RFC0145 - Home Gateway Administration

Draft

30 Pages

Abstract

This RFC specifies the architecture and mapping rules for an OSGi based admin service to access and manage the various aspects and underlying services of the Home Gateway Device. An implementation of the admin service enables OSGi services running on an Internet Gateway Device to access the core level functions of a home gateway. An important part of this RFC is the specification of mapping rules between the Broadband Forum's TR documents, such as TR-098, and the DMT Admin tree.

0 Document Information

0.1 Table of Contents

0.2 Terminology and Document Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in 9.1.

`Source code is shown in this typeface.`

0.3 Revision History

The last named individual in this history is currently responsible for this document.

Revision	Date	Comments
Initial 0.5	Mai 29th 2009	Initial version for review. Andreas Kraft
0.6	June 3rd 2009	Included changes according to the audio conference on June 2 nd 2009 and the first review of the document. Andreas Kraft
0.7	June 4th 2009	Included changes according to comments in the document and e-mails. Clarified session handling. Clarified event handling. Changed security considerations. Andreas Kraft
0.9	June 5th 2009	Updated illustration 5. Fixed typos. Rephrased description of the use of session.id property. Andreas Kraft
0.10	October 10th 2009	Updated RFC accordingly to REG discussions and experiences gained with the reference implementation. Use the term "IGD" consistently in the document. Moved the description for the DMT Subtree Admin in a separate section and made it more generic. Renamed properties to reflect their plural nature. Added dmtAction property. Added an illustration for overlapping subtrees. Andreas Kraft
0.11	Feburary 3rd 2010	Section 5.1.2 removed the limitation that the DMT Subtree Admin only manages trees below the root node. Also added limitation for overlapping "normal" nodes. Andreas Kraft
0.12	April 27th 2010	Updated RFC according to comments. Added a section to explain the terminus of *scaffold nodes*. Andreas Kraft

Revision	Date	Comments
0.13	April 29th 2010	Updated RFC according to comments. Fixed figures. Added a new section that explains the mapping, and moved the various mapping definitions to that section. Andreas Kraft, Steffen Drüs edow
0.14	August 25th 2010	Closed this RFC. Added a chapter (Chapter 8) to explain the reason and to point to RFC 141, which will continue the word on the ideas and concepts presented in RFC 145. Andreas Kraft

1 Introduction

This RFC specifies the general architecture for an OSGi admin service to access and manage the various aspects and underlying services of the Internet Gateway Device (IGD). This service enables OSGi services running on an IGD to access the core level functions of an IGD, for example NAT and the firewall. There are various benefits which would be gained from such a service:

- **Increase compatibility**
  Make a retail home gateway compatible for a variety of operators to support full retail business; install operator-specific software on a retail home gateway (e. g. Web-based Management UI, VoIP Termination (B2BUA), VoD Termination (RTSP)).

- **Accelerate Differentiation**
  Install standards based as well as proprietary applications on an IGD (e. g. TR-069 remote management agents, UPnP IGD service, TR-064 LAN-side CPE management).

- **Leverage WAN Services**
  Connect home appliances to WAN-centric services (e. g. VoIP phones and other devices that need to be accessed from the WAN).

- **Good User Experience**
  Provide customers with branded user interfaces accommodated to their needs and experiences, as well as easily support plug&play devices in the LAN.

The following high-level use cases cover only a small number of application that are enabled by this service.

- **The customer is managing certain aspects of an IGD, such as his credentials, Wi-Fi SSID, NAT/PAT forwarding.**
  An OSGi-service running on the router provides an HTML-based user interface for the customer where he can change and manage various settings of the IGD. The OSGi-service validates the user's input and makes the necessary changes via the IGD Service.

- **The IGD is managed remotely via a TR-069 remote management agent.**
  The IGD vendor does not provide a TR-069 remote management service on his own. Instead, an OSGi-based TR-069 service is installed by the ISP. The TR-069 service manages the IGD via the IGD Service. In case an ISP supports an other management protocol than TR-069, that ISP installs an OSGi bundle that implements that management protocol.

- **An IGD vendor implements the UPnP Forum's IGD specification as an OSGi service.**
  So far, IGD vendors implement the UPnP IGD services as part of their firmware. That means that they have to implement a minimal UPnP stack. In an OSGi-enabled residential gateway that functionality could be moved to the OSGi framework. That would make this service more manageable and adaptable to changes in the protocol and environment. Another aspect is that only one UPnP stack needs to be installed on the IGD.

- **A SIP B2B User Agent manages the port forwarding of the IGD.**
  Services, such as a SIP B2B User Agent, could access functions of the IGD which need to be managed in order to provide certain services. In this use case, temporarily opening the WAN firewall and forwarding of certain ports to other devices in the LAN is an essential part of a SIP service that is running as an OSGi service on the IGD.

2 Application Domain

The main application domain for a Home Gateway Admin service is the home gateway that acts as a manageable device between the home network (LAN) and the public Internet (WAN). Even a low-priced router model includes a lot of basic network functions that need at least some simple administrative care, either by the network operator, the home user, or even by connected third-party equipment.

The network operator wants to offer an easy to use client interface for the basic operations the user might need to perform in order to install the device. This includes, for example, the provisioning of credentials or setting the identification for a local Wi-Fi network. Sometimes an automated setup process helps the home user to install a new device without any interaction at all.

An experienced home user can change some of the more arcane settings of the home gateway. For example, for security reasons he wants to change the Wi-Fi identification, enable or disable services, or he needs to change some other settings. For this he needs a rich user interface for the device-internal functions. Today, the home gateway hosts a web page or other user interface application that enables the home user to administer these settings.

The home user bought a VoIP-enabled phone. After he connects the phone to the local network and provided the necessary phone settings, he expects the phone to work properly. The phone itself “knows” how to connect to a telephony service in the Internet, but for receiving calls some adjustments in the home gateway have to be made in order to forward IP calls to the device in the LAN.

3 Problem Description

The UPnP Forum defines an network-side interface to the functionality of an Internet Gateway Device [3].. However, the UPnP Forum does not define a similar interface to the internal services of a home gateway, nor does the OSGi specification, yet. Another standards body that defined a specification to manage the management functions for a home gateway is the Broadband Forum [6].. The BBF's Technical Recommendation TR-098 “Internet Gateway Device Data Model for TR-069” describes the Internet Gateway Device data model for the CPE WAN Management Protocol (TR-069) [5].. TR-069 defines the generic requirements of the management protocol methods which can be applied to any TR-069-enabled CPE.

Consider a residential service gateway that supports the UPnP Forum's IGD specification, and also runs OSGi technology to support networked residential services. A service running on the home gateway that wants to access and manage the gateway functions has no other choice as to call the IGD service interface via UPnP, even if both are running on the same hardware environment. Beside of the more complicated and error-prone service architecture and load on the LAN, a vendor of a home gateway needs to provide more resources to support access to the UPnP stack on the device, even if no other service on the OSGi part of the home gateway needs access to it.

A Home Gateway Admin, standardized by the OSGi Alliance, would provide well-defined means to the core functions of a home gateway, hiding complexity as well as vendor specific implementation details. A vendor, ISP, or other service provider could implement their IGD-enabled applications on top of the HG Admin architecture as a portable module, e.g. the HTML-based user interface to the router.

So far, no standardized OSGi-based HG-related architecture and means to access and manage IGD internally, and possibly other, functionality exists.

The following figure presents a rough sketch of an architecture that would utilize the OSGi Home Gateway Admin (HG Admin). The vendor specific core functions of the gateway are made available through a unified IGD Service interface, which enables various types of services, such as Management Agents, GUIs, and even the UPnP IGD service itself.

It is expected that an implementation of the Home Gateway Admin would usually contain native code parts.

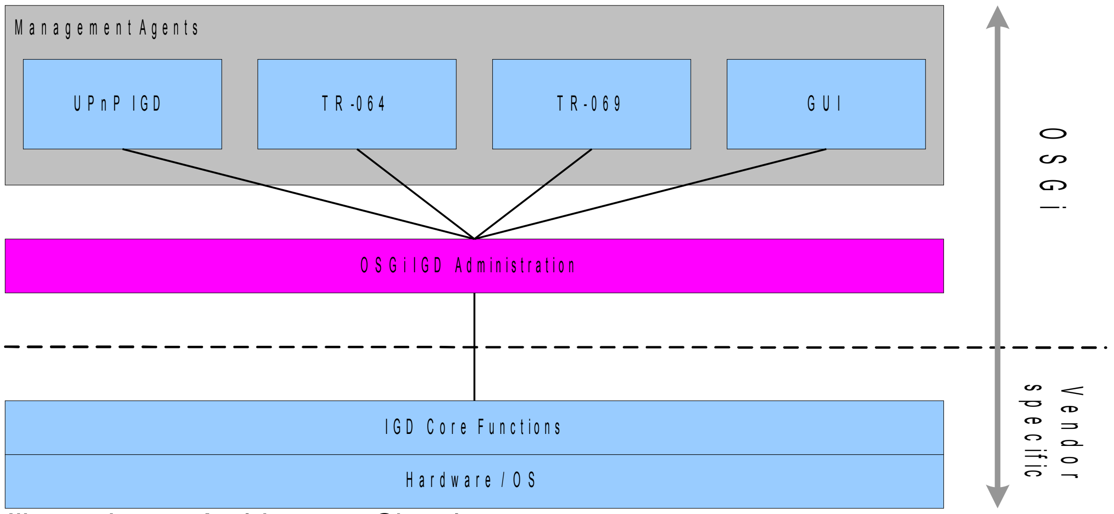


Illustration 1: Architecture Sketch

4 Requirements

The following functional and non-functional requirements are given for the Home Gateway Device Service.

4.1 Functional Requirements

- An implementation of the Home Gateway Admin service MUST enable the management of the core functionality of an IGD.
- An implementation of the Home Gateway Admin service MUST provide means to map the functionalities of the Broadband Forum's TR-098 specification [6]..
- An implementation of the Home Gateway Admin service MUST provide means to map the functionalities of the UPnP Forum's IGD 1.0 specification [3]..
- An implementation of the Home Gateway Admin service SHOULD support the new functionalities of the upcoming Broadband Forum's TR-098 Amendment 2.
- An implementation of the Home Gateway Admin service SHOULD support the new functionalities of the upcoming UPnP Forum's IGD 2.0 specification.
- The Home Gateway Admin SHOULD support the management of further TR-069-managed services, such as TR-104 [7]..
- The Home Gateway Admin MUST notify interested services when certain changes in the managed, e.g. the IGD, services happen. Examples:

- connect, disconnect, and reconnect to WAN,
- detect a new IP device, and
- firewall intrusion detection.

- A service that has been notifi ed by the Home Gateway Admin SHOULD be able to reject certain requests.
- The specification of the Home Gateway Admin MUST allow implementations to support and use IPv6.

4.2 Non-Functional Requirements

The following security considerations MUST be taken into account:

- Access to methods of the IGD's drivers MUST be controlled.
- The Home Gateway Admin SHOULD support the extensions to the UPnP IGD specification made by the Broadband Forum [6].
- The OSGi DMT Admin MUST be used in order to set and retrieve configuration values for managing the IGD.
- Events that originate in the core layer of the IGD must be forwarded to interested OSGi services by the OSGi Event Admin [11]..

5 Technical Solution

5.1 Scope

The technical solution covers all needed mechanisms to access the low level IGD functions from within the OSGi service platform. It enables the implementation of e. g. a web-based configuration user interface, a UPnP Internet Gateway Device service, or a TR-069 management agent. However, the solution does not cover the implementation of any of these examples.

5.2 General Considerations

OSGi-based implementations of IGD management functions are tightly coupled with the IGD's operating system and core functions (as specified in the HGI Residential Profile [8].). Hence, considering these implementations to be drivers in the meaning of the OSGi Device Access specification [OSGi Service Compendium, chapter 103] i s very recommended.

Moreover, plugging an additional hardware module to the IGD has a standardized process to find and install additional management bundles (Module Driver) as needed. For example, an additional IEEE 802.11n Wi-Fi access point might be plugged into the IGD, and the Device Access process finds and installs a suited driver that allows for managing this new module.

However, implementing the management bundles as Device Access “driver” is not mandatory.

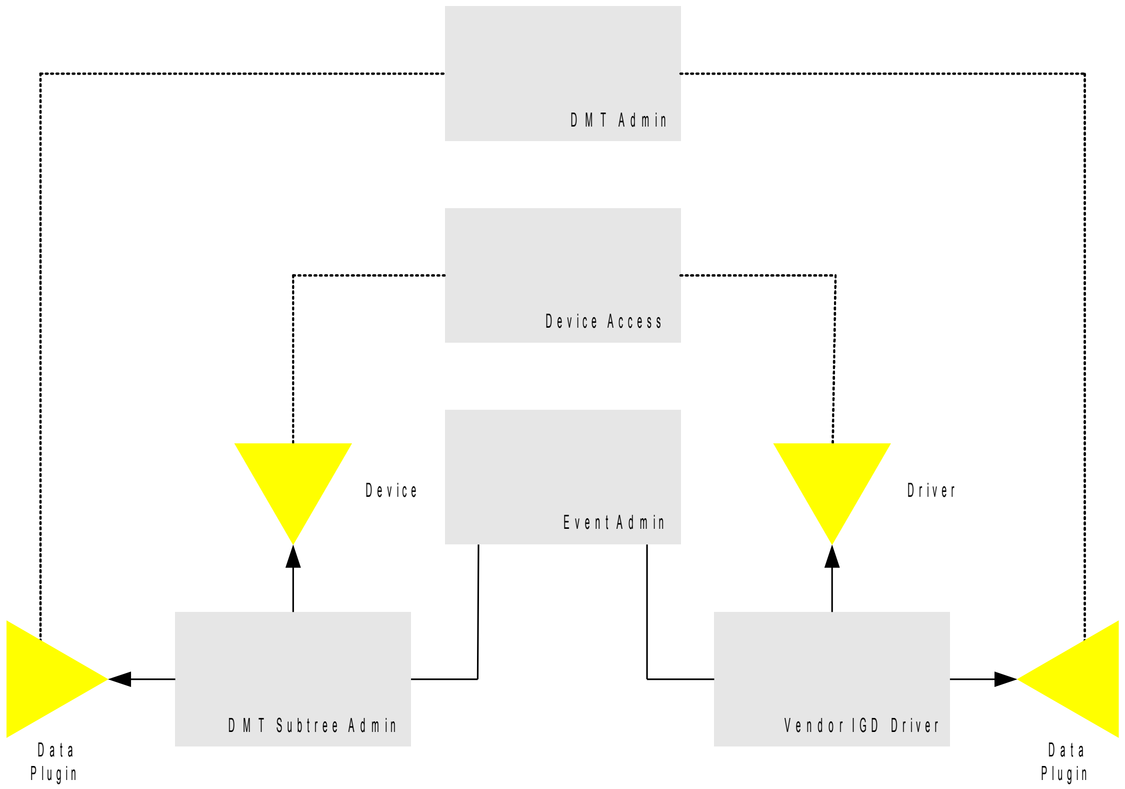


Illustration 2: Internet Gateway Device Administration components relationship

The OSGi DMT Admin provides a generic interface to access management aspects of a device. Implementations of the DMT Subtree Admin and Vendor IGD Drivers have to use the DMT Admin service to offer a configurations-based interface in order to set and retrieve values. State variables of the underlying Home Gateways have representations in the DMT Admin. Actions are called by setting the appropriate state variables in the Device Management Tree.

Vendor IGD Driver bundles register one or more Data Plugins to manage the configurations of all or only a sub-tree of the IGD. A software vendor can choose to implement the whole IGD configuration functionality in one Vendor IGD Driver bundle or to split it into many bundles to separate the functionality.

Generic mapping rules are defined in chapter 5.6 to map IGD state variables, actions, and functions to a well defined DMT configuration sub-tree. The root of this sub-tree is “./ *InternetGatewayDevice*”. Though this section mainly focuses on aspects of IGD management, the DMT Subtree Admin can be used manifold to manage overlapping subtrees in the DMT Admin in general.

Events that arise from the IGD core functions, for example when a disconnect from the network occurs, must be raised and distributed using the OSGi Event Admin service. No special event types are defined in this specification. Instead, event types from the DMT Admin services are used (s. [9].).

Within the OSGi service platform all events are considered as “active”. That means that any event will be distributed through the Event Admin service, and any permitted entity running in the OSGi service platform is able to receive these events. The Broadband Forum's TR-069 specification allows for “passive” notifications, which are handed over to a Remote Management System (RMS) as part of the next scheduled contact between the management agent locally installed on a home gateway and the RMS. For this case, the management agent is responsible for keeping track of all events and storing them persistently until the next contact.

5.3 Roles and Functional Blocks

Illustration 3 presents the functional blocks defined or referred to in this specification and are explained in this section.

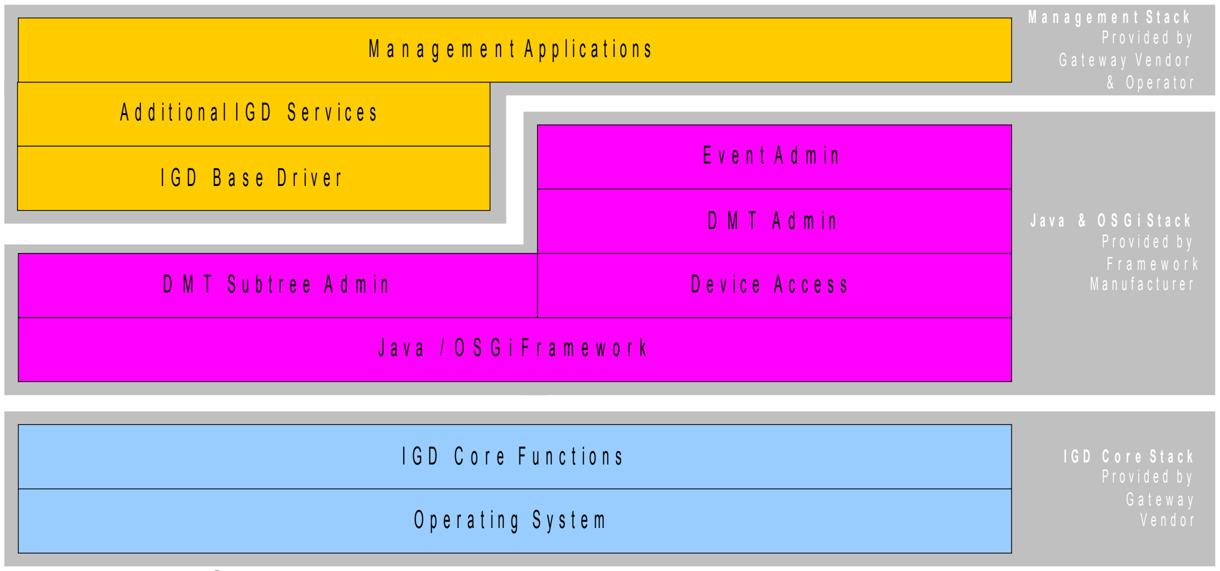


Illustration 3: IGD Functional Blocks

The roles used in the diagram are:

- *Gateway Vendor*: The vendor of the gateway. He is responsible for assembling the hardware and the software for the IGD, and usually delivering it to the Operator. He might develop and manufacture software and hardware components himself, or order them from a third party.
- *Framework Manufacturer*: The manufacturer of the OSGi Framework and the OSGi services. He provides an implementation to the Gateway Vendor, possibly tailored to the specific hardware.
- *Operator*: The Operator defines the product requirements of the IGD and provides these IGD to his customers. He orders IGD from the Gateway Vendor. This order might include the management-related OSGi services, but he can also obtain them from a third party provider.

The *HG Core Stack* with the *Operating System* and the *IGD Core Functions* are usually provided by the Gateway Vendor. The IGD Core Functions layer contains functionality for all the basic aspects of a IGD, such as configuration of the WAN interfaces, establishment and maintaining WAN connections, but also a firewall, NAT and routing functionality.

The *Java & OSGi stack* and other OSGi services are provided by the Framework Manufacturer. Services that are mandatory by this specification are *DMT Admin*, *Event Admin*, and the *DMT Subtree Admin*. As explained before, the *Device Access* service is optional. An Operator could define other means to install IGD plugin bundles.

The *Management Stack* contains the necessary services for managing an IGD. The *IGD Base Driver* implements the necessary base functions to access and manage the *IGD Core Functions*. Additional functions and/or drivers that can be installed if necessary and on-demand are represented in the *Additional IGD Services* block. The *IGD Base Driver* can be implemented and provided as a single bundle, or split into as many bundles as necessary. This specification doesn't imply any restrictions on the number of Vendor IGD Driver bundles. Finally, the *Management Applications* are the actual applications that are used to manage a service. This could be a TR-069 management agent for remote management of the IGD, a web user interface for the customer, or an implementation of the UPnP IGD service.

5.4 DMT Subtree Admin

This section specifies the operation of the DMT Subtree Admin service

5.4.1 DMT Admin restrictions

Using the DMT Admin to implement the a Home Gateway Administration service is a mandatory requirement because it allows for a loose coupling between the Management Services and the Vendor IGD Drivers. To understand the function and work of this loose coupling a discussion on the restrictions of the DMT Admin is necessary. The HG Admin is implementing solutions to overcome these restrictions.

The DMT Admin has the restrictions that subtrees of configuration data cannot overlap. A DMT Admin Data Plugin that was registered to manage, for example, the sub-tree "*./InternetGatewayDevice*" receives all requests to data objects below that name. It is not allowed to register a second Data Plugin that, for example, manages the sub-tree "*./InternetGatewayDevice/wan".* Because of this restriction services which register later in time are not allowed to register a Data Plugin to manage the tree or any sub-tree below "*./InternetGatewayDevice*". Figure 4 illustrates overlapping subtrees, one of an IGD Data Plugin and the overlapping one, the Vendor IGD Driver Data Plugin that can manage the data model for *SomeVendorDevice*.

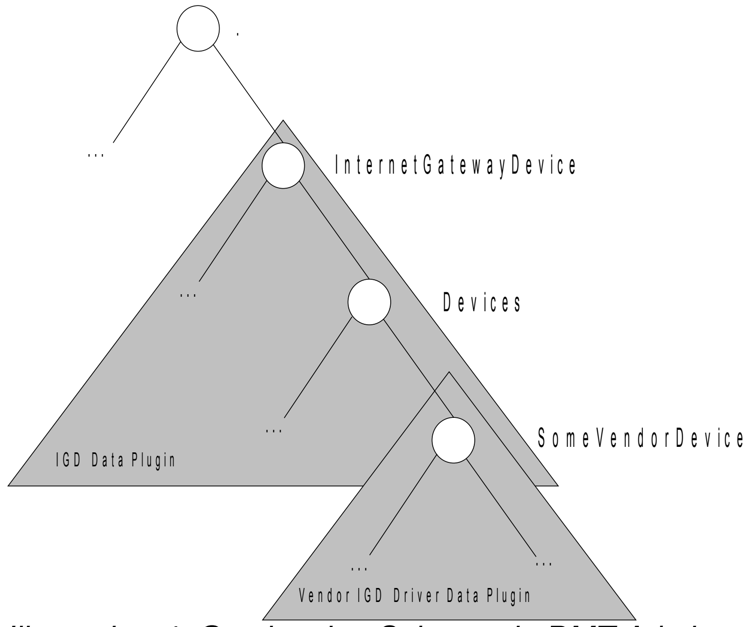


Illustration 4: Overlapping Subtrees in DMT Admin

A bundle provider can of course register the Data Plugin so that that it just registers itself somewhere in the DMT Admin's configuration tree. But in this case, any Management Service that likes to set or access configuration values to or from that bundle does not know the value's path in the configuration tree.

Another restriction concerns the raising of events. The DMT Admin only raises events when a part of the managed configuration is changed via a Data Plugin. A bundle that likes its configuration been managed by the DMT Admin and therefore registers a Data Plugin cannot use the DMT Admin to notify interested bundles if some part of its configuration has changed through other means. An example for this is that in an IGD the connection on the WAN interface was established or disconnected. This event arises in the IGD Core Functions layer. The responsible driver bundle can reflect this in its internal configuration mapping, but there are no means to trigger the DMT Admin to raise an event.

5.4.2 Managing and Mapping DMT Admin subtrees

The solution for the non-overlapping subtrees problem described above is to let the DMT Subtree Admin handle the mappings between the actual subtree under a specific configuration in the DMT Admin tree and the registered subtree of any Data Plugin of driver bundles. For this, the following procedure apply:

- The DMT Subtree Admin must implement the *ManagedServiceFactory* interface and register a ManagedServiceFactory service with the *PID=org.osgi.service.dmtsubtree* .
- For each configuration dictionary received, the DMT Subtree Admin must create and register a new Data Plugin. The only configuration value of this dictionary is *rootnode*. The Data Plugin must be registered with the property *dataRootURIs* set to the value of *rootnode*. This value indicates the subtree below the DMT root node it manages. The subtree can be located directly under the DMT tree's root node or anywhere else below it. The DMT Subtree Admin is responsible to fill in any missing nodes.
- When more than one dictionary contains the same value for *rootnode* only the first one received is registered. All others must be ignored.
- The DMT Subtree Admin must unregister the Data Plugin when the according configuration dictionary is deleted.
- The DMT Subtree Admin tracks Data Plugin services that have the properties *dataRootURIs* and *configurationPaths* set. The Filter string to track this is “(&(objectclass=info.dmtree.spi.DataPlugin)(dataRootURIs=*)(configurationPaths=*))”
- A bundle that wants to be configured through the DMT Admin and that maps its configuration subtree into the subtree of another bundle must register a Data Plugin with the property *dataRootURIs* set to any value. Since this property is an array of Strings, a Data Plugin can be registered to handle more than one configuration path. It is recommended that the bundle first checks with the DMT Admin for the availability of the desired path within the DMT Admin tree. This configuration path must not overlap with any previous registered Data Plugin. In addition it also sets the following properties: [1]
    - *configurationPaths:* this property defines the Data Plugin's intended position under a tree in the DMT Admin. This property contains an array of Strings. The number of entries in this array must match the number of entries of the *dataRootURIs* property.
    The value for that property must not start or end with a slash (character /). The path must be fully qualified, ie. It starts from the beginning of the DMT Tree. Example:

1 The names of the properties defined here might change when a more generalized mechanism to circumvent the problems with the DMT Admin is specified.

"./InternetGatewayDevice/Devices/SomeVendorDevice". ~~The DMT Subtree Admin tries to find a matching own registered Data Plugin. A match is found if the *configurationPath* element is part of the subtree of a *dataRootURIs element* of an own DataPlugin (see DMT Admin 117.2.3). In this case the DMT Subtree Admin does the mapping. If none of the own registered Data Plugins matches then the tracked Data Plugin is ignored for now. Note, that later the DMT Subtree Admin might register a Data Plugin that matches the previously ignored tracked Data Plugin. In this case the tracked Data Plugin must be associated then.~~
~~Overlapping paths are allowed following the rules specified in section Error: Reference source not found. The resolution is handled by the DMT Subtree Admin, depending on the value of the *configurationMultiples* property (s. below).~~
This property is mandatory. There is no default.

- *configurationMultiples*: this property defines whether there could be multiple instances for the given configuration path and objects. This property contains an array of Boolean. If set, the number of entries in this array must match the number of entries of the dataRootURIs property.
  An example is if an IGD has more than one WAN interfaces (e. g. one DSL and one 3G) and needs a driver for each of the WAN ports. In this case two different Data Plugin implementations would be used for the configuration of the driv[illegible]
  ~~to be notified about this violation. [illegible]isbehaving Data-Plugin~~~~It is the responsibility of the DMT Subtree Admin to assign a unique identifier for the configuration path to ensure a correct mapping for each Data Plugin. This identifier must be uniquely assigned for the configuration path and Data Plugin and stored persistently by the DMT Subtree Admin. This means that a once assigned identifier is always assigned to the same Data Plugin, even when that Data Plugin is unregistered and registered again, or the OSGi framework is restarted. Note, that the unique identifier storage might be removed when the DMT Subtree Admin is uninstalled. It is therefore necessary for a management domain to define other means to ensure the persistency of the unique identifiers.~~

  ~~Allowed values for this property are the string values "true" and "false". If the value is *true* then multiple instances are created. If the value is *false* then only one instance for the given configuration path is allowed. In the later case a newer registration of a Data Plugin with the same configuration path does not override an existing one, ie. it is ignored.~~
  ~~It is possible and allowed for a node to contain both multiple instances of subtrees (as indicated by the property *configurationMultiples* set to true) and non-multiple subtrees. In other words, a node can contain „multiple" and „normal" child nodes. A „normal" child node, though, must not overwrite the path name of an existing „multiple" or "normal" child node. This includes „multiple" child nodes that are currently mapped and those that are currently not mapped (but the multiple-ID has already been assigned). In case a "normal" child node violates this, the DMT Subtree Admin raises an OSGi Event with the *topic* "org/osgi/service/dmtsubtree/NODE_ALREADY_EXISTS". Even if one Data Plugin violates the rules above more than once, only one Event is raised, containing information about all offending *configurationPath* elements. Mandatory properties of this event are: *configurationPaths, a* String[] containing the violating plugin's *configurationPaths* elements, and *dataRootURIs*, also a String[] with the corresponding plugin's *dataRootURIs* elements. This allows the~~
  The setting of this property is optional. The default is *false*.

- *dmtActions*: This optional property contains a relative path that must point to a node below the Vendor Data Plugin's subtree (specified in *dataRootURIs)* that contains leaf nodes for internal DMT Subtree Admin management purposes. This way the DMT Subtree Admin can communicate with the Data Plugins it manages (see also 5.4.5). This property contains an array of Strings.
  If set, the number of entries in this array must match the number of entries of the *dataRootURIs* [illegible]erty.
  [illegible]e is only one leaf node defined that must be supported by a Vendor Data Plugin that sets this property. See also illustration 10 for an example.

  - *mappedPath*: The value of this leaf node is a String and contains the path under which the DMT Subtree Admin has put the mapping to this Vendor Data Plugin. If the leaf node does not exist, it is created by the DMT Subtree Admin. It is removed by the DMT Subtree Admin when the Vendor Data Plugin is unmapped.

- The DMT Subtree Admin receives tracker callbacks for registered (and unregistered) Data Plugin services. It is responsible to map the Configuration paths, which are assigned through the configuration, to the real Data Plugins that handle the subtrees accordingly.

- If a *ManagedFactoryConfiguration* is changed or removed, all tracked Data Plugins must be newly associated or the associations must be removed, accordingly.

- Configuration requests from Management Applications are done through the DMT Admin. The DMT Admin will request a DMT Session object from the DMT Subtree Admin's Data Plugin that is registered for handling the designated path. This DMT Session then will handle the actual requests by mapping the request paths and forwards the requested actions to the actual driver bundles' Data Plugins. This is done by getting DMT Session objects via the DMT Admin from the actual Data Plugins.
  When closing, committing or rolling back sessions the DMT Subtree Admin's DMT Session is responsible for performing the according actions on these session objects (see also 5.4.5 for a more detailed discussion).

The following illustration 5 presents the general relationships between the involved components.

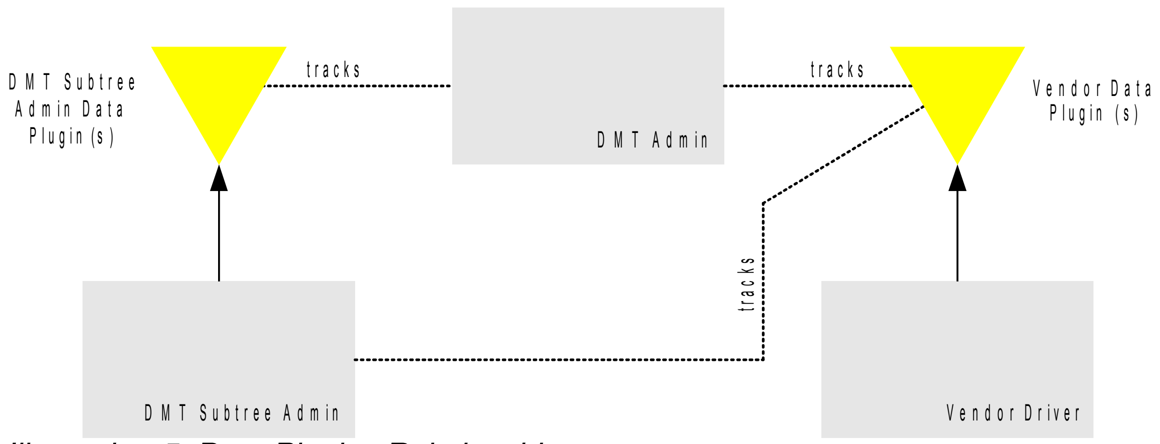


Illustration 5: Data Plugins Relationships

The creation and usage of Session objects are presented in illustration 6 .

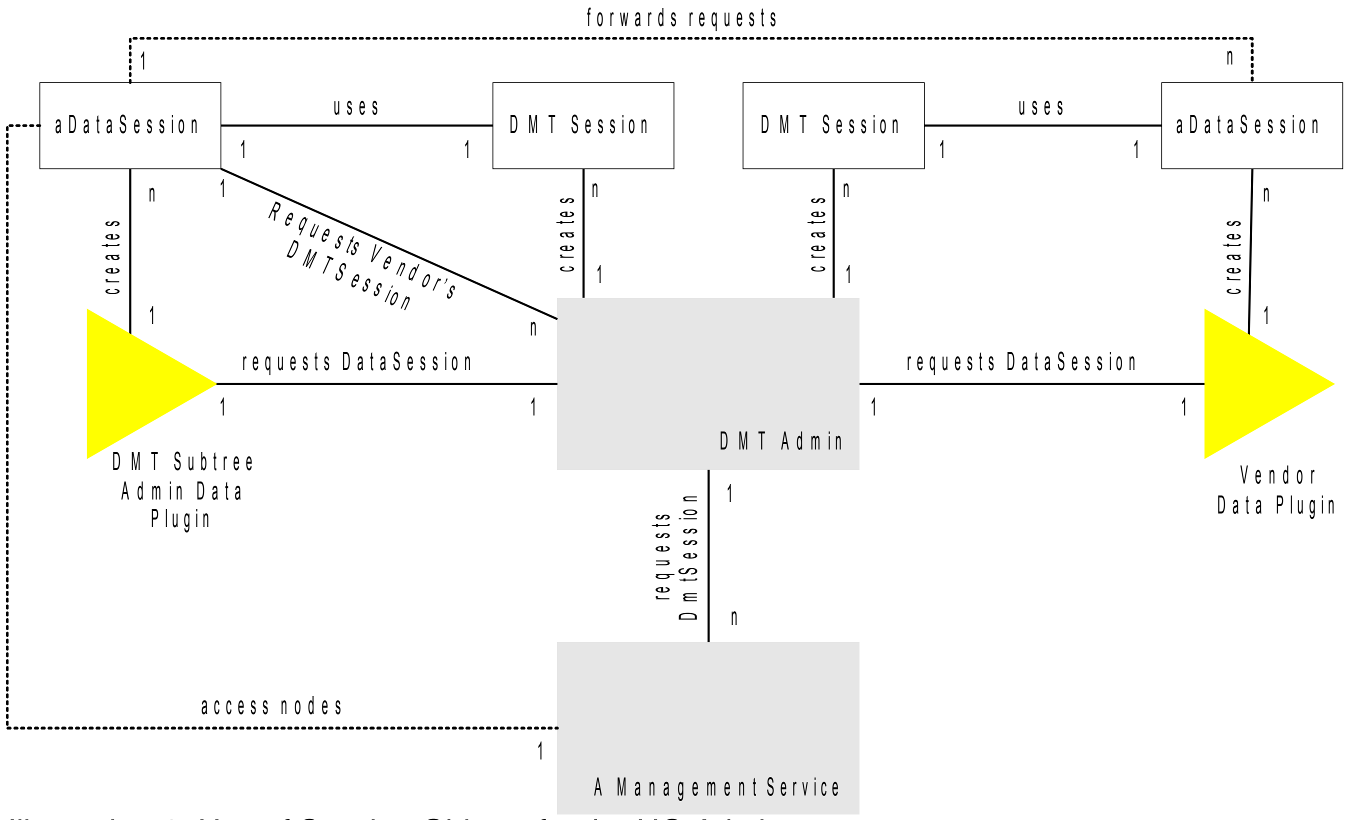


Illustration 6: Use of Session Objects for the HG Admin

It is strongly recommended, though not mandatory by this specification, to implement all Data Sessions as *Transactional Data Session* (s. DMT Admin Service Specification, chapter Data Sessions). In case of transactions the DMT Subtree Admin's DMT Session is responsible to create transactions, and to call the appropriate actions when the transaction is committed or rolled-back by the using Management Service.

It is strongly recommend, though not mandatory by this specification, to implement the DMT Session objects of the DMT Subtree Admin's Data Plugin in a way that it only requests DMT Session objects on-demand from the drivers' Data Plugins that are involved in the requests.

5.4.3 Mapping rules ~~for overlapping subtrees~~

Definitions

The definitions from chapter 117.2.3 (DMT Admin, "Tree Terminology") apply within this specification.

Additionally this specification defines the following terms:

own Data Plugin:	a Data Plugin that is registered by the DMT Subtree Admin. An own Data Plugin is registered for each configured *rootnod.*
mapped path:	the absolute path under which a data plugin has been mapped into the DMT tree. For Data Plugin's registered with *configurationMultiples* = { "true" } this *mapped path* includes the uniquely assigned identifier.

virtual subtree	the subtree of a vendor Data Plugin at its mapped location in the DMT tree. The root of the *virtual subtree* is its *mapped path.*
scaffold node:	a node that is created by the DMT Subtree Admin whenever the following condition applies: A registered Vendor Data Plugin is not located directly under the dataRootURI of an own Data Plugin. The resulting gap is filled by one or more scaffold nodes. .

Illustration 7: Scaffold nodes

Scaffold nodes must be removed as soon as it is not required anymore to fill a gap caused by a Data Plugin registration.

The DMT Subtree Admin is responsible for creating and managing scaffold nodes and to hande read operations on them. All operations of an info.dmtree.spi.ReadableDataSession are allowed. Any other operation is forbidden and leads to a DmtException of type COMMAND_NOT_ALLOWED.

General mapping behavior

Whenever a vendor Data Plugin is registered the DMT Subtree Admin tries to find a matching own registered Data Plugin. A match is found if the *configurationPaths* element is part of the subtree of a *dataRootURIs* element of an own Data Plugin (see DMT Admin 117.2.3). In this case the DMT Subtree Admin does the mapping. If none of the own registered Data Plugins matches then the tracked Data Plugin is ignored for now. Note, that later the DMT Subtree Admin might register a Data Plugin that matches the previously ignored tracked Data Plugin. In this case the tracked Data Plugin must be associated then.
Overlapping paths are allowed following the rules specified in the next section. The resolution is handled by the DMT Subtree Admin, depending on the value of the *configurationMultiples* property.

~~*The definitions from chapter 117.2.3 (DMT Admin, “Tree Terminology”) apply within this specification.*~~

~~Additionally this specification defines the following terms:~~

~~*own Data Plugin* – a Data Plugin that is registered by the DMT Subtree Admin. An own Data Plugin is registered for each configured *rootnod.*~~

~~*scaffold node* – a node that is created by the DMT Subtree Admin whenever one of the following applies~~

1. ~~A registered own Data Plugin is not located directly under the root of the DMT tree. The resulting gap is filled by one or more scaffold nodes.~~
2. ~~A registered Vendor Data Plugin is not located directly under the dataRootURI of an own Data Plugin. The resulting gap is filled by one or more scaffold nodes.~~

If ~~the~~an element of *configurationMultiples* is set to *true* It is the responsibility of the DMT Subtree Admin to assign a unique identifier for the according element of *configuration* ~~p~~*Paths* to ensure a correct mapping for each Data Plugin. This identifier must be uniquely assigned for the configuration path and Data Plugin and stored persistently by the DMT Subtree Admin. This means that a once assigned identifier is always assigned to the same Data Plugin, even when that Data Plugin is unregistered and registered again, or the OSGi framework is restarted. Note, that the unique identifier storage might be removed when the DMT Subtree Admin is uninstalled. It is therefore necessary for a management domain to define other means to ensure the persistency of the unique identifiers.
Allowed values for this property are the ~~string~~boolean values ~~“~~*true*~~”~~ and ~~“~~*false*~~”~~. If the value is *true* then multiple instances are created. If the value is *false* then only one instance for the given configuration path is allowed. In the later case a newer registration of a Data Plugin with the same configuration path does not override an existing one, ie. it is ignored.

It is possible and allowed for a node to contain both multiple instances of subtrees (as indicated by according the property element in *configurationMultiples* set to true) and non-multiple subtrees. In other words, a node can contain „multiple“ and „normal“ child nodes. A „normal“ child node, though, must not overwrite the path name of an existing „multiple“ or “normal” child node. This includes „multiple“ child nodes that are currently mapped and those that are currently not mapped (but the multiple-ID has already been assigned). In case a “normal” child node violates this, the DMT Subtree Admin raises an OSGi Event with the *topic* “org/osgi/service/dmtsubtree/NODE_ALREADY_EXISTS”. Even if one Data Plugin violates the rules above more than once, only one Event is raised, containing information about all offending *configurationPath* elements. Mandatory properties of this event are: *configurationPaths*, *a* String[] containing the violating plugin's *configurationPaths* elements, and *dataRootURIs*, also a String[] with the corresponding plugin's *dataRootURIs* elements. This allows the bundle or system manager to be notified about this violation.

Handling overlapping Data Plugins

If two vendor Data Plugins are registered with exactly the same configuration path and both *configurationMultiples* set to *false*, then only the first registered plugin is mapped and the second will be ignored. In this case the the DMT Subtree Admin raises an OSGi Event with the *topic* "org/osgi/service/dmtsubtree/NODE_ALREADY_EXISTS" with the same properties as defined in the previous section. If the *mapped path* gets ~~free~~available again (because the first plugin ~~will be unregistered again~~was un-registered), a re-evaluation and mapping of the currently unmapped vendor Data Plugin s will be performed.

If an element of~~the~~ *configurationPaths* of a registered vendor Data Plugin points to a *scaffold node* and the accoring element~~value~~ of *configurationMultiples* is *false*, then this mapping will be performed as shown in illustration 7~~Error: Reference source not found~~.~~Exception of type COMMAND_NOT_ALLOWED.~~

• ~~MTD Scaffold nodes must be removed as soon as it is not required anymore to fill a gap caused by a Data Plugin registration.~~

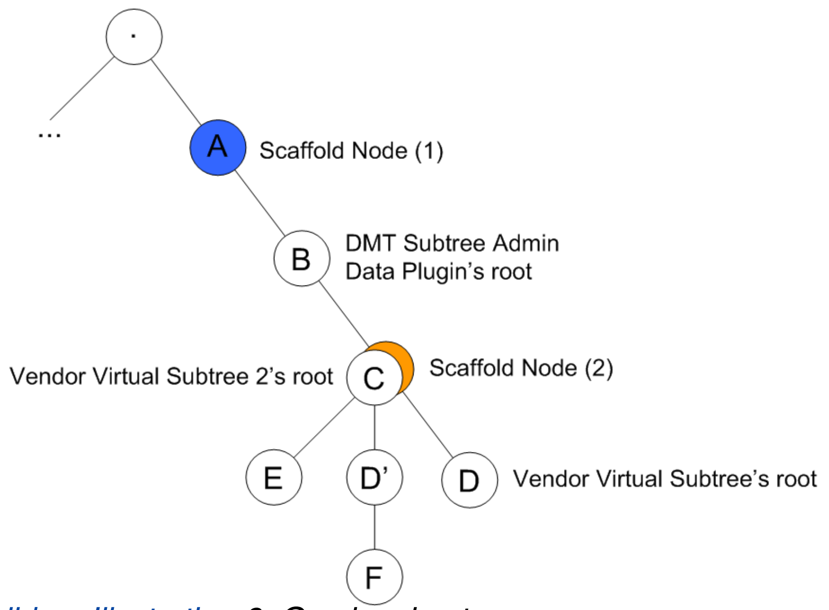
• ~~The DMT Subtree Admin is responsible for creating and managing scaffold nodes and to hande read operations on them. All operations of an info.dmtree.spi.ReadableDataSession are allowed. Any other operation is forbidden and leads to a~~ As the result there is one scaffold node and one virtual subtree mapped to the same *mapped path*. The risk of potential conflicts is given, if the name of a *virtual tree* childnode is the same as the child of the scaffold node. (In illustration 10~~Error: Reference source not found~~ this is the case with nodes D and D'). In order to resolve such conflicts, the following rule ~~is defined~~must be applied:

If a scaffold node and a virtual subtree's root are mapped to the same path, then the child node of a scaffold node always has precedence ~~about~~of the child node of a virtual subtree's root.

If in our example *DmtSession.getChildNodeNames()* ~~would be~~is called for node path "./A/B/C" ~~then~~ then the result ~~would~~must be [E] and [D] (not [D']).~~browsing the child nodes of [C] (*Scaffold Node (2)*) would return [D]. Error: Reference source not found For the example shown in illustration~~ All calls to node specific operations for node path "./A/B/C/D" must resolve this node to [D] and not [D'].

This rule ensures a defined behavior which is independent of the registration order of the plugins.

•


.
...
A
Scaffold Node (1)
B
DMT Subtree Admin Data Plugin's root
C
Scaffold Node (2)
Vendor Virtual Subtree 2's root
E
D'
D
Vendor Virtual Subtree's root
F

~~Abbildung~~Illustration 9: Overlapping trees

~~In case a second vendor plugin [C'] would register itself so that it overlaps [C] (i.e. configurationPath = ./A/B/C') the DMT Subtree Admin must merge the names of the children of this second plugin (C') and Scaffold node (2) [C]. The returned child node names for this node would be [E] and [D] and not [D']. In case of a conflict as described here the scaffold node always wins~~

5.4.4 Mapping Rules Examples

The following table shows examples which reflect the mapping rules defined in the previous section. It is assumed that the DMT Subtree Admin registered a Data Plugin for the *dataRootURIs = ./InternetGatewayDevice* .

Vendor Data Plugin			**Resulting Mapping**
dataRootURIs	**configurationPaths**	**configurationMultiples**	
./vendor_x/firewall	./InternetGatewayDevice/service/firewall	false	./InternetGatewayDevice /service/firewall
./vendor_x/interface/WAN	./InternetGatewayDevice/interface/WAN	true	./InternetGatewayDevice /interface/1234/WAN~~1234//~~

Table 1: Mapping Rules Examples

The following illustration shows an exemplary tree, the properties for the mappings as well as the values set by the DMT Subtree Admin.

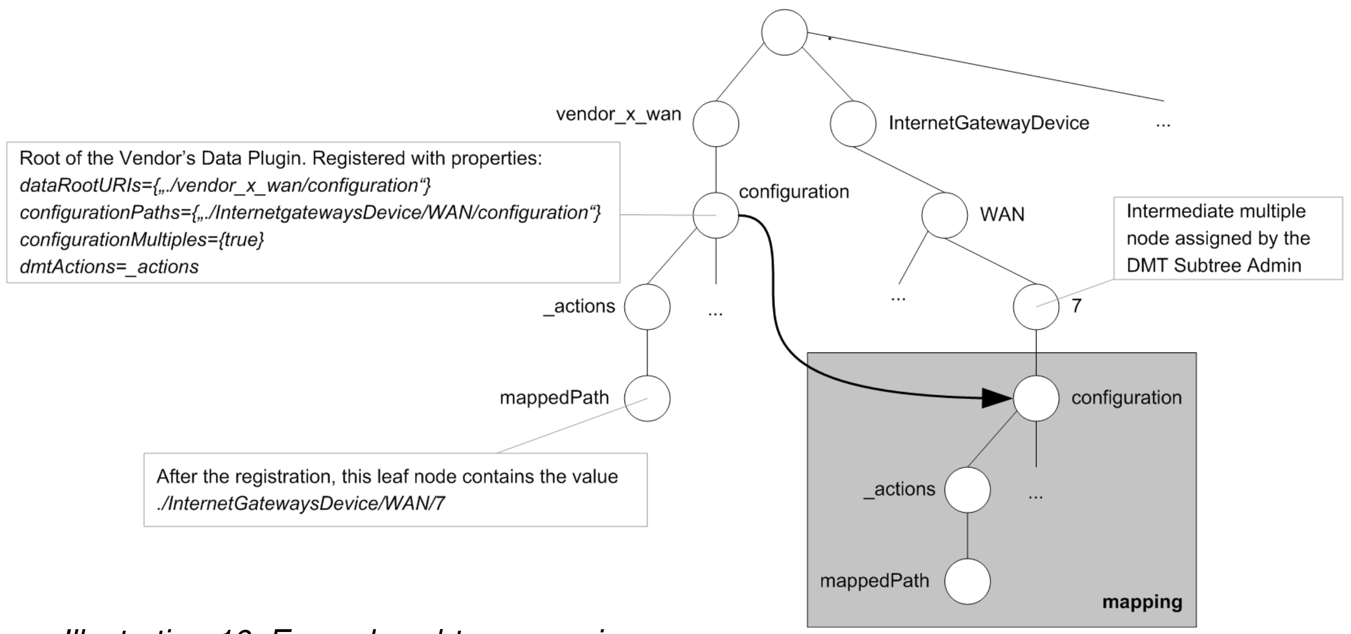

vendor_x_wan
InternetGatewayDevice
...
Root of the Vendor's Data Plugin. Registered with properties:
dataRootURIs={„./vendor_x_wan/configuration"}
configurationPaths={„./InternetgatewaysDevice/WAN/configuration"}
configurationMultiples={true}
dmtActions=_actions
configuration
WAN
Intermediate multiple node assigned by the DMT Subtree Admin
_actions
...
7
mappedPath
configuration
After the registration, this leaf node contains the value ./InternetGatewaysDevice/WAN/7
_actions
mappedPath
mapping

Illustration 10: Example subtree mapping

5.4.5 Additional expected behavior when using DMT Subtree Admin

The following rules apply when implementing and using the DMT Subtree Admin.

- Since the DMT Subtree Admin needs to open sessions to other Data Plugins, some restrictions for accessing the DMT Tree apply. There must be no session that accesses the root of the DMT in a way that acquire a lock in any way. Otherwise the DMT Subtree Admin is not able to get sessions to subtrees via other Data Plugins.

- The DMT Admin specification does not specify the minimum length of a node name and leaves this up to the implementation to decide. A domain that defines a data model and wants this to be managed by the DMT Admin must specify the minimum node name length the DMT Admin must support.

- The DMT Admin specification does not specify the length for the timeout for a session. Since heavy configuration and management activities can take some time, especially when done via a remote management system, the domain that defines the actual integration environment must define this value for the DMT Admin to be of appropriate length.

- A Vendor Data Plugin can be mapped into the DMT Tree under any path, even if between the root node and the mapping point no „real" nodes exists. Therefore, the DMT Subtree Admin must emulate the missing nodes in between. These nodes can be accessed only for traversing. Any attempt to write to such a node must result in an DmtException with the case PERMISSION_DENIED.

- When a transactional session is closed, committed, or rolled back the DMT Subtree Admin must call the appropriate actions on all the DataSessions it opened during the session. The order in which the sessions are handled is defined as follows: The DMT Subtree Admin must check which one of the involved sessions handled the longest path (the deepest path from the root node in regard to number of path elements) and close/commit/rollback this session first. If there is more than one session for which this criteria is true (same number of path elements from the root node) then the session that has been opened last must be closed/committed/rolled back first.

- If a Vendor Data Plugin is registered with the *configurationMultiples* set to „true“ the actual path in the mapped configuration tree is determined by the DMT Subtree Admin. By default, there is no easy mean for a Vendor Data Plugin to get this path. This is, however, essential if the data model that is implemented by the Vendor Data Plugin itself contains tables or something similar that is also needed to be registered in a multiple fashion. The Vendor Data Plugin can manage this on its own, but it would be much simpler if the DMT Subtree Admin could manage these multiple parts, too.
  To solve this problem the Vendor Data Plugin can make use of the *dmtActions* property. When the DMT Subtree Admin does the mapping it creates the leaf node *mappedPath<n>* in the Vendor Data Plugin. The Vendor Data Plugin can now determine the real path and start to register the multiple subtrees below that path.

5.4.6 Raising Events

As described above the DMT Admin only raises events for nodes and values that are changed through a DMT Session object. Values that change on the level of a Data Plugin (e.g. an internal value is changed by external means) are not evented. The following steps specify the procedure that must be implemented for the DMT Subtree Admin to raise events.

The Vendor Driver must raise events as follows:

- If a value in a driver has changed by any other means than through its associated Data Plugin, an event can be raised. It is up to the underlying management model of the driver bundle to decide only to raise events for some values.

- The events raised must use the Event Admin and must conform to the description of section 117.10.1, Event Admin based Events [11]..

- Only events with the following event topics must be raised. Any other event topic is not allowed to be raised by a vendor driver.

  - *info/dmtree/DmtEvent/ADDED:* New nodes were added.[2]

  - *info/dmtree/DmtEvent/DELETED*: Existing nodes were removed.

  - *info/dmtree/DmtEvent/REPLACED:* Existing node values or other properties were changed.

- No session.id property must be set. This distinguishes an event that is raised by the vendor driver from one that is raised by the DMT Admin..

- The *nodes* property contains the URI of the affected nodes. The URI path must start with the same value the Vendors Driver Data Plugin was registered wi th (the *dataRootURIs* property).

- The *newnodes* property is not set.

The DMT Subtree Admin must handle events as follows:

- The DMT Subtree Admin is catching all DMT Admin events with the topic of either "*info/dmtree/DmtEvent/ADDED", "info/dmtree/DmtEvent/DELETED", or*

2 An example for an ADDED event could be when a DHCP server in the IGD registers a new client's MAC address.

"info/dmtree/DmtEvent/REPLACED~~.~~ that originate from any Vendor Driver Data Plugin, indicated by the missing *session.id* property. If an event originates from a vendor's driver, then a new DMT Admin event is raised for each configured configurationPath of matching plugins, where the creation is following these rules:

- A new Event object is created with the same topic as the original event and no *session.id* property.
- The values of the *nodes* property are mapped to their ~~virtual~~ counterparts in the according mapped ~~subtree of the~~ virtual subtree ~~DMT Subtree Admin~~. The following rules apply here:
  - If the *configurationMultiples* property for the event-originating Data Plugin is false:
    - The prefix of the node value that matches the value of *dataRootURIs* is removed. Take R as the remaining value.
    - Define S as the ~~longest *configurationPaths* element of the according~~ *mappedPath* ~~mapped~~ of the vendor DataPlugin matching the original event's *node* property.
    - The new value is constructed as follows:
      <value of S > + <remaining value R>
  - If the configurationMultiples property for the originated Data Plugin is true:
    - The prefix of the node value that matches the value of *dataRootURIs* is removed. The remaining value is R.
    - Define S as the ~~longest *configurationPaths* element of the according~~ mappedPath of the vendor DataPlugin matching the original event's *node* property. NOTE: The mappedPath of multiple plugin's already includes the uniquely assigned identifier for that Data Plugin.
    - The new value is constructed as follows:
      <value of S > + ~~"/" + <uniquely assigned identifier for that Data Plugin> +~~ <remaining value R>
- The new event is raised via the Event Admin service.

Note, that in the end two events are raised: one that is raised by the Vendor Driver and one by the DMT Subtree Admin.

Illustration 11 presents the general relationships between the involved components.

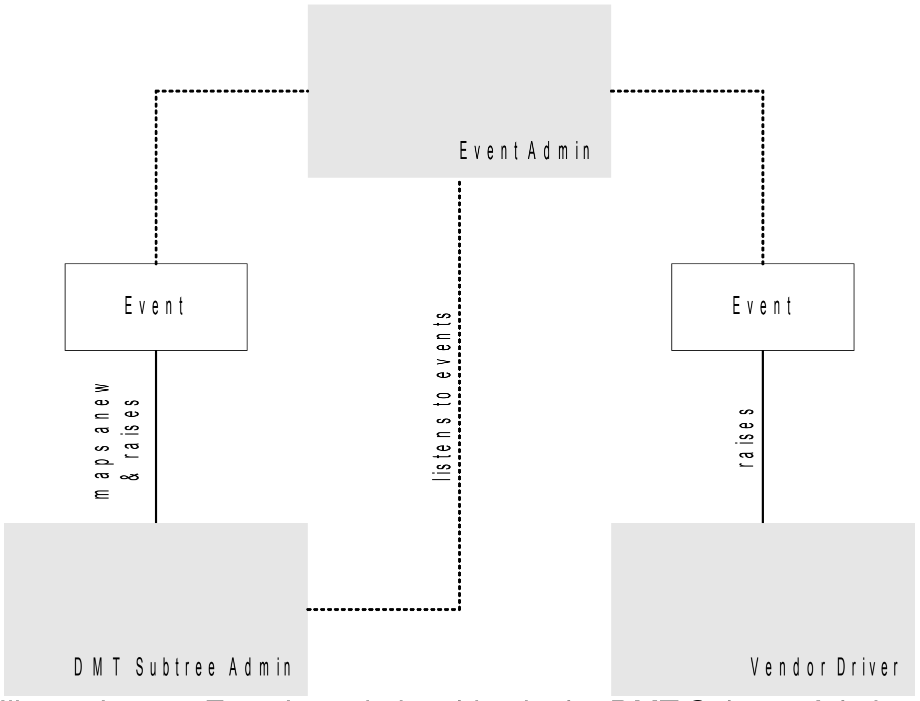


Illustration 11: Eventing relationships in the DMT Subtree Admin

5.4.7 Event Mapping Examples

The following table shows examples which reflect the mapping rules defined in the previous section. It is assumed that the Vendor Driver has been registered as shown as in the examples in section .

Node value of the vendor's driver event	Mapped node value
./vendor_x/firewall/enabled	./InternetGatewayDevice/service/firewall/enabled
./vendor_x/interface/WAN/disconnected	./InternetGatewayDevice/interface/ 1234/WAN/~~1234/~~disconnected

Table 2: Event Mapping Examples

5.5 Mapping for an Internet Gateway Device

This section specifies how a subtree in the DMT Admin is mapped for the data model for an IGD to the DMT tree and various Vendor IGD Data Plugins.

5.5.1 Registering IGD related Data Plugins

The following rules apply for mapping an IGD management tree.

- A new ServiceFactoryConfiguration *org.osgi.service.dmtsubtree* is created. The value „InternetGatewayDevice“ is assigned to the key *rootnode*. The DMT Subtree Admin creates a new Data Plugin that is responsible to manage and map the configurations under the subtree „./InternetGatewayDevice“ in the DMT Admin.

- Vendor IGD Drivers must register and map their Data Plugins via the DMT Subtree by registering the Data Plugins with the property *configurationPaths* set. The value of the property must start with „./InternetGatewaysDevice“ followed by a slash (/) and the appropriate sub-path of the Data Plugin's configuration path.

- Optionally, the Vendor IGD Data Plugins can have the property *configurationMultiples* set to *true* or *false*, depending on whether their configuration tree needs to be managed in a multiple fashion.

5.5.2 Remote Management Events and Notifications

Within the OSGi service platform all events are considered as "active". That means that any event will be distributed through the Event Admin service, and any permitted entity running in the OSGi service platform is able to receive these events.

Broadband Forum's TR-69 specification allows for "passive" notifications, which are handed over to a Remote Management System (RMS) as part of the next scheduled contact between the management agent and the RMS. For this case, the management agent is responsible for keeping track of all events and storing them persistently until the next contact. Specification of the management agent functionality is out of scope of this specification.

5.6 Mapping rules to DMT Admin Management Tree

This chapter defines mapping rules for mapping various management standards to the DMT Admin tree.

5.6.1 Mapping from Broadband Forum TR-069 and TR-098 to DMT

This specification does not define management entities of a home gateway. However, node paths in the DMT look slightly different than in the TR-069 specification. Following mapping rules apply:

- Every node path for TR-069 data models must start with "./" plus the root path of the respective data model. For TR-098 this is "*./InternetGatewayDevice*".

- Every dot (.) in the original TR-098 path must be substituted by a slash (/).

- Arrays must be represented by using the index number as a node (e. g. "./InternetGatewayDevice/WANDevice/1").

- Any TR-098 element that ends with "*numberOfEntries"* must be computed, not stored.

Like Java, TR-069 knows data types. The following table presents the mapping between TR-069 and DMT Admin data types.

TR-069 data type	DMT Admin data type	Remarks
String	FORMAT_STRING	Created by DmtData(String)
int	FORMAT_INTEGER	Created by DmtData(int)
unsignedInt	FORMAT_STRING	The unsignedInt is an unsigned integer. Since there is no suitable data type defined in Java, this data type should be mapped as String. Note, that a client that access this value needs some knowledge about the underlying data model, otherwise it has no means to detect the actual type (unsigned int) be requesting the data type.
boolean	FORMAT_BOOLEAN	Created by DmtData(boolean)
dateTime	FORMAT_STRING	The dateTime is a String object that is interpreted as an *dateTime* type defined in ISO 8601, which is used as a value of date and time in TR-069. Since there is no corresponding data type defined in DMT admin, this data type should be mapped as String. Note, that a client that access this value needs some knowledge about the underlying data model, otherwise it has no means to detect the actual type (dateTime) be requesting the data type.
base64	FORMAT_BASE64	Created by DmtData(byte[], boolean)

Table 3: Mapping of data types between TR-069 and DMT Admin

5.6.2 Mapping from UPnP IGD 1.0

The TR-098 data model is a superset of the UPnP IGD 1.0 specification, so all rules that apply to TR-098 apply also to the mapping of UPnP IGD 1.0.

5.6.3 Examples

The following table presents an example mapping between TR-098, UPnP IGD 1.0, and the DMT Admin tree.

TR-098	UPnP IGD 1.0	DMT Admin tree
InternetGatewayDevice.Layer3Forwarding.	<<service>> Layer3Forwarding	./InternetGatewayDevice/Layer3Forwarding
DefaultConnectionService	DefaultConnectionService	as TR-98
ForwardNumberOfEntries	not needed	computed

Table 4: Example mapping between TR-098, UPnP IGD, DMT Admin Tree

6 Considered Alternatives

6.1 Not using DMT Admin

DMT Admin does not meet all important requirements regarding the management of a Home Gateway, so it would be tempting not to use it rather than specifying a service that meets the requirements 100 percent. But specifying another service which replicates DMT Admin functionality by 80% and adds the 20% needed for IGD management would be inefficient as well.

An implementation that would mostly replicate most of the functionality of the DMT Admin would more burdensome than the proposed solution. This solution has the advantage that it enhances the DMT Admin without changing its specification.

7 Security Considerations

Read and write access to the following services, objects and entities must be restricted to authorized bundles and services only:

- Any access to new subtrees, especially the "./ *InternetGatewayDevice*" subtree of the DMT Admin tree. Access to this subtree must be handled by using Dmt Principal Permission, DMT Permission and the appropriate permission actions. Access to the subtree must be disabled by default.
- Every vendor-specific subtree a vendor driver uses to add his own configuration tree to the DMT Admin. Access to the vendor subtrees must be handled by using Dmt Principal Permission and the appropriate permission actions. Access to the sub-tree must be disabled by default. Only the DMT Subtree Admin bundle should have access to the values of the vendor's subtree.
- Data Plugins of the DMT Subtree Admin. By using the OSGi Permission Admin service [12]., no other than the DMT Admin is allowed to access methods of the DMT Subtree Admin's Data Plugins.

- Data Plugins of vendor-specific IGD drivers. By using the OSGi Permission Admin service [12]., no other than the DMT Admin is allowed to access methods of the vendors' Data Plugins.

Detailed information on security of the DMT Admin Service specification can be found at [10].

Events raised by the DMT Subtree Admin or vendor-specific drivers can be received by any bund le, which has enough TopicPermission to receive DMT Events..

8 RFC closed

After discussing in the OSGi Residential Expert Group the amount of efforts to add and specify a new service admin for OSGi, which would basically add new functionality to the DMT Admin, it was decided by the REG to abandon the idea to create a new service specification and instead to amend the DMT Admin specification itself.

The requirements derived from RFP 113 and the ide as presented in this RFC 145 (basically: overlapping trees, mounting of Data Plugins somewhere in the tree, the support for multiple nodes , and support for sending events from Data Plugins) are still valid. They will from now on further discussed and specified in RFC 141 (DMT Admin Extensions). This will lead to a new version of the DMT Admin specification itself.

The comments added to the latest version of this RFC 145 are not removed, since they reflect the status of the discussion as of the time this RFC 145 is closed.

9 Document Support

9.1 References

[1]. Bradner, S., Key words for use in RFCs to Indicate Requirement Levels, RFC2119, March 1997.

[2]. Software Requirements & Specifications. Michael Jackson. ISBN 0-201-87712-0

[3]. Internet Gateway Device (IGD) Standardized Device Control Protocol V 1.0, UPnP Forum

[4]. TR-064, Broadband Forum LAN-Side DSL CPE Configuration, BBF, May 2004

[5]. TR-069, Broadband Forum CPE WAN Management Protocol, BBF, May 2004

[6]. TR-098, Internet Gateway Device Data Model for TR-069, Amendment 1, BBF, November 2006

[7]. TR-104, Provisioning Parameters for VoIP CPE, BBF, September 2005

[8]. Home Gateway Technical Requirements: Residential Profile, Version 1.0, HGI, April 2008

[9]. OSGi Companion Specification, DMT Admin, Events, Section 117.10

[10]. OSGi Companion Specification, DMT Admin, Security, Secion 117.12

[11]. OSGi Companion Specification, Event Admin Service Specification, Chapter 113

[12]. OSGi Service Platform Core Specification, Permission Admin Service, Chapter 10

9.2 Author's Address

Name	Vivien Helmut, Andreas Kraft, Andreas Sayegh
Company	Deutsche Telekom AG, Laboratories
Address	Ernst-Reuter-Platz 7, D-10587 Berlin
Voice	+49 30 8353-58185
e-mail	a.kraft@telekom.de

9.3 Acronyms and Abbreviations

B2BUA	Back-to-back User Agent
BBF	Broadband Forum
CPE	Customer-Premises Equipment
DMT	Device Management Tree
GUI	Graphical User Interface
HG	Home Gateway
HGI	Home Gateway Initiative
IGD	Internet Gateway Device
ISP	Internet Service Provider
LAN	Local Area Network
NAT	Network Address Translation

PAT	Port Address Translation
RMS	Remote Management System
RTSP	Real-Time Streaming Protocol
SSID	Service Set Identifier
VoD	Video on Demand
VoIP	Voice over IP
WAN	Wide Area Network
Wi-Fi	Wireless Fidelity

9.4 End of Document