- **Registry** - Module functions can be exposed through a local repository interface, which is specific to the Service Platform technology. The registry holds up-to-date information on which module is installed, which module is running, etc. Modules can use the registry to lookup functions implemented by other modules.

- **Dependency Management** - A module can depend on other modules to work correctly, e.g. on resources being present or services running. The dependency management function takes care of these relations, resolves dependencies and cleans up when a module is uninstalled.

4.4 Access to smart home non IP networks

There are three common models to address appliances in non-IP networks (e.g. switches, dimmers, sensors).

Some systems allow an external gateway to be attached to the HG and then be accessed over IP-based protocols (such as HTTP) in order to perform actions and queries on appliances connected to the external gateway using other communication protocols.

Another option is to integrate a radio module into the HG. Usually, these interfaces are connected through a UART connector and exposed to the operating system as serial interfaces. Since the application layer communication on this serial interface is proprietary or technology-specific, some kind of driver software shall be included into the HG's software stack by the provider of the serial interface.

Quite common is also the approach to attach an interface through USB ports. Most of the systems provide USB dongles that implement some type of virtual serial interface (a serial interface tunnelled through USB). Although there are USB class standards like Communication Device Class and Human Interface Device which could be used for virtual serial ports, many USB dongles use chips which make use of non-standard USB classes. However, USB dongles implementing virtual serial ports are also exposed as serial interfaces to the OS. While the IP gateway option does not require anything from the HG but IP based communication, the latter two options require the HG service platform to:

a) support access to serial line communication; and

b) provide an API for USB port management.

4.5 Remote management

4.5.1 Essentials

- The remote management protocol for HG_EE is CWMP [i.6] with its related data models TR-181 [i.4] or TR-098 [i.5].

- There is only 1 management entity for any given HG_EE (see role model in Figure 1).

- LAN-side management (by e.g. the customer) is not allowed for HG_EE.

- Management/configuration of the service aspect of modules is out of scope for the present document.

- Regular download protocols for modules are HTTP and HTTPS.

4.5.2 Details

Management of the software configuration of an HG is under the control of a (single) Gateway Operator. This includes the approval of any software before it is installed on to the execution environment.

Third party service providers are therefore dependent on the Gateway Operator for the installation of modules to deliver their services. This means there is a single management entity, and so this entity is able and responsible to resolve any resource conflicts.

An execution environment for software modules is just another aspect of an HG that needs to be remotely managed. As HGI has agreed to specify the CWMP [i.6] protocol for HG management, it is highly advisable that the management of the execution environment be based on the CWMP protocol [i.6] and related data model (TR-181 [i.4], TR-098 [i.5]).

The present document does not standardize the configuration of service applications. The configuration of service applications can be carried out in any appropriate way, e.g. by a local user interface, a network connection to a third party backend, or through the RMS of the operator.

A crucial aspect of the entire software module management is security and access control. E.g. third party service applications should not be allowed to perform any life cycle operations. The RMS should have exclusive control and responsibility about which modules to install, uninstall or update.

The HG is not able and not in charge to determine the order of install, uninstall and update actions, because it does not know the semantic dependencies between modules. This definition implies that the RMS shall only issue requests to the HG for uniquely identified modules. For example, an RMS shall not issue a single request to the HG to update "all" modules (or other "wildcard" like requests), but shall specify each module separately. On the other side, the HG shall deny any RMS issued request that cannot be resolved to a well-defined action.

4.5.3 Issues with firmware updates and software module management

If the software execution platform is part of a device's firmware, then a firmware update probably contains an empty execution platform, and therefore it is probably necessary to reinstall all needed modules and configurations. On a large scale, considering millions of devices and hundreds of applications per device, a firmware update could lead to strong scalability challenges on the RMS side (ACS if CWMP is used), since not only firmware images have to be downloaded, but also all the modules that were installed before the update.

Moreover, a firmware update might not necessarily change anything in the execution environment configuration. In order to avoid such scalability issues, the HG should assure the following points:

- Software modules should be kept across firmware updates.
- Software modules should only be downloaded if they are not already downloaded onto the HG's memory.
- Same applies to module configuration data.

These requirements can have hardware and software implications (like e.g. more flash memory to cache software modules).

5 Generic requirements

The following clauses contain high-level requirements to make the HG execution environment for software modules reliable, manageable and useable to application developers. Requirements in each section apply to one of:

- HG (i.e. may be implemented anywhere in HG).
- HG_Core.
- HG_EE (i.e. may be implemented anywhere in HG_EE).
- The HG_SP function of the HG_EE.

5.1 Basic requirements

Requirements in this clause could be implemented in the HG_Core or the HG_EE, or both.

Table 1

N°	Requirement
OP2.0-1	The HG SHALL have an internal system clock to provide the date and time of day. See note.
OP2.0-2	The HG SHALL support synchronisation of its internal system clock with an external time server via its broadband connection. The synchronization interval SHALL be configurable so that the time deviation is less than 10 seconds.
OP2.0-3	A user interface to set time and date locally SHALL only be made available to the end user when network based time sync is lost.
OP2.0-4	The HG_Core SHOULD support the installation of additional USB drivers as kernel modules.
OP2.0-4a	The HG_SP SHOULD support the installation of additional USB drivers.
OP2.0-5	The HG_Core SHALL provide a way to place a strict upper limit on the CPU load used by the HG_EE. This limit applies only when the HG_Core needs the resources.
OP2.0-6	The HG_Core SHALL provide a way to strictly limit the runtime memory used by the HG_EE.
NOTE: Valid date and time are crucial for the validation of certificates.	

The following requirements enable the HG to support additional USB classes, for example (W)HAN technology interfaces.

Table 2

N°	Requirement
OP2.0-7	The HG_Core SHOULD include USB class drivers for Human Interface Device (HID).
OP2.0-8	The HG_Core SHOULD include USB class drivers for Communications Device Class (CDC). See note 1.
OP2.0-9	The HG_Core SHOULD include drivers for FTDI Virtual Com Ports. See note 2.
OP2.0-10	If the HG_Core supports OP2.0-9, then it SHALL support configuration of the FTDI VCP drivers in terms of a list of supported USB Vendor Identifiers and Product Identifiers. This list SHALL provide at least 100 entries. All configured Vendor/Product identifiers SHALL be attached to the FTDI device driver. Reconfiguration of the FTDI driver SHOULD NOT require a reboot of the HG.
OP2.0-11	The HG_Core SHOULD include USB drivers for Silicon Labs® CP210x USB to UART bridge and CP2110 HID USB to UART bridge. See note 3.
OP2.0-12	The HG_Core SHOULD include USB drivers for Prolific PL-2303 USB to Serial Bridge Controller. See note 4.
OP2.0-13	The HG_Core SHOULD include USB drivers for SUNPLUS SPCP825A USB to UART Controller. See note 5.
NOTE 1: See http://www.usb.org/developers/docs/devclass_docs/ for details of USB device classes. NOTE 2: Details for FTDI drivers are available at http://www.ftdichip.com/Drivers/VCP.htm. NOTE 3: See http://www.silabs.com/products/mcu/pages/usbtouartbridgevcpdrivers.aspx. NOTE 4: See http://www.prolific.com.tw. NOTE 5: See http://www.sunplusmcu.com/soft/DS_SPCP825A_En.pdf.	

5.2 Reliability and security

Table 3

N°	Requirement
OP2.0-14	All functions of the HG_Core SHALL remain available if the HG_EE malfunctions or crashes.
OP2.0-15	The HG_SP SHOULD ensure that any single module cannot exceed assigned resources (CPU load, runtime memory).
OP2.0-16	The HG_SP SHALL be able to log life cycle operations (install, uninstall, update, start, stop) and any failures associated with these for each software module.
OP2.0-17	The HG_SP SHALL be able to log any unauthorized operation attempt.
OP2.0-18	The HG_SP SHALL ensure that only particular modules authorized by the gateway operator are able to reboot or terminate the HG_EE.

5.3 Life cycle management

Life cycle management refers to the management of modules and their existence on the HG. The life cycle actions for a module are installation, uninstallation, update, start and stop.

5.3.1 General

Table 4

N°	Requirement
OP2.0-19	When performing lifecycle operations the HG_SP SHALL NOT affect the lifecycle of modules that do not have dependencies on the module operated upon.

5.3.2 Installation

Installing a module means loading a module package to the HG_EE from a management backend, or triggering the HG_SP to download a module package from a URL.

Table 5

N°	Requirement
OP2.0-20	An HG_SP SHALL be able to check the signature integrity of all modules. Signature integrity provides the authorization for a module to be installed. If the integrity check fails, the HG_SP SHALL reject the module before installation.
OP2.0-21	The HG_SP SHALL NOT start any newly installed module automatically.
OP2.0-22	All installed modules within the HG_EE SHALL be persistent across firmware updates of the HG_CORE and/or HG_SP.

5.3.3 Uninstallation

Uninstalling a module means removing the module from the HG_EE non-volatile memory.

Table 6

N°	Requirement
OP2.0-23	Before uninstalling a module, the HG_SP SHALL inform all dependent modules for example to give them the opportunity for graceful shutdown.
OP2.0-24	Before uninstalling a module the HG_SP SHALL initiate the stop of the module.
OP2.0-25	After uninstalling a software module, the HG_SP SHALL release all volatile and non-volatile resources of the uninstalled module.

5.3.4 Update

Updating a module means to replace an existing module on the HG_EE with a more recent version of the module.

Table 7

N°	Requirement
OP2.0-26	The HG_SP SHALL allow updating software modules.
OP2.0-27	The HG_SP SHALL re-start a module after update if the module, before being updated, had been running.
OP2.0-28	Before updating a module the HG_SP SHALL initiate the stop the module.

5.3.5 Start

Starting a module means assigning system resources to a module and invoking an initialisation operation of the module.

Table 8

N°	Requirement
OP2.0-29	The HG_SP SHALL be able to start a module.
OP2.0-30	The HG_SP SHALL inform dependent modules that another module has been started.

5.3.6 Stop

Stopping a module means invoking an operation of the module that allows a graceful release of the resources that it was using.

Table 9

N°	Requirement
OP2.0-31	The HG_SP SHALL be able to stop a module.
OP2.0-32	Before stopping a module, the HG_SP SHALL inform all depending modules about this.
OP2.0-33	After stopping a software module, the HG_SP SHALL initiate the release of any dynamic resources that were assigned to the module.

5.4 Remote management

HGI specifies the use of a CWMP-based remote management system to manage the core functions of a Home Gateway. It is therefore appropriate to specify the same system for the lifecycle management of software modules on the HG. Compliance with the TR-181 [i.4] or TR-098 [i.5] BBF data model specifications is assumed in all of the following requirements.

Table 10

N°	Requirement
OP2.0-34	All remote management actions related to the HG_EE SHALL be atomic. If a remote management action fails, the HG_EE SHALL revert to the previous state.
OP2.0-35	The HG_SP SHALL be able to be managed remotely using the CWMP Protocol [i.6] and related data model .
OP2.0-36	The HG_SP SHALL NOT be managed by the end user.
OP2.0-37	The HG_Core and HG_EE SHALL be manageable using the following RPCs of the CWMP Protocol, defined in TR-069 [i.6] (see note): • ChangeDUState • DUStateChangeComplete • AutonomousDUStateChangeComplete In addition, the following restriction applies to the ChangeDUState RPC: UpdateOpStruct SHALL always specify the UUID parameter. Any other configuration SHALL be answered with a FaultStruct message.
OP2.0-38	The HG_Core and HG_EE data model SHALL support at least the following CWMP objects: - Object .SoftwareModules.ExecEnv.{i}. - for remote management of the HG_EE - Objects .SoftwareModules.DeploymentUnit.{i}. and .ManagementServer.DUStateChangeComplPolicy. - for remote management and monitoring of the installed modular software - Object .SoftwareModules.ExecutionUnit.{i}. - for remote management, including starting and stopping, and monitoring of installed applications. - Objects .DeviceInfo.MemoryStatus. - DeviceInfo.ProcessStatus. - for remote management and monitoring of the whole software environment of the device.
OP2.0-39	The HG_SP SHALL provide a locally unique identifier for each software module.
OP2.0-40	The HG_SP SHALL be able to execute software installation or update requests from a Remote Management System, where software module locations are URLs.
OP2.0-41	The HG_SP SHALL support HTTP and HTTPS as download protocols for Deployment Units (see TR-157 [i.8] for definition of Deployment units)
OP2.0-42	The HG_SP SHALL execute all requests from an Remote Management System to uninstall software modules.
NOTE: BBF issued a non-normative Theory of Operation for these RPCs in TR-157 [i.8].	

5.5 Module dependencies and interaction

A module might need to control the functions or the data resources of other modules. The HG_SP provides the means to ensure consistency on module configuration (no unresolved and stale references).

The HG_SP needs to support modules being able to register, advertise, retrieve and invoke interfaces.

Table 11

N°	Requirement
OP2.0-43	The HG_SP SHALL be able to manage dependencies between modules. These dependencies SHOULD include a version number or a version range. If no version is specified, the latest available version SHALL be used.
OP2.0-44	The HG_SP SHALL support interaction between modules via module interfaces.
OP2.0-45	The HG_SP SHOULD be able to hide the existence of modules from other modules e.g. for security reasons.
OP2.0-46	The HG_SP SHALL provide a mechanism for modules to advertise their interfaces. Access to interfaces not advertised SHALL be barred.
OP2.0-47	The HG_SP SHALL be able to allow and deny access to an interface of a module.
OP2.0-48	The HG_SP SHOULD be able to allow and deny access to specific operations of a module interface.

5.6 Supported protocols stacks

A basic set of protocol stacks is specified, to avoid the need for module-specific implementations.

Table 12

N°	Requirement
OP2.0-49	The HG_SP SHALL provide an API to an HTTP and HTTPS client protocol stack. If technology specific standardized APIs are available these SHALL be used.
OP2.0-49a	The HG_SP SHALL provide an API for an HTTP server protocol stack. If technology specific standardized APIs are available these SHALL be used.
OP2.0-50	The HG_SP SHOULD provide an API for an HTTPS server protocol stack.
OP2.0-51	The HG_SP SHOULD provide an agreed API to an UPnP protocol stack for implementing control points and services. If technology specific standardized APIs are available these SHALL be used.

5.7 HG application programming interface (HG_API)

Table 13

N°	Requirement
OP2.0-52	The HG SHALL provide an API to support communication between HG_EE modules and the HG_Core.
OP2.0-53	The HG_API SHALL provide access to HG_Core functions as specified in the HGI Residential Profile [i.1].
OP2.0-54	The HG_API SHALL provide access to management functions as specified in TR-181 [i.4] and TR-098 [i.5] for basic functionalities.
OP2.0-55	The HG_API SHALL provide access to management functions as specified in TR-104 [i.7] for telephony services.
OP2.0-56	If the HG supports SIP, the HG_API SHALL provide access to the SIP functions as specified in [i.1] R.260-R.265
OP2.0-57	The HG_API SHOULD provide access to the USB Class Drivers (Printer, SerialPort, Mass Storage, USB Hub, Wireless Controller) functions of the HG_Core.
OP2.0-58	The HG_API SHALL be able to provide access to the file system functions of the HG_Core for internal/external mass storage devices access.
OP2.0-59	The HG_API SHOULD support the HG_EE being informed of events happening in the HG_Core functions as specified in the HGI Residential Profile [i.1].
OP2.0-60	The HG_API SHALL maintain an access control list (ACL) to allow and deny access to operations on its interfaces.

6 Technology specific requirements

This clause contains technology specific Service Platform requirements, instantiating the HG_EE and HG_SP requirements when a Java/OSGi solution is adopted. HG_Core requirements are out of scope for clause 6.

To implement an HG compliant to HGI Open Platform 2.1 as specified in this clause, the developer will follow the generic requirements listed in clause 5 for the HG core part, on top of which the technology specific OSGi requirements below reported will be implemented.

6.1 OSGI service platform

The OSGi Service Platform is a Java™ based HG_SP with the addition of a comprehensive component model. **This clause defines a profile of Java™ and OSGi technology**.

In specifying OSGi specific requirements, the terms used to identify entities in the Open Platform 2.1 enabled Home Gateway defined in clause 4.2 are replaced for simplicity by OSGi specific terms. This mapping is reported in Table 14.

Table 14

Term in the present document	Terminology in OSGi
HG_EE	The underlying JRE, the OSGi Service Platform, and all installed bundles.
HG_SP	The HG_SP is equivalent to the OSGi Service Platform core components as specified in Release 5 [i.13] or Release 6 [i.14], and the JRE.
Service Module	One or more OSGi bundle(s) (jar file with OSGi compliant MANIFEST.MF).

6.1.1 Essentials

- OSGi Dmt Admin Service is used as HG-API, while the vendor-specific implementation of access to management resources is carried out through Data Plugins to Dmt Admin.
- Config Admin Service is mandatory, and its implementation is vendor-specific, because it provides persistence and needs to store data on the HG in a proprietary way.
- Declarative Services and Service Tracker specifications implementations are mandatory, although optional in OSGi.

6.1.2 Architecture

The instantiation of the generic Home Gateway architecture (see [i.9], [i.10], [i.11], [i.12], [i.13], [i.14] and [i.15]), for OSGi technology is depicted in Figure 3.

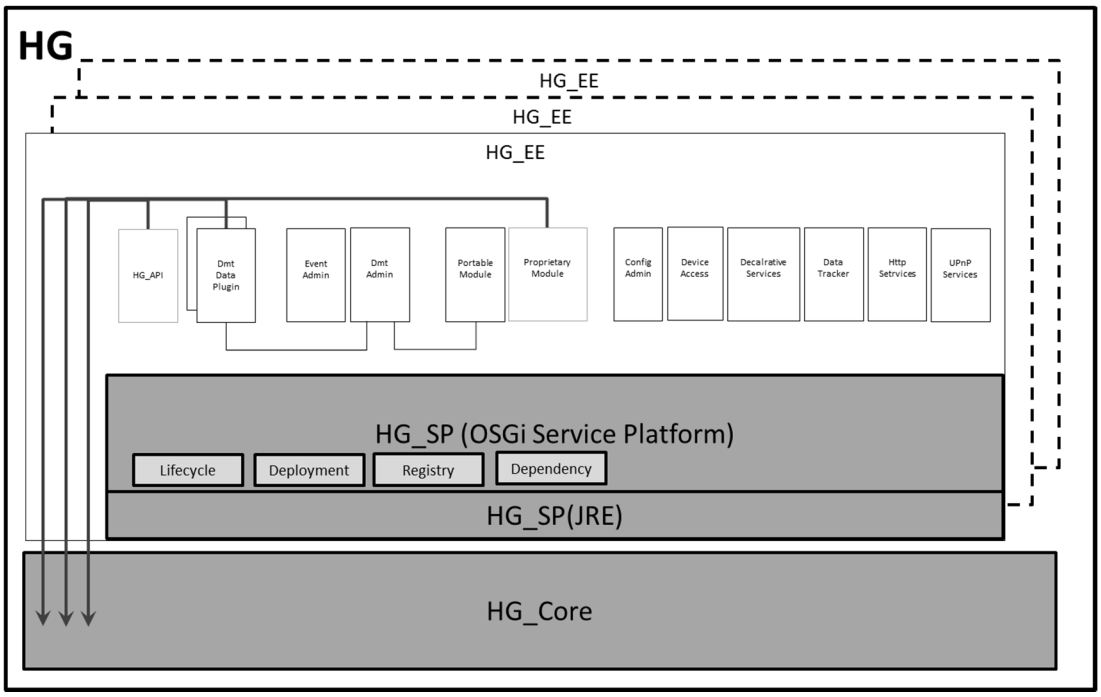

Figure 3: Component view of an OSGi based Home Gateway

6.1.3 OSGI specific requirements

The following requirements define a Java/OSGi-based execution environment. The rightmost column of table 15, labelled "Reference", shows the corresponding generic requirements, or N/A (not applicable) if no generic counterpart is defined.

NOTE: Applications developed under OSGi Service Platform Core R5 will run under OSGi Service Platform Core R6.

Table 15

N°	Requirement	Reference
OP2.0-61	The OSGi Service Platform SHALL at a minimum comply with the Java™ Runtime Environment (JRE) Java™ 8 compact1 profile (see [i.16]).	OP2.0-49 (see note)
OP2.0-61a	The OSGi Service Platform Core SHALL be compliant with the OSGi Service Platform Core R5 (see [i.13]) + OSGi Service Platform R4.3 Residential Service Specification [i.15] OR a later version.	OP2.0-23 to OP2.0-33, OP2.0-43, OP2.0-44, OP2.0-46
OP2.0-61b	The OSGi Service Platform Core SHOULD be compliant with the OSGi Service Platform Core R6 (see [i.14]) + OSGi Service Platform R6 Residential Service [i.15] specification.	OP2.0-23 to OP2.0-33, OP2.0-43, OP2.0-44, OP2.0-46
OP2.0-62	The OSGi Service Platform SHALL implement resource management following the guidelines in annex B.	OP2.0-5, OP2.0-6, OP2.0-15
OP2.0-63	The OSGi Service Platform Core SHALL be able to verify the integrity of bundles signed with certificates issued by an operator-specific list of root certificates (either public ones by Certification Authorities (CA) or private ones).	OP2.0-20
OP2.0-64	The OSGi Service Platform Core SHALL provide a trust model for authenticating the bundle signatures. The trust model SHALL support chains of certificates.	OP2.0-20
OP2.0-64a	The OSGi Service Platform Core SHALL support signature algorithms in signed jar, according to OP2.0-63.	OP2.0-20
OP2.0-65	The OSGi Service Platform Core SHALL support the Security Layer. If OSGi Service Platform Core Release 5 is implemented then Security Layer SHALL be compliant to [i.13], chapter 2. If OSGi Service Platform Core Release 6 is implemented then Security Layer SHALL be compliant to [i.14], clause 2.	OP2.0-45, OP2.0-47, OP2.0-48

N°	Requirement	Reference
OP2.0-65a	The OSGi Service Platform Core SHOULD support the Start Level API specification. If OSGi Service Platform Core Release 5 is implemented then Start Level API SHALL be compliant to [i.13], if OSGi Service Platform Core Release 6 is implemented then Start Level API SHALL be compliant to [i.14].	N/A
OP2.0-65b	The OSGi Service Platform Core SHOULD provide a Persistent Storage area to each installed and authorized bundle. If OSGi Service Platform Core Release 5 is implemented then Persistent Storage concept and API specification SHALL be found in clause 4.5.2 of [i.13], if OSGi Service Platform Core Release 6 is implemented then Persistent Storage concept and API specification SHALL be found in clause 4.5.2 of [i.14].	N/A
OP2.0-66	All bundles installed in the OSGi Service Platform SHALL be persistent across firmware updates of the HG.	OP2.0-22

OP2.0-67	The HG_SP SHALL deny invocations to unauthorized bundles, at least the following operations in all signature variants (because they could compromise the Service Platform itself):
	`System.exit`
	`System.load`
	`System.loadLibrary`
	`System.setSecurityManager`
	`System.setIn`
	`System.setErr`
	`System.setOut`
	`System.clearProperty`
	`System.setProperty`
	`System.setProperties`
	`Runtime.getRuntime().halt`
	`Runtime.getRuntime().exec`
	`Runtime.getRuntime().loadLibrary`
	`Runtime.getRuntime().traceInstructions`
	`Runtime.getRuntime().traceMethodCalls`
	`Runtime.getRuntime().addShutdownHook`
	`Runtime.getRuntime().removeShutdownHook`
	Any attempt by unauthorized bundles to invoke one the above operations SHALL result in throwing an exception to the invoking bundle, typically a `java.lang.SecurityException`.
		OP2.0_18

OP2.0-68	HG_SP SHALL deny access to the `java.reflect` package to unauthorized bundles.	N/A
OP2.0-69	For unauthorized bundles, the HG_SP SHALL deny the installation of native code libraries for the Java™ Native Interface (JNI). Any attempt to install a bundle that contain native code libraries that is not authorized, SHALL result in throwing an exception to the entity that tried to install the bundle, typically a `java.lang.SecurityException`.	N/A
OP2.0-70	The OSGi Service Platform SHALL include an implementation of the Device Manager as defined in the Device Access specification. If OSGi Service Platform Residential Release 4.3 is implemented then Device Manager SHALL be compliant with clause 103 of [i.13], if OSGi Service Platform Residential Release 6 is implemented then Device Manager SHALL be compliant with clause 103 of [i.14].	N/A
OP2.0-71	The OSGi Service Platform SHALL include an implementation of the Configuration Admin Service specification. If OSGi Service Platform Residential Release 4.3 is implemented then Configuration Admin Service SHALL be compliant to clause 104 of [i.13], if OSGi Service Platform Residential Release 6 is implemented then Configuration Admin Service SHALL be compliant to clause 104 of [i.14].	N/A

N°	Requirement	Reference
OP2.0-72	Configuration data stored through the OSGi Service Platform Configuration Admin Service SHALL be persistent across firmware updates of the HG or of any bundles of OSGi Service Platform.	N/A
OP2.0-72a	The OSGi Service Platform SHALL support the Metatype Service. If OSGi Service Platform Residential Release 4.3 is implemented then Metatype Service SHALL be compliant to clause 105 of [i.13], if OSGi Service Platform Residential Release 6 is implemented then Metatype Service SHALL be compliant to clause 105 of [i.14].	N/A
OP2.0-73	The OSGi Service Platform SHALL support the Declarative Services (also known as Service Component Runtime) specification. If OSGi Service Platform Residential Release 4.3 is implemented then Declarative Services SHALL be compliant to clause 112 of [i.13], if OSGi Service Platform Residential Release 6 is implemented then Declarative Services SHALL be compliant to clause 112 of [i.14].	N/A
OP2.0-74	The OSGi Service Platform SHALL support the (Service) Tracker specification. If OSGi Service Platform Residential Release 4.3 is implemented then Tracker SHALL be compliant to clause 701 of [i.13], if OSGi Service Platform Residential Release 6 is implemented then Tracker SHALL be compliant to clause 701 of [i.14].	N/A
OP2.0-75	The OSGi Service Platform SHALL support the DMT Admin Service specification. If OSGi Service Platform Residential Release 4.3 is implemented then DMT Admin Service SHALL be compliant to clause 117 of [i.13], if OSGi Service Platform Residential Release 6 is implemented then DMT Admin Service SHALL be compliant to clause 117 of [i.14].	N/A
OP2.0-76	The OSGi Service Platform SHALL support the Event Admin specification. If OSGi Service Platform Residential Release 4.3 is implemented then Event Admin SHALL be compliant to clause 113 of [i.13], if OSGi Service Platform Residential Release 6 is implemented then Event Admin SHALL be compliant to clause 113 of [i.14].	N/A
OP2.0-77	Implementations of the Event Admin specification SHALL support a configurable limit on the number of simultaneously active Java™ threads.	N/A
OP2.0-78	OSGi Service Platform SHALL support the Log Service specification. If OSGi Service Platform Residential Release 4.3 is implemented then Log Service SHALL be compliant to clause 101 of [i.13], if OSGi Service Platform Residential Release 6 is implemented then Log Service SHALL be compliant to clause 101 of [i.14].	N/A
OP2.0-79	Implementations of the OSGi Log Service SHALL limit their use of RAM.	N/A
OP2.0-80	The implementation of the OSGi Log Service specifications SHALL NOT write data into any built-in flash memory of the HG.	N/A
OP2.0-81	The OSGi Service Platform SHALL support the Http Service specification. If OSGi Service Platform Residential Release 4.3 is implemented then Http Service SHALL be compliant to clause 1012 of [i.13], if OSGi Service Platform Residential Release 6 is implemented then Log Service SHALL be compliant to clause 102 of [i.14].	OP2.0-49a
OP2.0-82	The implementation of OSGi Http Service specification SHALL support a configurable limit on the number of simultaneously active Java™ threads.	N/A
OP2.0-83	The OSGi Service Platform SHOULD support the UPnP™ Device Service specification. If OSGi Service Platform Residential Release 4.3 is implemented then UPnP™ Device Service SHALL be compliant to clause 111 of [i.13], if OSGi Service Platform Residential Release 6 is implemented then UPnP™ Device Service SHALL be compliant to clause 111 of [i.14].	OP2.0-51
OP2.0-84	The OSGi Service Platform SHALL support the Conditional Permission Admin Service specification. If OSGi Service Platform Core Release 5 is implemented then Conditional Permission Admin Service SHALL be compliant to clause 9 of [i.13], if OSGi Service Platform Core Release 6 is implemented then Conditional Permission Admin Service SHALL be compliant to clause 9 of [i.14].	OP2.0-47
OP2.0-85	The OSGi Service Platform SHALL include one or more Data Services as specified in DMT Admin Service Specification Version 2. If OSGi Service Platform Residential Release 4.3 is implemented then DMT Admin Service SHALL be compliant to clause 117 of [i.13], if OSGi Service Platform Residential Release 6 is implemented then DMT Admin Service SHALL be compliant to clause 117 of [i.14]. The above Data Services SHALL allow OSGi Bundles to have access to the TR-181 [i.4] and TR-098 [i.5] data models.	OP2.0-54 to OP2.0-56
OP2.0-85a	If the HG supports telephony services then the Data Services required in OP2.0-85 SHALL also allow OSGi Bundles to access the TR-104 [i.7] data model.	OP2.0-55
OP2.0-86	The OSGi Service Platform SHALL be manageable according to the Theory of Operation contained in TR-157 Amendment 3, Appendix II [i.8].	OP2.0-34 to OP2.0-42
OP2.0-87	If the HG_Core provides SIP functionality as stated by R.260-R.265 of [i.1] on the OSGi Service Platform SHALL include an implementation of the JSR180, SIP API for J2ME [i.18].	OP2.0-56
OP2.0-88	The HG_SP SHOULD include the javax.usb API, as specified in JSR-80 [i.17].	OP2.0-57

N°	Requirement	Reference
OP2.0-89	The HG_SP SHOULD support the Java™ Communications API Version 3 (javax.comm) implementation.	OP2.0-57
NOTE: HTTP and HTTPS clients are provided by the java.net.URL class in the Java™ 8 compact1 profile.		

Annex A (normative): JAVA™ compatibility guidelines

This annex defines some Java™ runtime compatibility guidelines to enable developers to create portable applications that run on a HGI 2.1 compliant OSGi Service Platform as it has been described in the present document.

The Java™ implementation behind an HGI 2.1 OSGi Service Platform shall provide bytecode compatibility and a minimum runtime. This Java™ implementation along with the runtime is what it will be called in the rest of the document, HGI 2.1 Java™ Profile.

Portable applications are applications that can run on any HGI 2.1 OSGi Service Platform, compatible platform without modification or recompilation. To remain portable, applications shall not exceed the Java™ profile described here.

A.1.1 Bytecode compatibility

The present document requires a Java™ 8 JVM. Note that Google's Dalvik VM (used in the Android OS) is not bytecode compatible with this JVM implementation in any way, so it is not compliant with the present document.

A.1.2 JAVA™ runtime

The minimum Java™ profile is Java™ 8 compact1. This profile is backward compatible with the OSGi/Minimum-1.0 profile, that is a CDC-1.1/Foundation-1.1 without javax.microedition packages, and thus with the HGI Open Platform 2.0 specification (see [i.19]).

A.1.3 Extensions

The HGI Open Platform 2.0 specification required explicitly the addition of the Java™ Secure Socket Extension (JSSE) that were only optionally available in the CDC-1.1/Foundation-1.1 profile. The Java™ 8 compact1 profile includes these extensions by default.

Annex B (normative): Resource control guidelines for JAVA™

When we talk about managed resources in Java, we talk about anything in the Java™ Runtime that has constrained availability, be it physically (there is only a certain amount of memory in the device), or logically (you might create as many Java™ threads as you like, but if you run too many performance dies). Some applications might use resources sparing and careful ("The Good"), some are malware and want to harm the system ("The Bad"), and some are just not well programmed ("The Ugly").

When it comes down to the resources we shall control, we find this common characteristic: Excessive use of the resource harms system stability, and is typically a result of intentional or unintentional bad application programming. (Note that this annex does not deal with limitation of resources due to accounting and charging reasons.)

Proprietary VMs (e.g. IBM J9, Oracle Java™ Embedded Client, /K/ Embedded Mika Max, Myriad Jbed) provide resource management features. Some OSGi vendors have shown that many features are feasible at the OSGi bundle level (e.g. ProSyst®, Makewave®).

B.1.1 RAM

Talking about RAM usage, Java™ provides completely transparent memory management regarding both allocation and release of memory. There is no Java™ standard way for an application to control memory management policies.

The Java™ VM allocates and releases memory blocks by its own, proprietary policies. At first sight, it appears to be a good idea to manage the maximum amount of memory an application might use, but this idea also leads to the non-trivial challenge how to find an appropriate figure. Some examples:

1) Available memory divided by number of applications: Some applications need more, some less memory. Calculating the memory quota this way may lead to badly working applications, since they would get different memory quotas depending on the number of installed applications.

2) Fixed amount of memory for each application: Limits the number of installable applications to a constant limit, and as long as you run fewer applications and/or applications that use very little memory probably a lot of memory is wasted.

Therefore, your algorithm to calculate memory quotas could be probably wrong, too static or inefficient.

If implemented, memory management in Java™ is a proprietary feature that some Java™ products provide. RAM is one of the resources that no Java™ standard addresses today. However, OSGi platform vendors do provide products that are able to assign memory quotas to bundles using proprietary JVM mechanisms.

B.1.2 CPU sharing

Another common issue on constrained devices is that an application might consume too much CPU power and harm other applications in terms of user experience or even stability. In a Java/OSGi environment, there is no other entity to control CPU allocation of an application than the Java™ Virtual Machine itself.

It is not trivial to quantify or measure CPU load for an application, so it is also not trivial for an operator to configure CPU sharing policies in a consistent and efficient way.

A standard API for CPU allocation management is available to a certain extent on some Java™ platforms.

(Java™ 2 Platform Standard Edition 5.0 or higher) through Java™ Management Extensions.

B.1.3 Devices

Accessing internal or external devices (built-in radio interfaces, USB dongles) of an HG from multiple applications might cause racing conditions, dead locks, undesired and blocking latency and many other issues. A solution for these issues is to delegate all direct communication between the Java™ Runtime and a device to a single application, an OSGi driver (not to be mixed up with the native, OS based driver). These OSGi drivers' task is to coordinate access to their connected devices.

B.1.4 Running JAVA™ threads

Running too many Java™ threads simultaneously is a typical programming trap developers run into. First of all, applications should never create threads by just invoking new Thread(…), but at least use a thread pool with a maximum size. Moreover, a centralized entity that manages exclusively all available threads could help too, and could be offered as part of an SDK.

B.1.5 Open TCP connections

Again, too many open outgoing or incoming TCP connections could seriously harm a Java™ Runtime's consistency and stability, and are also typical in combination with using an uncontrolled number of threads. A centralized entity that controls exclusively all available Java™ Socket and ServerSocket instances could help.

B.1.6 Conclusions

There are many Java™ entities that could be managed on Java™ level, including running threads and open connections; unfortunately, there is currently no appropriate standard mechanism for this, but even a proprietary implementation is better than nothing. Implementations should take care of the resources' status: It is not important how many thread objects have been created, but are running, and you also should not care about the number of Socket class instances, but about opened sockets.

On the other side, resources like memory and CPU allocation can only be managed successfully through the JVM itself. The policies of these allocations, as well as actions upon violation of policies, depend on many factors like limitations of the device, types of applications and even business models. These policies are hard to standardize and should be implemented in a proprietary manner.

However, the Java™ Runtime should provide an API to allow a Java™ based resource policy management that includes Java™ level entities as well as memory and CPU.

HGI will cooperate with other organizations to enable standard resource management APIs in future. While plain Java™ APIs should provide access to the management features of the JVM, an OSGi API should provide application based resource management.

For now, the HGI strongly recommends that resource management for the aforementioned resource types be implemented using standard APIs where possible and proprietary APIs otherwise.

Annex C (informative): Bibliography

UPnP Forum: "Device Architecture 1.0".

NOTE: See http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0-20080424.pdf.

HGI SBI-052: "Hardware Virtualisation in Home Gateways, and comparison to NFV, Multi-Threading and Containers".

NOTE: See http:// http://www.homegatewayinitiative.org/.

JSR 219: "Foundation Profile 1.1.2".

NOTE: See https://jcp.org/ja/jsr/detail?id=219.

IETF RFC 2616: "Hypertext Transfer Protocol -- HTTP/1.1".

NOTE: See http://www.ietf.org/rfc/rfc2616.txt.

ETSI TS 103 425 (V1.1.1) (11-2016): "Publicly Available Specification (PAS); Smart Machine-to-Machine communications (SmartM2M) Home Gateway Initiative RD039-Requirements for Wireless Home Area Networks (WHANs) Supporting Smart Home Services".

History

Document history		
V1.1.1	November 2016	Publication