DISTRIBUTION AND FEEDBACK LICENSE, Version 2.0

DATE OF DISTRIBUTION: 6 April 2010

The OSGi Alliance hereby grants you a limited copyright license to copy and display this document (the “Distribution”) in any medium without fee or royalty. This Distribution license is exclusively for the purpose of reviewing and providing feedback to the OSGi Alliance. You agree not to modify the Distribution in any way and further agree to not participate in any way in the making of derivative works thereof, other than as a necessary result of reviewing and providing feedback to the Distribution. You also agree to cause this notice, along with the accompanying consent, to be included on all copies (or portions thereof) of the Distribution. The OSGi Alliance also grants you a perpetual, non-exclusive, worldwide, fully paid-up, royalty free, limited license (without the right to sublicense) under any applicable copyrights, to create and/or distribute an implementation of the Distribution that: (i) fully implements the Distribution including all its required interfaces and functionality; (ii) does not modify, subset, superset or otherwise extend the OSGi Name Space, or include any public or protected packages, classes, Java interfaces, fields or methods within the OSGi Name Space other than those required and authorized by the Distribution. An implementation that does not satisfy limitations (i)-(ii) is not considered an implementation of the Distribution, does not receive the benefits of this license, and must not be described as an implementation of the Distribution. “OSGi Name Space” shall mean the public class or interface declarations whose names begin with “org.osgi” or any recognized successors or replacements thereof. The OSGi Alliance expressly reserves all rights not granted pursuant to these limited copyright licenses including termination of the license at will at any time.

EXCEPT FOR THE LIMITED COPYRIGHT LICENSES GRANTED ABOVE, THE OSGi ALLIANCE DOES NOT GRANT, EITHER EXPRESSLY OR IMPLIEDLY, A LICENSE TO ANY INTELLECTUAL PROPERTY IT, OR ANY THIRD PARTIES, OWN OR CONTROL. Title to the copyright in the Distribution will at all times remain with the OSGi Alliance. The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted therein are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

THE DISTRIBUTION IS PROVIDED "AS IS," AND THE OSGi ALLIANCE (INCLUDING ANY THIRD PARTIES THAT HAVE CONTRIBUTED TO THE DISTRIBUTION) MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE DISTRIBUTION ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

NEITHER THE OSGi ALLIANCE NOR ANY THIRD PARTY WILL BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THE DISTRIBUTION.

Implementation of certain elements of this Distribution may be subject to third party intellectual property rights, including without limitation, patent rights (such a third party may or may not be a member of the OSGi Alliance). The OSGi Alliance is not responsible and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights.

The Distribution is a draft. As a result, the final product may change substantially by the time of final publication, and you are cautioned against relying on the content of this Distribution. You are encouraged to update any implementation of the Distribution if and when such Distribution becomes a final specification.

The OSGi Alliance is willing to receive input, suggestions and other feedback (“Feedback”) on the Distribution. By providing such Feedback to the OSGi Alliance, you grant to the OSGi Alliance and all its Members a non-exclusive, non-transferable, worldwide, perpetual, irrevocable, royalty-free copyright license to copy, publish, license, modify, sublicense or otherwise distribute and exploit your Feedback for any purpose. Likewise, if incorporation of your Feedback would cause an implementation of the Distribution, including as it may be modified, amended, or published at any point in the future (“Future Specification”), to necessarily infringe a patent or patent application that you own or control, you hereby commit to grant to all implementers of such Distribution or Future Specification an irrevocable, worldwide, sublicenseable, royalty free license
under such patent or patent application to make, have made, use, sell, offer for sale, import and export products or services that implement such Distribution or Future Specification. You warrant that (a) to the best of your knowledge you have the right to provide this Feedback, and if you are providing Feedback on behalf of a company, you have the rights to provide Feedback on behalf of your company; (b) the Feedback is not confidential to you and does not violate the copyright or trade secret interests of another; and (c) to the best of your knowledge, use of the Feedback would not cause an implementation of the Distribution or a Future Specification to necessarily infringe any third-party patent or patent application known to you. You also acknowledge that the OSGi Alliance is not required to incorporate your Feedback into any version of the Distribution or a Future Specification.

I HEREBY ACKNOWLEDGE AND AGREE TO THE TERMS AND CONDITIONS DELINEATED ABOVE.
Preface

This document is Early Draft 3 of the OSGi Service Platform Release 4 Residential Version 4.2 specifications. As an early draft, it contains non-final specification work and it is not organized in the format normally associated with final release OSGi specifications. This document contains copies of OSGi design documents which either propose to modify existing published OSGi specifications from the OSGi Service Platform Release 4 Version 4.2 specification documents or propose new specifications to potentially be incorporated in the final OSGi Service Platform Release 4 Residential Specification Version 4.2 documents.

Since this early draft is not a complete specification document, the reader is expected to be familiar with OSGi Technology and the currently published OSGi Service Platform Release 4 Version 4.2 specification documents. The reader should refer to http://www.osgi.org/About/Technology for more information on the OSGi Technology. There the reader can find a description of the OSGi Technology, as well as links to whitepapers and the OSGi Service Platform Release 4 Version 4.2 specification documents, which are all available for download.

In an effort to make this early draft available as quickly as possible, it contains OSGi design documents (“RFCs”). These documents have been declassified by the OSGi Alliance so that they may be made available in this early draft. This early draft contains a majority of the design documents the OSGi expert groups currently anticipate will be incorporated into the final specification documents.

Pursuant to the Distribution and Feedback License above, the OSGi expert groups welcome your feedback on this early draft. Feedback can be provided by opening a bug at https://www.osgi.org/bugzilla/enter_bug.cgi?product=OSGi%20Specification .

BJ Hargrave
Chief Technical Officer
OSGi Alliance
Residential Design Documents

OSGi Service Platform Release 4
Version 4.2 – Residential Early Draft 3

Revision 1.2
6 April 2010
RFC-140 Residential Management Tree

Draft

41 Pages

Abstract

Different industries are interested in the application of OSGi for their business, in which remote management is a key issue. Telecom operators (both fixed and mobile ones), server managers, and automotive manufactures, etc. need solutions to remotely manage their instances of OSGi frameworks. The main problem with that need is that is very difficult to solve with a single solution, taking into account that the management protocols are many and different for the different industries. To achieve this goal, a management object model should be defined to expose OSGi framework manageable information through a management agent.
1 Document Information

1.1 Table of Contents

1 Document Information................................................................................................................2
   1.1 Table of Contents................................................................................................................2
   1.2 Terminology and Document Conventions...........................................................................3
   1.3 Revision History..................................................................................................................3

2 Introduction..................................................................................................................................6

3 Application Domain.....................................................................................................................6

4 Problem Description....................................................................................................................7
   4.1 Management agent making use of OSGi standardized Java interfaces..............................7
   4.2 Mobile specification approach...............................................................................................8
   4.2.1 Support for TR-069 notifications.......................................................................................8
   4.2.2 Limitations in the number of services available in the DMT............................................9
   4.2.3 Mapping TR-069 to the OMA-DM inspired DMT model................................................10
   4.2.4 Conclusion.......................................................................................................................10

5 Requirements...............................................................................................................................10

6 Technical Solution......................................................................................................................11
   6.1 Legend...................................................................................................................................13
   6.2 Framework Object...............................................................................................................14
   6.3 BundleState Object.............................................................................................................20
   6.4 PackageState Object............................................................................................................25
   6.5 ServiceState Object.............................................................................................................27
   6.6 Filters Object.......................................................................................................................30
   6.7 BundleResources Object.....................................................................................................36
   6.8 Configuration Object..........................................................................................................37
   6.9 Policy Object......................................................................................................................37
   6.10 Log Object.........................................................................................................................37
   6.11 Limitations........................................................................................................................38

7 Considered Alternatives..............................................................................................................39

8 Security Considerations.............................................................................................................40

9 Document Support.......................................................................................................................40
   9.1 References..........................................................................................................................40
   9.2 Author's Address..................................................................................................................41
   9.3 Acronyms and Abbreviations..............................................................................................41
1.2 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.

Source code is shown in this typeface.

1.3 Revision History

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

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>Sep. 12 2008</td>
<td>Initial Draft</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Koya Mori, NTT Corporation, <a href="mailto:mori.kouya@lab.ntt.co.jp">mori.kouya@lab.ntt.co.jp</a></td>
</tr>
<tr>
<td>0.2</td>
<td>Oct. 9 2008</td>
<td>2nd Draft</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Overall architecture is changed based on REG meeting and RFC-139 definition.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Koya Mori, NTT Corporation, <a href="mailto:mori.kouya@lab.ntt.co.jp">mori.kouya@lab.ntt.co.jp</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ikuo Yamasaki, NTT Corporation, <a href="mailto:yamasaki.ikuo@lab.ntt.co.jp">yamasaki.ikuo@lab.ntt.co.jp</a></td>
</tr>
<tr>
<td>0.3</td>
<td>Nov. 10 2008</td>
<td>3rd Draft</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Definition tables for all nodes in Management Objects are added.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Koya Mori, NTT Corporation, <a href="mailto:mori.kouya@lab.ntt.co.jp">mori.kouya@lab.ntt.co.jp</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ikuo Yamasaki, NTT Corporation, <a href="mailto:yamasaki.ikuo@lab.ntt.co.jp">yamasaki.ikuo@lab.ntt.co.jp</a></td>
</tr>
<tr>
<td>0.4</td>
<td>Jan. 19 2009</td>
<td>4th Draft</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Revising tree architecture based on the discussion at the REG meeting.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Filter format description is added.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Koya Mori, NTT Corporation, <a href="mailto:mori.kouya@lab.ntt.co.jp">mori.kouya@lab.ntt.co.jp</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ikuo Yamasaki, NTT Corporation, <a href="mailto:yamasaki.ikuo@lab.ntt.co.jp">yamasaki.ikuo@lab.ntt.co.jp</a></td>
</tr>
<tr>
<td>0.5</td>
<td>Feb. 12 2009</td>
<td>5th Draft</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Revising Bundles Object based on discussion at the REG meeting.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Koya Mori, NTT Corporation, <a href="mailto:mori.kouya@lab.ntt.co.jp">mori.kouya@lab.ntt.co.jp</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ikuo Yamasaki, NTT Corporation, <a href="mailto:yamasaki.ikuo@lab.ntt.co.jp">yamasaki.ikuo@lab.ntt.co.jp</a></td>
</tr>
<tr>
<td>Revision</td>
<td>Date</td>
<td>Comments</td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| 0.6      | Mar. 26 2009 | 6th Draft  
Revising Filters Object specification based on discussion and analysis.  
Koya Mori, NTT Corporation, mori.kouya@lab.ntt.co.jp  
Ikuo Yamasaki, NTT Corporation, yamasaki.ikuo@lab.ntt.co.jp |
| 0.7      | May. 14 2009 | 7th Draft  
Revising Resources sub-tree and Filters Object specification based on discussion at the REG meeting.  
Koya Mori, NTT Corporation, mori.kouya@lab.ntt.co.jp  
Ikuo Yamasaki, NTT Corporation, yamasaki.ikuo@lab.ntt.co.jp  
Shigekuni Kondo, NTT Corporation, kondo.shigekuni@lab.ntt.co.jp |
| 0.8      | Jun. 04 2009 | 8th Draft  
Revising wording and ServiceState Object specification based on discussion at the REG meeting.  
Koya Mori, NTT Corporation, mori.kouya@lab.ntt.co.jp  
Ikuo Yamasaki, NTT Corporation, yamasaki.ikuo@lab.ntt.co.jp  
Shigekuni Kondo, NTT Corporation, kondo.shigekuni@lab.ntt.co.jp |
| 1.0      | Sep. 18 2009 | 9th Draft  
Bundle StartLevel configuration is moved to the Framework Object.  
Koya Mori, NTT Corporation, mori.kouya@lab.ntt.co.jp  
Ikuo Yamasaki, NTT Corporation, yamasaki.ikuo@lab.ntt.co.jp  
Shigekuni Kondo, NTT Corporation, kondo.shigekuni@lab.ntt.co.jp |
| 1.1      | Oct. 16 2009 | 10th Draft  
Lifecycle sub-tree of Bundle is moved to the Framework Object.  
Revising BundleState Object specification and BundleResources Object specification based on discussion at the REG meeting.  
Koya Mori, NTT Corporation, mori.kouya@lab.ntt.co.jp  
Ikuo Yamasaki, NTT Corporation, yamasaki.ikuo@lab.ntt.co.jp  
Shigekuni Kondo, NTT Corporation, kondo.shigekuni@lab.ntt.co.jp |
<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Comments</th>
<th>Authors</th>
</tr>
</thead>
</table>
| 1.2      | Jan. 20 2010 | 11th Draft  
Revising example of the Filter and TargetSubtree (Case1 and Case4).  
Declaring Residential management tree Data plug-in root path.  
Koya Mori, NTT Corporation, mori.kouya@lab.ntt.co.jp  
Ikuo Yamasaki, NTT Corporation, yamasaki.ikuo@lab.ntt.co.jp  
Shigekuni Kondo, NTT Corporation, kondo.shigekuni@lab.ntt.co.jp |         |
| 1.3      | Feb. 19 2010 | 12th Draft  
Revising BundleState Object and PackageState Object, ServiceState Object.  
Adding NumberOfEntries nodes below the $/OSGi/<instance-id>/.  
Moving Filters Object and Local node to below the $/OSGi/<instance-id>/.  
Koya Mori, NTT Corporation, mori.kouya@lab.ntt.co.jp  
Ikuo Yamasaki, NTT Corporation, yamasaki.ikuo@lab.ntt.co.jp  
Shigekuni Kondo, NTT Corporation, kondo.shigekuni@lab.ntt.co.jp |         |
| 1.4      | Apr. 1 2010  | 13th Draft  
Revising Certificate information format section and Service Property Dictionary nodes section based on discussion at the REG meeting.  
Koya Mori, NTT Corporation, mori.kouya@lab.ntt.co.jp  
Ikuo Yamasaki, NTT Corporation, yamasaki.ikuo@lab.ntt.co.jp  
Shigekuni Kondo, NTT Corporation, kondo.shigekuni@lab.ntt.co.jp |         |
2 Introduction

Traditionally, fixed telecommunication operators don’t have knowledge about what runs in the customer’s local area network (LAN). They provide connectivity and manage the wide area network (WAN) that provides this connectivity, but they do not know anything about the devices and networks behind the gateway (xDSL mainly) that interconnect WAN and LAN. Recently the need for management of customer networks and devices is increasing in order to make the deployment of new complex services at home (home automation, tele-health, VoIP, IPTV, surveillance, etc) feasible with reasonable costs. For example to avoid sending technicians to the customer premises for solving problems.

There are two main kinds of devices that need to be managed remotely in operator’s business: those which come from the fixed line business managed via the TR-069 protocol, that is standardized by the Broadband FORUM [6], and those which come from the mobile business managed by OMA DM, which is standardized by the Open Mobile Alliance. Up to now, the OSGi specifications cover the OMA DM ones, but lacks of specification about how an OSGi framework on a device could be managed by TR-069.

One key question is: why do we need two technologies to manage devices in a converged scenario in which the two kinds of devices are going to interact between them? The question has a better answer from the business point of view than from a technical point of view. Fixed and mobile businesses have evolved independently during the last years. Both technologies have acquired enough critical mass not only for having success in product implementation and roll-out, but also to be a de facto standard in their respective applications domains.

Despite the fact that the best solution would be a single protocol to manage all kind of devices, we are aware that at least for the short-medium term both technologies must coexist. For the future, the model shouldn’t be closed to add new solutions, for example a mixed model that unifies both worlds.

3 Application Domain

Driven by triple play service delivery in the home network, fixed line access service providers have the need to configure home devices to ensure the proper service delivery. Broadband Forum’s CPE WAN Management Protocol (CWMP, alias TR-069) enables them to do this. By using a remote management server (Auto Configuration Server, ACS), they are able to manage TR-069 enabled devices. TR-069 provides them with possibilities to configure parameters, be informed of parameter changes (notification), start diagnostic tools, update firmware, etc.

Similarly, for the mobile world, the OMA defined the OMA-Device Management specification for remote management of mobile devices. OMA-DM offers similar tools to the mobile service providers as TR-069 to fixed line service provider, but OMA-DM is of course tailored to the specifics of the mobile environment.

As OSGi technology offers a flexible software environment for all these different devices, the remote management of the platform is of interest for both fixed and mobile service providers. As such, it should be possible to integrate
the remote management of the OSGi platform, and the applications running on top of it, in the existing management infrastructure.

The DMT Admin service with its mobile management tree in the Mobile specification for OSGi R4 standardizes the remote management of an OSGi platform. As it is largely inspired by OMA-DM, it needs to be evaluated for multi protocol support.

# 4 Problem Description

In a scenario in which service providers offer a growing number of services, to use specific solutions for the management of those services is not the most suitable option. To speed up the deployment of these services, such as triple play, home automation or tele-health, it is essential to offer general management solutions that allow for the management of a large number of services and the flexible life-cycle management of applications.

These devices usually are already managed by a standard protocol, so it makes sense that an OSGi framework, which hosts the services, running on a device could be managed in the same way as the other resources of the device. Of course, the remote management should be fully integrated in the existing remote management solutions of the service provider to avoid duplicating management infrastructure and to increase performance on the devices.

Currently, there are two options in OSGi for remote management:

- create a management agent bundle making use of the Java object interfaces,
- create a protocol adapter bundle that interacts with the DMT Admin service, as defined in the OSGi Mobile specification.

## 4.1 Management agent making use of OSGi standardized Java interfaces

Currently, for the management of a bundle, the OSGi specifications define different Java objects with which a management application can interact. Using this approach, a management agent can implement extensive management of the OSGi framework, as well as any service standardized. Mapping the Java interfaces to the specific remote management protocol and data model tree is up to the management agent.

For runtime interaction with a bundle, a bundle can register a service in the service registry. However, this service interface is not standardized. Also, mapping the service interface to a general management model is not standardized. A current approach is to implement a proprietary service interface on all bundles to be managed. By tailoring this interface so that it easily maps to the management protocol primitives, it is simple for the management agent to map remote management commands to the bundle’s service interface. The disadvantage is the proprietary service interface, so that 3rd party bundles might not be compliant.

As a conclusion we can say that this current approach allows for extensive remote management of any aspect of the OSGi platform, but lacks a standardized service interface definition for bundles to implement.
4.2 Mobile specification approach

The Mobile Expert Group has provided its own solution based on the OMA [3] Device Management [4] specification to provide a remote management solution. The OSGi Mobile specification contains two chapters related to remote management:

- chapter 3: detailing the mobile management tree
- chapter 117: detailing: the DMT Admin service, bundle plugin interface specification, the notification service

The Device Management Tree model of OMA-DM was chosen as meta-data model and operational model. However, it was intended to be mappable to other protocols.

An analysis of mapping the Mobile specification DMT model to TR-069, however, shows that the current DMT model approach (as defined in the OSGi R4 Mobile Specification) introduces some issues. For example:

- Limitations for active or passive notifications on any parameter in the object tree
- A limited number of services have been mapped to the DMT model
- The complexity of mapping a new protocol to the OMA-inspired DMT model, which could imply performance issues on limited devices.

4.2.1 Support for TR-069 notifications

TR-069 offers the feature of active and passive notifications. By setting a parameter's notification attribute, a remote manager requests to be notified with the parameter's new value at the time the value changes (active notification) or at the next periodic inform (passive notification). Notification can be configured on any parameter of the TR-069 object tree. This approach enables the remote manager to be informed not only of changes in status variables of the platform, but also of configuration changes performed by a local manager, e.g. through a local Web interface.

The Mobile specification offers a few features that could help to implement TR-069 notification support:

- The DMT Admin service sends events using the Event Admin service when operations have been performed on nodes (nodes added, removed or copied; node values changed etc.)
- The OSGi Notification service defines a way to alert a remote management server. Protocol adapters on their turn have to implement a RemoteAlertSender interface (and register it) for use by the notification service. Notifications are sent by calling sendNotification on the notification service:

- The Monitor Admin service: A bundle can register a Monitorable service, to be used by the Monitor Admin service. By registering a Monitorable service, the bundle exposes access to a number of status variables. Notification can be implemented by the StatusVariable provider. If it does, it will call the update method on the Monitor Listener. The Monitor Admin service then generates an event on the Event Admin service. The Monitor service is currently also represented in the DMT tree.

Two problems arise when trying to map the current approach to TR-069:

- TR-069 defines that notification is applicable to any parameter in the object tree.

Currently, the DMT Admin service only send events for operations on DMT nodes that were performed using the DMT Admin API. For example: if configuration changes are performed by
using the Configuration Admin service API, no events will be sent. Most of the current implementations do not perform all changes via the DMT Admin service. Therefore, the events sent by the DMT Admin service are an only subset and thus not very reliable as single source of events (and thus as single source of TR-069 notifications).

The OSGi Monitor service only supports notification of changes on Status Variables, exposed through a Monitorable service, and enabled by the bundle to support on-change notification (i.e. dynamic Status Variables).

- Requesting notification is not fully under the control of the remote manager. In the case of a bundle using the notification service, there is no standardized way to configure the bundle to send alerts when the value of one of the implemented DMT nodes changes. In the case of the monitor service, the sending of events can be controlled, but is limited to dynamic Status Variables.

The current DMT Admin service has no attributes properties on its nodes to be used to configure notification behavior, such as active notification and passive notification defined in TR-069. Therefore, a remote manager cannot control the notification behavior of DMT nodes in a standardized way.

To conclude, the current options, as provided in the Mobile specification, limit notification of parameter changes to Status Variables, explicitly enabled for monitoring. There is no standardized approach available to monitor changes on any node in the DMT.

4.2.2 Limitations in the number of services available in the DMT

The OSGi R4 Mobile specification mapped a number of services to the DMT tree. However, these services are limited to the services listed in the Mobile Specification. Other interesting services, as listed in the OSGi R4 Service Compendium are not yet mapped (standardized) in the DMT tree:

The DMT tree defined in the Mobile Specification contains objects for the following services:

- Configuration Admin service
- Log service
- Monitor Admin service
- Application Admin service
- Conditional Permission Admin service
- Deployment service

A number of areas that could be of interest to a remote manager are currently missing in the DMT tree:

- Startlevel management
- Bundle management: managing individual bundles as opposed to deployment packages (inventory, life-cycle management, exported services, …)
- Service management: getting a remote view on services registered in the service registry
- Permission Admin management
4.2.3 Mapping TR-069 to the OMA-DM inspired DMT model

Within the OSGi Mobile specification, the choice has been made to model the DMT after OMA-DM.

As a result, creating an OMA-DM protocol adapter is quite straightforward. Although no major hurdles have been identified in creating a TR-069 protocol adapter, it is less straightforward:

- The TR-069 RPC primitives have to be translated to the DMT Admin service interface methods (which are OMA-DM RPC inspired).

- The TR-069 tree has to be mapped to the DMT tree. Translating object model specific features like DMT meta nodes, or TR-069 attributes is not straightforward. It might require specific extensions to the DMT, e.g. to support TR-069 attributes, or a single node in the DMT might result in multiple objects in a TR-069 data model, etc.

- The TR-069 data types have to be mapped to the DMT Admin data types. However, TR-069 data types, such as “unsignedint” and “dateTime” (ISO 8601), cannot be translated appropriately into DMT Admin data types defined in the current specification. Translating these data types might result a limitation of the available value range and a complex object that consists of multiple nodes, respectively.

4.3 Conclusion

The OSGi Mobile specification delivers a standardized data model (the DMT), and standardized interface (on the DMT Admin service) to enable remote management through a protocol adapter. However, the current specification lacks management objects for a number of interesting areas. Also, there is some support lacking for TR-069 notifications. Furthermore, since the DMT model is OMA-DM inspired, implementing a TR-069 protocol adapter is not straightforward, although not impossible.

5 Requirements

REQUIREMENT[1]: A management tree, which is mappable to multiple remote management protocols, MUST be standardized. The solution MUST be mappable at least to OMA DM and TR-069 protocols. The model MUST be open to add new protocols in the future, like a possible common solution to substitute OMA-DM and TR-069.

REQUIREMENT[2]: A bundle MAY implement a non-standard sub-tree of the management tree. The solution MUST support the management of this type of sub-trees. As such, it MUST define the interface to be implemented by the bundle to support this management.

REQUIREMENT[3]: The management tree SHOULD cover bundle life cycle management and service monitoring.
REQUIREMENT[4]: The management tree SHOULD cover all services defined in the release 4 of the OSGi specifications. If prioritization is needed, at least the following services MUST be covered: Start Level, Permission Admin, Conditional Permission Admin, Configuration Admin and Log Service.

REQUIREMENT[5]: Support for notification of parameter value changes is required for both framework and services sub-trees, as well as for bundles implementing a sub-tree of the management tree. Some lightweight mechanism MUST allow identifying which parameters have changed and have not already been notified.

REQUIREMENT[7]: The solution MUST have a good performance in order to run on devices with limited resources.

REQUIREMENT[8]: The solution SHOULD specify a guideline of RPC mapping between the DMT Admin service interfaces and remote management protocols, such as TR-069.

REQUIREMENT[10]: The solution SHOULD enable bundles running on an OSGi framework to manage Internet Gateway functions of a Home Gateway by using an integrated management tree.

REQUIREMENT[11]: The solution SHOULD enable bundles to hook into Internet Gateway functions such as DHCP.

6 Technical Solution

This RFC defines the Residential Management Tree which is handled via DMT Admin service as it is available on an OSGi Residential Platform. The protocol used between the remote server and the device is not specified, but it is expected that the TR-069 protocol will be the management protocol used to manipulate this tree.

Although the top level nodes of this tree depend on the user policy, this RFC supposes that TR-106 structural requirements defined by the Broadband Forum [6] will be adopted by the Residential Management Tree architecture. The top level nodes of a tree adhering to TR-106 are depicted in Figure 1 (See TR-106 Amendment1 [7]). The partial tree, enclosed by the dashed line, is specified by TR-106, which is expected to be used with the Residential Management Tree in many circumstances in residential service domain. This RFC does not define any restrictions on the architecture enclosed in the dashed line, and the ancestor nodes of /Device/Services/OSGi node can be arbitrarily defined by users. Therefore, the parent node of the OSGi node is referred to as "$" in the following sections.

The OSGi Residential Management Tree is a relative tree. Devices can place the root of this tree anywhere in the Device Management Tree. In this RFC, this relative location in the Device Management Tree is indicated with the $ sign. The root of the OSGi tree is set in the System property that must not change during runtime:

residentialmanagement.osgi.root

The OSGi Residential Management Tree consists of a number of distinct parts as shown in Figure 1. Each of the sub-trees in the figure is explained in the following sections. Users of this tree may add a user-defined sub-tree under the $/OSGi/<instance_id> node. Moreover, there is no restriction on the use of sub-trees defined in Mobile Management Tree [8] as a user-defined sub-tree in the Residential Management Tree.
$/OSGi/<instance_id> node is used to represent an instance of the OSGi framework on a device. In most cases there is only one instance of OSGi, so only $/OSGi/1 is available. However, in some cases such as multiple OSGi Frameworks run on the residential home gateway, there exist several OSGi instances in the Residential Management Tree. In this case, the local OSGi framework on which the bundle registering Data Plugin of the Residential Management Tree is running should be identified, so that the DMT Admin implementation can access the indicated node path via appropriate way. Therefore, The leaf node, $/OSGi/<instance_id>/Local node , represents whether the OSGi Framework object is local or not as boolean value. How the Data Plugin of the Residential Management Tree on the local OSGi Framework finds and communicates with the other OSGi Frameworks is out of scope of this document.

The pairs between each OSGi Framework and the corresponding <instance_id> must be kept persistently beyond restart of the local OSGi Framework and reconnection of other OSGi Frameworks. OSGINumberOfEntries and BundleStateNumberOfEntries, ServiceStateNumberOfEntries, PackageStateNumberOfEntries are represent the number of those instances existing at the moment. When the instances are added or deleted, the value of the nodes must be incremented or decremented, respectively.

The Residential Management Tree adopts the complementary data model with the Mobile Management Tree defined in the OSGi Mobile Specification. On the one hand, some top level nodes, Configuration, Policy and Log, have definitions similar to the Mobile Specification. On the other, the Residential Management Tree has some original sub-trees: Framework, BundleState, PackageState, ServiceState ,Filters, and BundleResources. Especially, all of Framework, BundleState, PackageState, ServiceState Subtrees are designed to be compliant to TR-106 defined by Broadband Forum[7].

The Framework sub-tree is to manipulate the OSGi framework on which this management tree is implemented and to control the life cycle of installed bundles instead of the Deployment sub-tree defined in the Mobile
Specification. The BundleState sub-tree is used to derive information of individual bundles. The PackageState and ServiceState sub-trees provide information of available Packages and Services on the OSGi framework, respectively. The Filters sub-tree is used to filter information contained in a tree. The BundleResources sub-tree is used to derive resources in a bundle jar file.

Basically, the structure of the sub-trees in the Residential Management Tree corresponds to the one defined in “RFC-139 JMX Control of OSGi”, which defines four interfaces for OSGi framework core APIs; FrameworkMBean, BundleStateMBean, PackageStateMBean and ServiceStateMBean. Although the fundamental architecture adheres to the API of RFC-139, some features such as command execution and structural data exchange are adjusted to realize functionality in a hierarchical object tree, because the Java interfaces are difficult to map completely to the Residential Management Tree which adheres to the DMT Admin model. One of the biggest differences is that returned values of OSGi Core API's methods cannot be returned in DMT Admin interfaces.

### 6.1 Legend

All nodes of the Residential Management Tree are described in a table format. This table format defines the following meta information:

- **Add** – An x indicates that the implementation must support the creation of the given node by the management system.

- **Get** – An x indicates that the implementation must support retrieval of the properties of the given node (including the value).

- **Replace** - An x indicates that the implementation must support setting the value of the given node. Support for changing other properties is optional. Note, that this column does not correspond to the node attribute changing, which can be provided by an implementation even if the node value cannot be changed, for example in case it supports setting the Title property.

- **Delete** - An x indicates that the implementation must support deletion of the given node by the management system.

- **Exec** - An x indicates that the implementation must support the execute operation for the given node.

- **Type** - The node type for an interior node, or the data type for a leaf node. The following data types are defined: str, int, float, date, time, bin, xml, bool, b64.

- **Cardinality** - The range of occurrences of the given node. * means infinite.

- **Scope** - The scope indicates the creation strategy. It can have the following values:
  - **P** - Permanent. A permanent node cannot be changed by the management system. It can, however, appear due to an internal device event, for example, the addition of a network interface.
  - **D** - Dynamic. A node that must be created by the management system. Such a creation can then automatically create other nodes.
  - **A** - Automatic. A node that is created automatically by a managed object if its parent node is created.
6.2 Framework Object

The Framework Object is a managed object that allows manipulation of the OSGi framework functions; StartLevel configuration, Bundle install, Framework Lifecycle control and Bundle lifecycle control. The Framework Object is an optional managed object, therefore this object does not have to be implemented if the user does not need to manipulate the framework.

The tree structure of Framework Object can be accessed from the $/OSGi/<instance_id>/Framework sub-tree. Figure 2 shows the structure of the Framework Object sub-tree.

![Framework Object Diagram]

The Framework Object consists of 5 parts; StartLevel, InstallBundle, FrameworkLifecycle, BundleControl and Ext. These sub-trees represent the individual functions that are manipulated through the Residential Management Tree.

The StartLevel sub-tree contains RequestedStartLevel, ActiveStartLevel and InitialBundleStartLevel. RequestedStartLevel and InitialBundleStartLevel are Start Level configurations of the OSGi framework. ActiveStartLevel is a read-only node, not writable, which represents the start level of the framework at the moment. The FrameworkLifecycle sub-tree controls the state of the OSGi framework itself, but the detail behaviors corresponding to these commands depend on the implementation of the OSGi framework. The InstallBundle sub-tree is used to install bundles into the OSGi framework. This sub-tree supports the simultaneous installation of multiple bundles. The BundleControl sub-tree controls a bundle start level and lifecycle of an individual bundle after installation. The Ext sub-tree extends the data model of the Framework Object.

The Framework Object must support transactions because all changes to the $/OSGi/<instance_id>/Framework tree must be done in an atomic session to keep the OSGi Framework consistent. Only atomic sessions can perform the required changes to the node values.

The following introduces the steps taken to install a bundle. At first, $/OSGi/<instance_id>/Framework/InstallBundle/<id> node must be created by ReadWriteDataSession#createInteriorNode() method, its children will be created automatically by the Framework Object. Next, the value of Location and URL must be set by ReadWriteDataSession#setNodeValue() method.
After that, when TransactionalDataSession#commit() method is called, InstallBundle sub-tree should call BundleContext#installBundle(String, InputStream) method where the first argument is the Location and the second argument is the InputStream object retrieved from the specified URL.

[REMARK] If commit() method is called BEFORE any value of Location and URL node is set by setNodeValue() method, InstallBundle sub-tree must NOT call installBundle() method. If commit() method is called AFTER either of them is set, installBundle() method must be called. Multiple bundles can be installed by just one commit() if multiple $/OSGi/<instance_id>/Framework/InstallBundle/<id> are created before one commit() is called.

The BundleControl sub-tree provides bundle start level and life cycle operations of an individual bundle after installation. Regarding bundle life cycle control, the operation is basically conducted by indicating the desired state of the bundle; Active, Resolved or Uninstalled in the $/OSGi/<instance_id>/Framework/BundleControl/<bundle_id>/Lifecycle/DesiredState node. This sub-tree supports lazy activation and transient start (or stop). When commit() is called, Bundle#start(int option) or Bundle#stop(int option) is called where the argument is the value set in the $/OSGi/<instance_id>/Framework/BundleControl/<bundle_id>/Lifecycle/Option node. The value of Option node must be retained beyond the data plug-in bundle reboot. Details of the values are shown in Tale 6.1

[REMARK] A Protocol Adapter (or local manager) should set not only “Resolved” or “Active” in DesiredState but also Option node in same transactional session. The reason is that if a protocol adapter (or a local manager) sets only the DesiredState node, not the Option node, a value that the remote manager does not expect might be kept in the Option node. In that case, the operation resulting from this value is likely to differ from what the remote manager expects.

Note that BundleUpdate is indicated directly as a command because this kind of operation does not represent the specific desired state of a bundle. The value set to BundleUpdate node is the string of the new bundle’s URL. The OperationResult node represents the result of the latest operation on this bundle.

To use the TR-069 protocol to control the Residential Management Tree, the node name of $/OSGi/<instance_id>/Framework/InstallBundle/<id> should be represented by a numeric character string, not a literal node name such as Location or URL. The reason is that TR-069 has only the RPC called “AddObject” to create a new node in a tree, and the RPC can take as an argument the path name of the collection of objects for which new nodes are to be created.

[REMARK] According to TR-069, a dynamic node path in the Residential Management Tree should be defined as numeric character. Therefore, the path name of the new node should be assigned as an instance number by incrementing the number; the management system cannot specify the identifier of the new node.

All nodes for the Framework Object sub-tree are explained in Table 6.1.

Table 6.1 Framework sub-tree Nodes

<table>
<thead>
<tr>
<th>URI</th>
<th>Add</th>
<th>Get</th>
<th>Replace</th>
<th>Delete</th>
<th>Exec</th>
<th>Type</th>
<th>Cardinality</th>
<th>Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Framework</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>P</td>
<td>Framework Root node.</td>
</tr>
<tr>
<td>Framework/StartLevel</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>P</td>
<td>Interior node that contains the values for StartLevel configuration. If the StartLevel service is unavailable, this node and its child nodes must not be created.</td>
</tr>
<tr>
<td>URI</td>
<td>Add</td>
<td>Get</td>
<td>Replace</td>
<td>Delete</td>
<td>Exec</td>
<td>Type</td>
<td>Cardinality</td>
<td>Scope</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>---------</td>
<td>--------</td>
<td>------</td>
<td>------</td>
<td>-------------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>Framework/StartLevel/RequestedStartLevel</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>int</td>
<td>1</td>
<td>P</td>
<td>A leaf node used to configure the Framework's StartLevel. When this node value is replaced or the Bundles sub-tree starts, StartLevel#setStartLevel with the specified value must be called. This value must be kept persistently.</td>
</tr>
<tr>
<td>Framework/StartLevel/ActiveStartLevel</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>int</td>
<td>1</td>
<td>P</td>
<td>A leaf node that contains the Framework's current StartLevel. This node is read-only to get the Framework's StartLevel.</td>
</tr>
<tr>
<td>Framework/StartLevel/InitialBundleStartLevel</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>int</td>
<td>1</td>
<td>P</td>
<td>A leaf node used to configure the initial bundle StartLevel. When this node value is replaced or the Bundles sub-tree starts, StartLevel#setInitialStartLevel with the specified value must be called.</td>
</tr>
<tr>
<td>Framework/FrameworkLifecycle</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>node</td>
<td>1</td>
<td>P</td>
<td>Interior node that contains the values for Lifecycle control of the OSGi Framework.</td>
</tr>
<tr>
<td>Framework/FrameworkLifecycle/Restart</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>bool</td>
<td>1</td>
<td>P</td>
<td>A leaf node used to restart the OSGi Framework. This node is writable to set the restart command. If this node value is replaced with 'TRUE', the Framework sub-tree must restart the OSGi Framework.</td>
</tr>
<tr>
<td>Framework/FrameworkLifecycle/Shutdown</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>bool</td>
<td>1</td>
<td>P</td>
<td>A leaf node used to shutdown the OSGi Framework. This node is writable to set the shutdown command. If this node value is replaced with 'TRUE', the Framework sub-tree must shutdown the OSGi Framework.</td>
</tr>
<tr>
<td>Framework/FrameworkLifecycle/Update</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>bool</td>
<td>1</td>
<td>P</td>
<td>A leaf node used to update the OSGi Framework. This node is writable to set the update command. If this node value is replaced with 'TRUE', the Framework sub-tree must update the OSGi Framework.</td>
</tr>
<tr>
<td>Framework/InstallBundle</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>node</td>
<td>1</td>
<td>P</td>
<td>Interior node that contains the values for bundle installation.</td>
</tr>
<tr>
<td>Framework/InstallBundle/&lt;id&gt;</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>node</td>
<td>D</td>
<td></td>
<td>Interior node that represents a bundle which will be installed to the OSGi Framework. This node is created for each bundle to be installed. When a transaction is committed, the Framework Object attempts to install the bundle specified under this node. See details of the installation operation described in the Location definition.</td>
</tr>
<tr>
<td>URI</td>
<td>Add</td>
<td>Get</td>
<td>Replace</td>
<td>Delete</td>
<td>Exec</td>
<td>Type</td>
<td>Cardinality</td>
<td>Scope</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>---------</td>
<td>--------</td>
<td>------</td>
<td>------</td>
<td>-------------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>Framework/InstallBundle/&lt;id&gt;/Location</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>str</td>
<td>1</td>
<td>A</td>
<td>A leaf node used to set the bundle location under which the specified bundle is installed. This node is writable because the node is used to install a bundle. If the Location is replaced by setNodeValue() method and the URL is empty String when commit() method is called, BundleContext#installBundle(location) must be called where location is the specified value in this node. When the transaction is committed where both this node Location and URL are replaced and commit() method is called, BundleContext#installBundle(location, in) must be called where location is the specified value of this node and in is an InputStream retrieved from the specified URL. InstallBundle sub-tree must call installBundle() method only if either values of Location or URL node is set by setNodeValue() method and called commit() method. Even if commit() is called, installBundle() method must not be called if either of them is not set. If the installation of a bundle succeeds, the Framework Object must delete the Framework/InstallBundle/&lt;id&gt; node corresponding to the bundle and its descendants. If the installation of a bundle fails, the Framework Object will NOT retry the operation and instead creates an Error node to represent the reason for operation failure. Once the bundle install process fails, the &lt;id&gt; object is not checked when commit() method is called next time.</td>
</tr>
<tr>
<td>Framework/InstallBundle/&lt;id&gt;/URL</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>str</td>
<td>1</td>
<td>A</td>
<td>A leaf node used to be set the bundle's jar file URL from which the specified bundle is installed. This node is writable because the node is used to install a bundle. The default value of this node is an empty string. See details of the installation operation described in the Location definition.</td>
</tr>
<tr>
<td>Framework/InstallBundle/&lt;id&gt;/Error</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>str</td>
<td>0,1</td>
<td>A</td>
<td>A leaf node used to represent the reason for an installation failure. When the installation of a bundle fails, the Framework Object should create this node and set a value specifying the error reason.</td>
</tr>
<tr>
<td>URI</td>
<td>Add</td>
<td>Get</td>
<td>Replace</td>
<td>Delete</td>
<td>Exec</td>
<td>Type</td>
<td>Cardinality</td>
<td>Scope</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>-----</td>
<td>-----</td>
<td>---------</td>
<td>--------</td>
<td>------</td>
<td>------</td>
<td>-------------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>Framework/BundleControl</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Framework/BundleControl/&lt;bundle_id&gt;</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Framework/BundleControl/&lt;bundle_id&gt;/BundleStartLevel</td>
<td>X</td>
<td>X</td>
<td>int</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Framework/BundleControl/&lt;bundle_id&gt;/Lifecycle</td>
<td>X</td>
<td></td>
<td>node</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Framework/BundleControl/&lt;bundle_id&gt;/Lifecycle/DesiredState</td>
<td>X</td>
<td>X</td>
<td>int</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Framework/BundleControl/&lt;bundle_id&gt;/Lifecycle/BundleUpdate</td>
<td>X</td>
<td>X</td>
<td>str</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td></td>
</tr>
</tbody>
</table>

**Description**

- **Framework/BundleControl**
  - Interior node that contains the control functions for each bundle.

- **Framework/BundleControl/<bundle_id>**
  - Interior node that represents a Bundles instance. This node is automatically added and `<bundle_id>` is incremented. This number must equal the bundle id, which is Bundle#getBundleId() returns.

- **Framework/BundleControl/<bundle_id>/BundleStartLevel**
  - The configuration node for the StartLevel of the bundle. When the node value is gotten, this object must return the current StartLevel of the bundle. If this value is changed, StartLevel#setBundleStartLevel with the specified value must be called.

- **Framework/BundleControl/<bundle_id>/Lifecycle**
  - This is a parent node for commands related to the bundle's life-cycle control. This sub-tree is described in detail later.

- **Framework/BundleControl/<bundle_id>/Lifecycle/DesiredState**
  - A leaf node used to control the bundle's life-cycle. When this node value is replaced with the state described below or the BundleControl sub-tree starts, the BundleControl sub-tree must change the bundle life-cycle to the specified bundle state. The BundleControl sub-tree must return an error, if the specified state cannot be understood. This state must be one of the following: 0 – Uninstalled 4 – Resolved 32 – Active
The Framework Object does not have to retry the operation if life-cycle control fails, but the error status should be written in the OperationResult. This value must be kept persistently.

- **Framework/BundleControl/<bundle_id>/Lifecycle/BundleUpdate**
  - A leaf node used to update the bundle. This node is writable to set the update command. When this node value is replaced with an URL string, Bundle#update(InputStream) must be called where InputStream is the specified URL. If the specified URL is an empty string, Bundle#update() must be called so that the specified bundle is updated with the jar-file indicated by the BundleLocation. If the update fails, the Bundle object does not have to retry the operation. If DesiredState and Update are replaced during same transaction, BundleControl
<table>
<thead>
<tr>
<th>URI</th>
<th>Add</th>
<th>Get</th>
<th>Replace</th>
<th>Delete</th>
<th>Exec</th>
<th>Type</th>
<th>Cardinality</th>
<th>Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Framework/BundleControl/&lt;bundle_id&gt;/Lifecycle/</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>int</td>
<td>1</td>
<td>A</td>
<td></td>
<td>A leaf node used to set start or stop option. A Protocol Adapter (PA) (or local manager) should set not only DesiredState but also this node in the same transactional session. After called commit() method, Lifecycle sub-tree must call Bundle#start(int option) or Bundle#stop(int option). The value of this node must be kept beyond the data plug-in bundle reboot. Option node contains integer format data as below. 0:default 1:START_TRANSIENT or STOP_TRANSIENT 2:START_ACTIVATION_POLICY 3:START_ACTIVATION_POLICY</td>
</tr>
<tr>
<td>Option</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(Example 1) If a Protocol Adapter wants to start a Bundle transiently, the Protocol Adapter should set “Active” to DesiredState node and “1” to Option node.  (Example 2) If a Protocol Adapter wants to start a bundle not transiently, the Protocol Adapter should set DesiredState node to “Active” and Option node to “0”. (Example 3) If a Protocol adapter wants to stop a bundle transiently, it should set DesiredState node to “Resolved” and this node to “1”.</td>
</tr>
<tr>
<td>Framework/BundleControl/</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>str</td>
<td>1</td>
<td>A</td>
<td></td>
<td>This node holds the latest operation's result which is conducted through the Framework/BundleControl/&lt;bundle_id&gt;/Lifecycle/ sub-tree, so that the Management System can derive the result of the bundle life-cycle operation. If the operation succeeds, the value string must start with “Success: ”. Otherwise it must start with “Fail”.</td>
</tr>
<tr>
<td>&lt;bundle_id&gt;/Lifecycle/OperationResult</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Framework/BundleControlNumberOfEntries</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>str</td>
<td>1</td>
<td>A</td>
<td></td>
<td>A leaf node that represents the number of &lt;bundle_id&gt; nodes.</td>
</tr>
<tr>
<td>Framework/Ext</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>node</td>
<td>1</td>
<td>P</td>
<td></td>
<td>Interior node that can contain values for user extensions.</td>
</tr>
</tbody>
</table>
6.3 BundleState Object

Figure 3 shows the BundleState sub-tree architecture. This sub-tree is used to obtain the information of a bundle from the OSGi framework. When a bundle is installed on the OSGi framework, a new node of $\text{/OSGi/}\langle\text{instance}_\text{id}\rangle/\text{BundleState/}\langle\text{bundle}_\text{id}\rangle$ is automatically added and $\langle\text{bundle}_\text{id}\rangle$ is incremented. In other words, $\langle\text{bundle}_\text{id}\rangle$ is equivalent to Bundle#getId(). The identifiers including SymbolicName and Version are used to identify a bundle on the OSGi framework. BundleType, an integer value, represents the type of the bundle. Manifest represents the Manifest header by String value. StartLevel shows the bundle's start level on this OSGi framework. State shows the bundle state; Installed, Resolved, Starting, Active, Stopping and Uninstalled. Location, a String value, represents the BundleLocation of the bundle. Fragments, Hosts, Required, Requiring contain the comma-separated list of corresponding bundle identifiers. TrustedSignerCertificates and NonTrustedSignerCertificates contains the list of the Certificate chains. The one is only trusted certificates, the other is only NOT trusted certificates by the framework.

The BundleState Object must be kept after uninstallation of the bundle until the org.osgi.framework.Bundle object of the bundle will be deleted from the OSGi Framework. In other words, once created, the BundleState Object of a bundle must remain available while the OSGI Framework continues to run.

All nodes for the BundleState Object sub-tree are explained in Table 6.2.
Table 6.2  BundleState sub-tree Nodes

<table>
<thead>
<tr>
<th>URI</th>
<th>Add</th>
<th>Get</th>
<th>Replace</th>
<th>Delete</th>
<th>Exec</th>
<th>Type</th>
<th>Cardinality</th>
<th>Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BundleState</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>node 1</td>
<td>A</td>
<td>This is a parent node for status information of the bundle. This sub-tree is detailed later.</td>
</tr>
<tr>
<td>BundleState/&lt;bundle_id&gt;</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>node 0..*</td>
<td>A</td>
<td>A node that represents a Bundles instance. This number must equal the bundle id, which Bundle#getBundleId() returns.</td>
</tr>
<tr>
<td>BundleState/&lt;bundle_id&gt;/SymbolicName</td>
<td>X</td>
<td>str</td>
<td>1</td>
<td></td>
<td></td>
<td>str 1</td>
<td></td>
<td>A</td>
<td>The Bundle-SymbolicName of the bundle.</td>
</tr>
<tr>
<td>BundleState/&lt;bundle_id&gt;/Version</td>
<td>X</td>
<td>str</td>
<td>1</td>
<td></td>
<td></td>
<td>str 1</td>
<td></td>
<td>A</td>
<td>The version of the bundle.</td>
</tr>
<tr>
<td>BundleState/&lt;bundle_id&gt;/BundleType</td>
<td>X</td>
<td>int</td>
<td>1</td>
<td></td>
<td></td>
<td>int 1</td>
<td></td>
<td>A</td>
<td>A node indicating the type of the bundle. The node value must be equivalent to the value of PackageAdmin#getBundleType().</td>
</tr>
<tr>
<td>BundleState/&lt;bundle_id&gt;/Manifest</td>
<td>X</td>
<td>str</td>
<td>1</td>
<td></td>
<td></td>
<td>str 1</td>
<td></td>
<td>A</td>
<td>This node is a leaf node that contains Manifest headers as String. The node implementation must get Dictionary object by Bundle#getHeaders(). For converting the Dictionary object to String, this node must have the value adhering to the following format (LF means linefeed): &quot;&lt;key&gt;:&lt;value&gt;LF&lt;key&gt;:&lt;value&gt;LF...&quot; where order of appearance depends on its implementation.</td>
</tr>
<tr>
<td>BundleState/&lt;bundle_id&gt;/Status</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>node 1</td>
<td>A</td>
<td>This node is the parent of the nodes that represent the status of the bundle.</td>
</tr>
<tr>
<td>BundleState/&lt;bundle_id&gt;/Status/Location</td>
<td>X</td>
<td>str</td>
<td>1</td>
<td></td>
<td></td>
<td>str 1</td>
<td></td>
<td>A</td>
<td>The BundleLocation of the bundle.</td>
</tr>
<tr>
<td>BundleState/&lt;bundle_id&gt;/Status/State</td>
<td>X</td>
<td>int</td>
<td>1</td>
<td></td>
<td></td>
<td>int 1</td>
<td></td>
<td>A</td>
<td>The state of the bundle as returned by Bundle#getState(). This state is one of the following: 0 – Not Available 2 – Installed 4 – Resolved 8 – Starting 16 – Stopping 32 – Active</td>
</tr>
<tr>
<td>BundleState/&lt;bundle_id&gt;/Status/StartLevel</td>
<td>X</td>
<td>int</td>
<td>1</td>
<td></td>
<td></td>
<td>int 1</td>
<td></td>
<td>A</td>
<td>The StartLevel of the bundle. Because this value is a read-only node in this sub-tree, user must use</td>
</tr>
<tr>
<td>URI</td>
<td>Add</td>
<td>Get</td>
<td>Replace</td>
<td>Delete</td>
<td>Exec</td>
<td>Type</td>
<td>Cardinality</td>
<td>Scope</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>---------</td>
<td>--------</td>
<td>------</td>
<td>------</td>
<td>-------------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>URI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BundleState/&lt;bundle_id&gt;/</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status/PersistentlyStarted</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BundleState/&lt;bundle_id&gt;/</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status/LastModified</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BundleState/&lt;bundle_id&gt;/</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Host</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BundleState/&lt;bundle_id&gt;/</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fragments</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BundleState/&lt;bundle_id&gt;/</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Required</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BundleState/&lt;bundle_id&gt;/</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requiring</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BundleState/&lt;bundle_id&gt;/</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TrustedSignerCertificates</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BundleState/&lt;bundle_id&gt;/</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NonTrustedSignerCertificate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Description**

- The Framework Object to configure the StartLevel.
- The status of the bundle at the last shutdown of the OSGi Framework. If the bundle is Active when the OSGi Framework shutdowns, the value must be TRUE. Otherwise, the value must be FALSE.
- The latest time at which the bundle has been modified.
- A leaf node that represents the list of bundle IDs of the host bundles to which this Bundles instance is attached. The value is a comma-separated list of bundle IDs. If there are no host bundles, this value must be an empty string.
- A leaf node that represents the list of bundle IDs of attached fragment bundles for this Bundles instance. The value is a comma-separated list of bundle IDs. If there are no fragment bundles, this value must be an empty string.
- A leaf node that represents the list of bundle IDs of required bundles. The value is a comma-separated list of bundle IDs. If there are no required bundles, this value must be an empty string.
- A leaf node that represents the list of bundle IDs of requiring bundles. The value is a comma-separated list of bundle IDs. If there are no requiring bundles, this value must be an empty string.
- A leaf node that represents the list of the X.509Certificate information, only trusted by the framework. Those can be gotten by Bundle#getSignerCertificates(SIGNERS_TRUSTED). See 6.3.1 for the format of it.
- A leaf node that represents the list of the X.509 Certificate information, only NOT trusted by the framework. Those are all gotten by Bundle#getSignerCertificates(SIGNERS_ALL) except ones in TrustedSignerCertificates node. See 6.3.1 for the format of it.
### 6.3.1 Certificate information format

The following expression is format of TrustedSignerCertificates and NonTrustedSignerCertificates node value, each of which is list of the certificate information of Signers of the bundle.

There is a possibility that a bundle is signed by multiple signers and a Certificate has Certificate chain which consists of distinguished name (DN)s. The last element is root.

**Example of Certificate chain**

```
  cn=Duffy Duck, o=Tweety Inc., C=JP …  cn=Bugs Bunny, o=ACME, C=US
```

Three kinds of separator are required to represent the list.

- **Separator1**: between “Certificate Chain1” and “Certificate Chain2” of a bundle.
- **Separator2**: between DN1 and DN2 in a Certificate Chain.
- **Separator3**: between elements in a DN.

This RFC adopts comma for Separator1 and semicolon for Separator3, respectively, where each DN is enclosed in ‘<’ and ‘>’ (Separator2).

**Example of the value**

```
<cn=Duffy Duck; o=Tweety Inc.; C=JP><cn=Bugs Bunny; o=ACME; C=US>,<cn=Road Runner; o=ACME; C=US><cn=Bugs; o=ACME; C=US>
```

[Remarks] Some nodes defined in this RFC, such as Fragments, Hosts, Required, Requiring in BundleState Object, contains comma sererated list. These are designed to be compliant to TR-106[7], which has the following definition:

*For strings that are defined to contain comma-separated lists, the format is defined as follows. Between every pair of successive items in a comma-separated list there MUST be a separator. The separator MUST include exactly one comma character, and MAY also include one or more space characters before or after the comma. The entire separator, including any space characters, MUST NOT be considered part of the list items it separates. The last item in a comma-separated list MUST NOT be followed with a separator. Individual items in a comma-separated list MUST NOT include a space or comma character within them. If an item definition requires the use of spaces or commas, that definition MUST specify the use of an escape mechanism that prevents the use of these characters.*

To be compliant to TR-106, Separator 1 must be comma, first. Here, Java String of a DN may contain comma for separator 3. However, it cannot be allowed for TR106 as quoted above. On the other hand, RFC1779[9] allows either comma or semicolon for separator 3. Therefore this RFC adopts semicolon for separator 3. This means, the implementation of these nodes must replace comma in a Java String in a DN into semicolon.

---

<table>
<thead>
<tr>
<th>URI</th>
<th>Add</th>
<th>Get</th>
<th>Replace</th>
<th>Delete</th>
<th>Exec</th>
<th>Type</th>
<th>Cardinality</th>
<th>Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BundleState/&lt;bundle_id&gt;/BundleStateExt</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>node</td>
<td>1</td>
<td>A</td>
<td>Interior node that can contain values for user extension.</td>
</tr>
</tbody>
</table>
On the other hand, a remote manager must parse the value of the node by comma first. Then, individual item in a comma-separated list represents one Certificate chain. The remote manager must parse the Certificate chain by separator 2 to get DN Strings in the Certificate chain, then parse the DN by separator 3.
6.4 PackageState Object

The PackageState Object is a managed object that allows the package information to be derived. This object can be used to retrieve package dependencies between bundles.

Figure 4 shows the overall architecture of the PackageState object. The 
$/OSGi/<instance_id>/PackageState/<id> node is created for an individual package existing on the OSGi framework. This node represents a package's information including bundle dependencies. The pairs between each package and the corresponding <id> must be kept persistently as long as the package is exported from the same bundle, which means the same bundle ID, beyond restart of the OSGi Framework.

The Name node contains a qualified package name, and the Version node contains the version number of the package. The ExportingBundle node shows a bundle identifier exporting the package. On the other hand, the ImportingBundles sub-tree shows the bundles importing the package. These nodes can be used to get information on packages shared between bundles.

The PackageState Object may support only readable session because it does not contain any writable node.

All nodes for the PackageState Object sub-tree are explained in Table 6.3.
Table 6.3  PackageState sub-tree Nodes

<table>
<thead>
<tr>
<th>URI</th>
<th>Add</th>
<th>Get</th>
<th>Replace</th>
<th>Delete</th>
<th>Exec</th>
<th>Type</th>
<th>Cardinality</th>
<th>Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PackageState</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>node 1 P</td>
<td></td>
<td>PackageState Root node containing package information existing on the OSGi Framework. The children of this node must represent the actual package status when this sub-tree is accessed.</td>
</tr>
<tr>
<td>PackageState/&lt;id&gt;</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>node 0..* A</td>
<td></td>
<td>A node that represents the Package instance. This is equivalent to org.osgi.service.packageadmin.ExportedPackage object.</td>
</tr>
<tr>
<td>PackageState/&lt;id&gt;/</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>str 1 A</td>
<td></td>
<td>The qualified name of the package.</td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PackageState/&lt;id&gt;/</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>str 1 A</td>
<td></td>
<td>The version of the package.</td>
</tr>
<tr>
<td>Version</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PackageState/&lt;id&gt;/</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bool 1 A</td>
<td></td>
<td>A leaf node that represents the removal status of the package. If a bundle exporting the package has already been uninstalled or updated but the package is still used, this node must be TRUE.</td>
</tr>
<tr>
<td>RemovalPending</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PackageState/&lt;id&gt;/</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>str 1 A</td>
<td></td>
<td>A leaf node that represents the bundle ID of the exporting bundle. The value is a bundle ID of string.</td>
</tr>
<tr>
<td>ExportingBundle</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PackageState/&lt;id&gt;/</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>str 1 A</td>
<td></td>
<td>A node that represents the list of bundle IDs of importing bundles. The value is a comma-separated list of bundle IDs. If there is no importing bundle, this value must be an empty string.</td>
</tr>
<tr>
<td>ImportingBundles</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6.5 ServiceState Object

The ServiceState Object is a managed object that allows the service information to be derived. This object can be used to retrieve service dependencies between bundles.

[Diagram: ServiceState Object]

Figure 5 shows the overall architecture of the ServiceState object. The $/OSGi/<instance_id>/ServiceState node is created for an individual service instance existing on the OSGi framework. This node represents a service's information including registering bundle and using bundles. Registering a service to the OSGi framework automatically adds a new Service instance under the $/OSGi/<instance_id>/ServiceState node with incrementation of <service_id>. In other words, <service_id> equals the service.id of the service instance in the OSGi framework.

The Properties node contains service properties of the service, which include the interface names implemented by the service. The Properties node must contain all service properties which consist of string, boolean or numeric data types including single-dimension arrays or vectors. However non-serializable data types can be discarded from the Properties sub-tree, since these types of properties are difficult to be represented in object trees. The RegisteringBundle node shows the id of the registering bundle. On the other hand, the UsingBundles sub-tree shows bundles using the service. These nodes can be used to get information on the relationships between registering bundle and using bundles.

In order to get service interface names of a service, $/OSGi/<instance_id>/ServiceState/<service_id>/Properties/objectClass/Values/<n> node can be used.

The ServiceState Object may support only readable session because it does not contain any writable node.

All nodes for the ServiceState Object sub-tree are explained in Table 6.4.
## Table 6.4 ServiceState sub-tree Nodes

<table>
<thead>
<tr>
<th>URI</th>
<th>Add</th>
<th>Get</th>
<th>Replace</th>
<th>Delete</th>
<th>Exec</th>
<th>Type</th>
<th>Cardinality</th>
<th>Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServiceState</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>node</td>
<td>1</td>
<td>P</td>
</tr>
<tr>
<td>ServiceState/&lt;service_id&gt;</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>node</td>
<td>0..*</td>
<td>A</td>
</tr>
<tr>
<td>ServiceState/&lt;service_id&gt;/Properties</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>node</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>ServiceState/&lt;service_id&gt;/Properties/&lt;id&gt;</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>node</td>
<td>0..*</td>
<td>A</td>
</tr>
<tr>
<td>ServiceState/&lt;service_id&gt;/Properties/&lt;id&gt;/Key</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>str</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>ServiceState/&lt;service_id&gt;/Properties/&lt;id&gt;/Types</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>str</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>ServiceState/&lt;service_id&gt;/Properties/&lt;id&gt;/Cardinality</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>str</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>ServiceState/&lt;service_id&gt;/Properties/&lt;id&gt;/Values</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>str</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>ServiceState/&lt;service_id&gt;/PropertiesNumberOfEntries</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>str</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>ServiceState/&lt;service_id&gt;/RegisteringBundle</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>str</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>ServiceState/&lt;service_id&gt;/UsingBundles</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>str</td>
<td>1</td>
<td>A</td>
</tr>
</tbody>
</table>
6.5.1 Service Property Dictionary nodes

The service property Dictionary consists of key-value pairs. The service property Dictionary is mapped to a sub-tree. The URI for a service property item is the following:

$/OSGi/<instance_id>/ServiceState/<service_id>/Properties/<id>

Key nodes are leaf nodes which represent the key of the service property. Their type, cardinality, and value are represented as separate nodes. These sub-nodes are:

- **Type** – Type contains only one Java type name of Values such as `java.lang.Float`, `float`, `java.lang.String`, `char`, etc.

- **Cardinality** – Defines if the value is a scalar, an array, or a vector. It can take the following values:
  - scalar – For simple, unstructured values, like a string or a byte[].
  - array – When the value is a Java array (but not byte[]).
  - collection – When the value must be a Java Collection object.

- **Values** – If Cardinality is array, Values contains the list of values of the service properties. Otherwise if Cardinality is collection, Values contains the list of values of the service properties value, where all values have the same Java type. That is the limitation because it would be too complex to handle different Java types in Collection's values. Otherwise, Values contains only one value.

The actual value (Values child nodes) is mapped to a DmtData type if possible. If this mapping is not possible, the node must be a str node and the Java class of the given type must be able to parse the value in a constructor.

The mapping of data types between DMT Admin and TR-069 are described in RFC149.
6.6 Filters Object

The Filters Object is a managed object that searches the nodes in a tree that correspond to the filter expression. This Filter Object is a generic mechanism for the whole management tree below the $/OSGi node although Filters sub tree is located under the $/OSGi/<instance_id> node. In other words, the Filter Object plugin must be able to filter all nodes in the tree except Filters Object itself and Log Object because the Log Object already has an original filtering mechanism.

The Filters Object can be used to group bundles, packages, services and other information in the tree. Since the filter string set in the Filter node has no restriction in terms of its usage policy, users can use this function in accordance with their needs.

The $/OSGi/<instance_id>/Filters object is used to search nodes by filtering values or names of nodes located under the sub-tree specified by $/OSGi/<instance_id>/Filters/<search_id>/TargetSubtree. At first, the user needs to create the $/OSGi/<instance_id>/Filters/<search_id> node by incrementing the <search_id> number, which should be a numeric character string as demanded by the TR-069 protocol. Then the user sets the desired partial path in the $/OSGi/<instance_id>/Filters/<search_id>/TargetSubtree node, that specifies the sub-tree required to provide information as the result, and sets an appropriate filter string in $/OSGi/<instance_id>/Filters/<search_id>/Filter. When the user accesses under the $/OSGi/<instance_id>/Filters/<search_id>/Result node, the sub-trees that match the filter are extracted from the sub-trees specified by $/OSGi/<instance_id>/Filter/<search_id>/TargetSubtree and are copied as children sub-trees of the $/OSGi/<instance_id>/Filters/<search_id>/Result node. Therefore, the sub-trees that match the filter string are aligned under the $/OSGi/<instance_id>/Filters/<search_id>/Result node with the absolute path from the $ node.

The <search_id> node is a dynamic node which means that the path name of the new node should be assigned as an instance number by incrementing the largest existing number as demanded by TR-069. Therefore, the node name should be defined as numeric character in the Residential Management Tree.

All nodes for the Filters Object sub-tree are explained in Table 6.5.
### Table 6.5 Filters Object sub-tree Nodes

<table>
<thead>
<tr>
<th>URI</th>
<th>Add</th>
<th>Get</th>
<th>Replace</th>
<th>Delete</th>
<th>Exec</th>
<th>Type</th>
<th>Cardinality</th>
<th>Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filters</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>node 1</td>
<td>P</td>
<td>The root node of Filters Object.</td>
</tr>
<tr>
<td>Filters/&lt;search_id&gt;</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>node 0..*</td>
<td>D</td>
<td>Represents a filter search request. This node is created to set the filter expression and to get the sub-trees, which match with the filter expression. The &lt;search_id&gt; is assigned in ascending order to Filters instances when they are set. This is a unique ID of Filters instance and must be kept persistently.</td>
</tr>
<tr>
<td>Filters/&lt;search_id&gt;/</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>str 1</td>
<td>A</td>
<td>Partial path to the sub-tree under which Filter is going to be matched. This must be the absolute path of the top node name of the sub-tree. The default value of this node is an empty string, which means no filtering must be done. This value must be kept persistently.</td>
</tr>
<tr>
<td>TargetSubtree</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Filters/&lt;search_id&gt;/Filter</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>str 1</td>
<td>A</td>
<td>Contains the filtering expression. The filter must be given in the OSGi Filter format. See Filter and Target Expression section. An empty string indicates that no filtering must be done. An empty string is the default value for this node. This value must be kept persistently.</td>
</tr>
<tr>
<td>Filters/&lt;search_id&gt;/Result</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>node 1</td>
<td>A</td>
<td>Matched sub-trees are stored under the Result node. Any children of this node must represent the actual status of the sub-tree whenever the children nodes in this sub-tree are accessed. The result must be only those sub-trees that satisfy the specified Filter node. And the filtered sub-trees must include all nodes located under the sub-tree specified by the TargetSubtree node. See Filter and Target Expression section.</td>
</tr>
</tbody>
</table>

### 6.6.1 Filter and Target Expression

The expression of $/OSGi/<instance_id>/Filters/<search_id>/TargetSubtree must be a partial path from the “$” node to the top node of the target sub-tree to be filtered and must end with “/”, which implies that a partial path must indicate an interior node. The characters “*” and “-” must be allowed to be used in the path as a wild-card; “*” indicates a wild-carded node for 1 level, while “-” indicates multiple levels in a node path respectively. Multiple wild-card can be used in a node path. If an incorrect string, such as a string including an incorrect format path, is specified in $/OSGi/<instance_id>/Filters/<search_id>/TargetSubtree, the Filter Object should throw a DmtException to the caller and should keep the TargetSubtree node value as an empty string.
The filter expression of $/OSGi/<instance_id>/Filters/<search_id>/Filter should be a LDAP search filter (RFC 1960. See Filter for a description of the filter string syntax.). This expression is equivalent to the OSGi Filter format.

The key string in a filter must be a node name which exists in the sub-tree specified by $/OSGi/<instance_id>/Filters/<search_id>/TargetSubtree. Therefore it must not contain "/". All nodes equivalent to the specified node name must be the targets for the filtering. If the indicated node is not included in the sub-tree, the Filters Object must not create sub-trees under the Result node. If there exists multiple nodes with the indicated node name under the specified TargetSubtree, the Filters Object must perform the filtering for all nodes that match the node name. A wild-card must not be allowed to be used in a key string. Both interior and leaf node can be specified as the key string of a filter.

The value string in a filter indicates either a leaf node value or a node name. If an interior node is specified as the key string, node names of children nodes of the specified interior node must be recognized as the target values to be matched. On the other hand, if a leaf node is specified, the value of the specified leaf node must be recognized as the target value. A wild-card can be used at the end of the value string to conduct prefix-searches in either cases.

In a filter expression, types of value should be ignored because the Filter Object must recognize the filter strings as String. Therefore, the filter matching operation demands that the filter value should be translated into the type appropriate for each leaf node.

For the service properties sub-trees in the ServiceState Object and the configuration properties sub-trees in the Configuration Object, such as $/OSGi/<instance_id>/ServiceState/<service_id>/Properties and $/OSGi/<instance_id>/Configuration/<pid>/Keys respectively, the filtering must be done in a different way from other sub-trees in the following three respects:

1. For those sub-trees, the value of $/OSGi/<instance_id>/Filters/<search_id>/Filter must be described in the same way as in the properties filtering implemented by BundleContext#getServiceReferences() method.

2. A key name of service property or configuration property may conflict names of other nodes under the specified TargetSubtree. Therefore, to match against service properties or configuration properties, a key must be prefixed with the commercial at sign ‘@’ (0u0040) in a filter expression. It means, a remaining key string without ”@” indicates the property key to be searched, and the value string represents the desired value of the property. For example, ”@objectClass” will refer to a service property with the name "objectClass".

3. The key and value in the filter string are processed in a case sensitive manner unless the key name references a service property or configuration property, which are case insensitive.

Assume that the key string of a filter is prefixed with “@”. The Filter Object must recognize the nodes which has the name as same as the remaining key string without “@” under the service properties subtree or the configuration properties subtree, or the nodes which has the name as same as the key string including “@” under other subtrees.

When the $/OSGi/<instance_id>/Filters/<search_id>/Result sub-tree is accessed, the returned sub-tree must reflect the current situation; the detailed mechanism of the synchronization depends on the implementation. When the node is accessed, the filter matching sub-trees are created. The Filters Object searches nodes that satisfy the specified keys and values in $/OSGi/<instance_id>/Filters/<search_id>/Filter against the sub-trees specified by $/OSGi/<instance_id>/Filters/<search_id>/TargetSubtree, and must create a matched sub-tree under the $/OSGi/<instance_id>/Filters/<search_id>/Result node. The absolute node path from the $ node must be created with the actual node name of the wild-card appearing in the node path specified by $/OSGi/<instance_id>/Filters/ <search_id>/TargetSubtree.
The sub-trees under the $/OSGi/<instance_id>/Filters/<search_id>/Result must be read-only in order to keep consistency among data-plugins related to the filter search. The Filter Object must prevent attempts to access the sub-tree to change node value or properties, and must throw DmtException to the caller.

The following expressions are examples of the Filter and TargetSubtree:

[Case1] Simple TargetSubtree and Filter usage

- **TargetSubtree:** $/OSGi/1/BundleState/
- **Filter:** (SymbolicName=org.osgi.*)

- Bundles whose Bundle-SymbolicName correspond to "org.osgi.*" are matched. The Result node contains corresponding sub-trees that have absolute node paths descending from $ node. A possible sub-tree under the Result node is described below ($ node should be changed to the actual node path depending on each execution environment):
[Case2] Filter including an interior node name as a key value and TargetSubtree including "*" as a wild-card

- TargetSubtree: $/OSGi/*/PackageState/
- Filter: (&(ExportingBundle=5)(ImportingBundles=10))

  - Packages that have a leaf node named "5" as a child node of ExportingBundle and have a leaf node named "10" among children nodes of ImportingBundles are matched. The filter search is performed for all OSGi Frameworks included in the management tree, because the <instance_id> of the OSGi Framework is represented as a wild-card. The Result node contains corresponding sub-trees that have absolute node paths descending from $ node. A possible sub-tree under the Result node is described below ($ node should be changed to the actual node path depending on each execution environment):

```
Result
  .$/
  .OSGi
    .1
      .PackageState
        .27
          .Name
          .Version
          .RemovalPending
          .ExportingBundle = 5
          .ImportingBundles = 10, 13
    .3
      .PackageState
        .11
          .Name
          .Version
          .RemovalPending
          .ExportingBundle = 5
          .ImportingBundles = 10, 22
```

[Case3] Filter for service properties filtering

- TargetSubtree: $/OSGi/1/ServiceState/
- Filter: (@application=automation)

  - Services that have properties including "application" as a key and "automation" as a value are matched. The Result node contains corresponding sub-trees that have relative node paths descending from the <service_id> node. A possible sub-tree under the Result node is described below ($ node should be changed to the actual node path depending on each execution environment):
[Case4] TargetSubtree including "-" as multiple level wild-card

- TargetSubtree: $/OSGi/-/BundleControl/
- Filter: (BundleStartLevel=2)
  - Bundles whose BundleStartLevel correspond to 2, which is obtained by calling StartLevel#getBundleStartLevel(Bundle), are matched. The filter search is performed for all OSGi Frameworks included in the management tree, because the <instance_id> of the OSGi Framework is included in the wild-card. The Result node contains corresponding sub-trees that have absolute node paths descending from $ node. A possible sub-tree under the Result node is described below ($ node should be changed to the actual node path depending on each execution environment):

```
Result
  .$
  .OSGi
    .1
      .Framework
        .BundleControl
          .5
            .BundleStartLevel = 2
            .Lifecycle
          .9
            .BundleStartLevel = 2
            .Lifecycle
    .2
      .Framework
        .BundleControl
          .8
            .BundleStartLevel = 2
            .Lifecycle
```
6.7 BundleResources Object

The BundleResources sub-tree is optional and it is used to derive resources in the bundle jar file. The BundleResources sub-tree consists of interior and leaf nodes that correspond to actual file-paths and files in the jar file respectively, so that a remote manager can derive a bundle's resources by a simple operation. For example, suppose there is /META-INF/Manifest.mf file in a jar file, $OSGi/<instance_id>/BundleResources/<bundle_id>/META-INF/Manifest.mf nodes are automatically created by the BundleResources Object, which implements the BundleResource sub-tree. Therefore, a remote manager simply gets the node value of $OSGi/<instance_id>/BundleResources/<bundle_id>/META-INF/Manifest.mf.

The <directory> node is automatically created and represents a directory included in a jar file. The <bundle_id> node must indicate the root directory of the jar file. On the other hand, <file_name> node is also created automatically as a leaf node, but represents a file name in a jar file. The <file_name> node contains the content of the file as node value, which is encoded in base 64 format. If the size of the content exceeds a limit which depends on the BundleResources Object implementation, the BundleResources Object should abort the reading of file content and should throw a DmtException.

Because the BundleResources sub-tree is created automatically based on the actual file architecture of the indicated jar file, this sub-tree must be read-only. A remote manager can not add, replace, delete and exec nodes in this sub-tree.

Note: Assume that adopted remote management protocol implements a method that retrieves data of all nodes under the specified node recursively (such as “GetParameterValues” RPC of TR-069 protocol). When a remote manager indicates $OSGi/<instance_id>/BundleResources/<bundle_id> node or ancestor nodes of it, heavy data transaction between the remote manager and client would occur because the BundleResources sub-tree contains all file of the bundle jar-file. Therefore, when using such kind of method in an operation, a remote manager should carefully specify the node in terms of performance.

[REMARK] A considered alternative of the BundleResources sub-tree architecture is shown in section 7.

All nodes for the Resources Object sub-tree are explained in Table 6.6.
Table 6.6 Resources sub-tree Nodes

<table>
<thead>
<tr>
<th>URI</th>
<th>Add</th>
<th>Get</th>
<th>Replace</th>
<th>Delete</th>
<th>Exec</th>
<th>Type</th>
<th>Cardinality</th>
<th>Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BundleResources</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>node</td>
<td>1</td>
<td>This is a parent node for resources included in the bundle. This sub-tree is detailed later.</td>
</tr>
<tr>
<td>BundleResources/&lt;bundle_id&gt;</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>node</td>
<td>0..*</td>
<td>A node that represents a Bundles instance. This number must equal the bundle id, which Bundle#getBundleId() returns.</td>
</tr>
<tr>
<td>BundleResources/&lt;bundle_id&gt;/&lt;directory&gt;</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>node</td>
<td>0..*</td>
<td>Represents a directory name. This node is automatically created to show a file-path in a bundle jar-file.</td>
</tr>
<tr>
<td>BundleResources/&lt;bundle_id&gt;/&lt;file_name&gt;</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>b64</td>
<td>0..*</td>
<td>The content of the file specified in the node path. This leaf node is automatically created for an individual file in a bundle jar-file. The node value is encoded in base 64 format.</td>
</tr>
</tbody>
</table>

6.8 Configuration Object

The Configuration Object manages the Configuration Admin service via the DMT Admin service. The features of this object are the same as those defined in the Mobile Management Tree except for the position of the top node: here the top node can be accessed by $/OSGi/<instance_id>/Configuration while the top node in Mobile Management Tree can be accessed at $/Configuration.

6.9 Policy Object

The Policy Object manages the Conditional Permission Admin service and the Permission Admin service via the DMT Admin service. The features of this object are the same as those defined in the Mobile Management Tree except for the position of the top node: here the top node can be accessed by $/OSGi/<instance_id>/Policy while the top node in the Mobile Management Tree can be accessed at $/Policy.

6.10 Log Object

The Log Object manages the Log service via the DMT Admin service. The features of this object are the same as those defined in the Mobile Management Tree except for the position of the top node: here the top node can be accessed by $/OSGi/<instance_id>/Log here while the top node in the Mobile Management Tree can be accessed at $/Log.
6.11 Limitations

The Residential Management Tree has several limitations due to the specification of a remote management protocol (TR-069) utilized with the DMT Admin service. The following section describes these limitations.

6.11.1 Maximum and Minimum values of numeric parameters

Due to the TR-069 specification, only the data types Integer or Unsigned Integer are available in the Residential Management Tree. Therefore, the maximum value of an unsigned integer should be less than 4294967295, and the minimum value should be greater than or equal to -2147483648. Consequently, the Bundle ID and the service.id which are originally a Long value in the OSGi specification should be represented as integer value in the Residential Management Tree.

6.11.2 Life-cycle control of the Framework

The life-cycle control of the OSGi Framework depends on the implementation of the framework on which the Residential Management Tree can run. This RFC doesn't specify the results of calling $/OSGi/<instance_id>/Framework/FrameworkLifecycle/*.

6.11.3 Service Properties Expression

The following list summarizes the restrictions and rules that should be obeyed in order to maintain a valid ServiceState Object.

- Complex data types, multiple-dimension arrays or vectors might be discarded from the Properties sub-tree, since these types of properties are difficult to be represented in the object tree if the data isn't serializable.
7 Considered Alternatives

7.1 BundleResources sub-tree architecture

There is considerable alternative for designing the BundleResources sub-tree to retrieve resources inside the bundle jar file. The following architecture is another design of the BundleResources sub-tree, which was eventually discarded as the result of discussion.

The `<resource_id>` node is a dynamic node, which means the path name of the new node is automatically assigned as an instance number by the Bundles Object. In advance, the management system needs to create the `<resource_id>` node. Then it has to indicate the file path by setting the Path node parameters and can retrieve the contents of the specified file as the value of the Content node.

This architecture has pros and cons compared to the proposed architecture in Section 6.8.

First, this architecture enables a scalable implementation because the remote manager is able to decide which resources should be retrieved from bundles arbitrarily. The remote manager, therefore, can avoid heavy data traffic between remote manager and client, when an ancestor node of the Resources sub-tree is specified by the GetParameterValues RPC defined in the TR-069.

Secondly, the typical usage of the Resources sub-tree would be for diagnostics scenarios; when some problems occur, a remote operator who needs to find the cause of the problem retrieves the content of the bundle JAR file by using this sub-tree.

But this architecture limits the diagnostics ability of a remote manager. The remote manager has to check many files by repetitively creating `<resources_id>` nodes until the cause of problem is detected. This situation prevents an effective diagnostics through the Resources sub-tree.

On the other hand, the proposed architecture in Section 6.8 provides a better diagnostics ability than this but the performance decreases in terms of data transactions when retrieving data recursively.
Consequently, the architecture proposed in Section 6.8 is chosen for the Resources sub-tree due to the effectiveness of the diagnostics ability, even though there is a risk of a performance drawback when a remote manager indicates `${instance_id}/BundleResources/${bundle_id}` node or ancestor nodes.

# 8 Security Considerations

All security requirements follow the DMT Admin specification.

# 9 Document Support

## 9.1 References


[3]. OMA, Open Mobile Alliance. The mission of the Open Mobile Alliance is to facilitate global user adoption of mobile data services by specifying market driven mobile service enablers that ensure service interoperability across devices, geographies, service providers, operators, and networks, while allowing businesses to compete through innovation and differentiation. [http://www.openmobilealliance.org/](http://www.openmobilealliance.org/)

[4]. OMA Device Management specification v1.2. The goal of the Device Management Working Group is to specify protocols and mechanisms that achieve management of mobile devices including the necessary configuration to access services and management of the software on mobile devices. [http://www.openmobilealliance.org/release_program/dm_v1_2C.html](http://www.openmobilealliance.org/release_program/dm_v1_2C.html)


[6]. The Broadband Forum is a global consortium of nearly 200 leading industry players covering telecommunications, equipment, computing, networking and service provider companies. Established in 1994, originally as the ADSL Forum and later the DSL Forum, the Broadband Forum continues its drive for a global mass market for broadband, to deliver the benefits of this technology to end users around the world over existing copper telephone wire infrastructures. [http://www.broadband-forum.org/about/forumhistory.php](http://www.broadband-forum.org/about/forumhistory.php)

[7]. Data model template for TR-069 enabled devices, TR-106 amendment 1, November 2006
9.2 Author's Address

<table>
<thead>
<tr>
<th>Name</th>
<th>Koya Mori</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>NTT Corporation</td>
</tr>
<tr>
<td>Address</td>
<td>Y320C, 1-1 Hikari-no-oka, Yokosuka, Kanagawa, Japan</td>
</tr>
<tr>
<td>Voice</td>
<td>+81-46-859-3446</td>
</tr>
<tr>
<td>e-mail</td>
<td><a href="mailto:mori.kouya@lab.ntt.co.jp">mori.kouya@lab.ntt.co.jp</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Ikuo Yamasaki</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>NTT Corporation</td>
</tr>
<tr>
<td>Address</td>
<td>Y320C, 1-1 Hikari-no-oka, Yokosuka, Kanagawa, Japan</td>
</tr>
<tr>
<td>Voice</td>
<td>+81-46-859-8537</td>
</tr>
<tr>
<td>e-mail</td>
<td><a href="mailto:yamasaki.ikuo@lab.ntt.co.jp">yamasaki.ikuo@lab.ntt.co.jp</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Shigekuni Kondo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>NTT Corporation</td>
</tr>
<tr>
<td>Address</td>
<td>Y320C, 1-1 Hikari-no-oka, Yokosuka, Kanagawa, Japan</td>
</tr>
<tr>
<td>Voice</td>
<td>+81-46-859-3444</td>
</tr>
<tr>
<td>e-mail</td>
<td><a href="mailto:kondo.shigekuni@lab.ntt.co.jp">kondo.shigekuni@lab.ntt.co.jp</a></td>
</tr>
</tbody>
</table>

9.3 Acronyms and Abbreviations

9.4 End of Document
Abstract

Different industries are interested in applying OSGi to advance their businesses, in which remote management is a key issue. In the residential area, the TR-069 is the one of the de-facto standard protocol for remote management. The best way to realize remote management based on the TR-069 on OSGi is utilizing DMT Admin service, which has been defined in the OSGi Alliance for the mobile device management. In this case, TR-069 is implemented as a protocol adapter of the DMT Admin. The DMT Admin service, however, is designed mainly for OMA-DM, which is the de-facto standard protocol in mobile area. Although these protocols have the similar objectives and functionality, there are several differences. One of which is the data types definition. Therefore, this RFC defines new data types in the DMT Admin service.
1 Document Information

1.1 Table of Contents

1 Document Information.................................................................2
  1.1 Table of Contents.........................................................2
  1.2 Terminology and Document Conventions.................................3
  1.3 Revision History..............................................................3

2 Introduction............................................................................4

3 Application Domain....................................................................4

4 Problem Description....................................................................5
  4.1 Management agent making use of OSGi standardized Java interfaces.....5
  4.2 Mobile specification approach................................................6
  4.2.1 Support for TR-069 notifications...........................................6
  4.2.2 Limitations in the number of services available in the DMT..............7
  4.2.3 Mapping TR-069 to the OMA-DM inspired DMT model...................8
  4.3 Conclusion..........................................................................8

5 Requirements...........................................................................8

6 Technical Solution......................................................................9
  6.1 Addition of data types........................................................9
  6.2 Javadoc..............................................................................9
  6.3 info.dmtree
     Class DmtData.........................................................................9
     6.3.1 FORMAT_LONG...........................................................10
     6.3.2 FORMAT_DATETIME..................................................10
     6.3.3 DmtData.......................................................................11
     6.3.4 DmtData.......................................................................11
     6.3.5 getDateDateTime............................................................12
     6.3.6 getLong.......................................................................12
     6.3.7 toString......................................................................12

7 Considered Alternatives...........................................................13

8 Security Considerations...........................................................13

9 Document Support....................................................................13
  9.1 References.........................................................................13
  9.2 Author's Address..............................................................14
  9.3 Acronyms and Abbreviations................................................14
  9.4 End of Document..............................................................14
1.2 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.

1.3 Revision History

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

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>Sep. 16 2008</td>
<td>Initial Draft</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Koya Mori, NTT Corporation, <a href="mailto:mori.kouya@lab.ntt.co.jp">mori.kouya@lab.ntt.co.jp</a></td>
</tr>
<tr>
<td>2nd</td>
<td>Jan. 20 2009</td>
<td>2nd Draft, Generic setter/getter methods are added, Limitation of notification is described.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Koya Mori, NTT Corporation, <a href="mailto:mori.kouya@lab.ntt.co.jp">mori.kouya@lab.ntt.co.jp</a></td>
</tr>
<tr>
<td>1.0.0</td>
<td>Oct. 16 2009</td>
<td>Public Draft, Javadoc and Considered Alternatives are added, Notification property definition is changed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Koya Mori, NTT Corporation, <a href="mailto:mori.kouya@lab.ntt.co.jp">mori.kouya@lab.ntt.co.jp</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ikuo Yamasaki, NTT Corporation, <a href="mailto:yamasaki.ikuo@lab.ntt.co.jp">yamasaki.ikuo@lab.ntt.co.jp</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Shigekuni Kondo, NTT Corporation, <a href="mailto:kondo.shigekuni@lab.ntt.co.jp">kondo.shigekuni@lab.ntt.co.jp</a></td>
</tr>
<tr>
<td>1.1.0</td>
<td>April 3 2010</td>
<td>For 3rd Public Draft, Based on the discussion of REG F2F in Mountain View, the following modifications are done;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• abstract was modified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Requirements which this RFC cannot meet are removed in Section 5.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• all descriptions on notifications are removed in Section 6.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 7.1 is added for future work on notification.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Koya Mori, NTT Corporation, <a href="mailto:mori.kouya@lab.ntt.co.jp">mori.kouya@lab.ntt.co.jp</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ikuo Yamasaki, NTT Corporation, <a href="mailto:yamasaki.ikuo@lab.ntt.co.jp">yamasaki.ikuo@lab.ntt.co.jp</a></td>
</tr>
</tbody>
</table>
2 Introduction

Traditionally, fixed telecommunication operators don’t have knowledge about what runs in the customer’s local area network (LAN). They provide connectivity and manage the wide area network (WAN) that provides this connectivity, but they do not know anything about the devices and networks behind the gateway (xDSL mainly) that interconnect WAN and LAN. Recently the need for management of customer networks and devices is increasing in order to make the deployment of new complex services at home (home automation, tele-health, VoIP, IPTV, surveillance, etc) feasible with reasonable costs. For example to avoid sending technicians to the customer premises for solving problems.

There are two main kinds of devices that need to be managed in operator’s business: those which come from the fixed business managed through TR-069 and standardized by the Broadband FORUM, and those which come from the mobile business, managed through OMA-DM and standardized by the Open Mobile Alliance. The DMT Admin specification of OSGi covers the OMA-DM ones. In addition, the design of DMT Admin Service specification potentially allows adaption of other remote management protocols other than the OMA-DM.

The best way to realize remote management based on TR-069 on OSGi is utilizing the DMT Admin service. In this case, TR-069 is implemented as a protocol adapter of the DMT Admin. However, there are several architectural differences between the TR-069 and the OMA-DM, although these two protocols have the similar objectives and functionality.

3 Application Domain

Driven by triple play service delivery in the home network, fixed line access service providers have the need to configure home devices to ensure the proper service delivery. Broadband Forum’s CPE WAN Management Protocol (CWMP, alias TR-069) enables them to do this. By using a remote management server (Auto Configuration Server, ACS), they are able to manage TR-069 enabled devices. TR-069 provides them with possibilities to configure parameters, be informed of parameter changes (notification), start diagnostic tools, update firmware, etc.

Similarly, for the mobile world, the OMA defined the OMA-Device Management specification for remote management of mobile devices. OMA-DM offers similar tools to the mobile service providers as TR-069 to fixed line service provider, but OMA-DM is of course tailored to the specifics of the mobile environment.
As OSGi technology offers a flexible software environment for all these different devices, the remote management of the platform is of interest for both fixed and mobile service providers. As such, it should be possible to integrate the remote management of the OSGi platform, and the applications running on top of it, in the existing management infrastructure.

The DMT Admin service with its mobile management tree in the Mobile specification for OSGi R4 standardizes the remote management of an OSGi platform. As it is largely inspired by OMA-DM, it needs to be evaluated for multi protocol support.

4 Problem Description

In a scenario in which service providers offer a growing number of services, to use specific solutions for the management of those services is not the most suitable option. To speed up the deployment of these services, such as triple play, home automation or tele-health, it is essential to offer general management solutions that allow for the management of a large number of services and the flexible life-cycle management of applications.

These devices usually are already managed by a standard protocol, so it makes sense that an OSGi framework, which hosts the services, running on a device could be managed in the same way as the other resources of the device. Of course, the remote management should be fully integrated in the existing remote management solutions of the service provider to avoid duplicating management infrastructure and to increase performance on the devices.

Currently, there are two options in OSGi for remote management:

- create a management agent bundle making use of the Java object interfaces,
- create a protocol adapter bundle that interacts with the DMT Admin service, as defined in the OSGi Mobile specification.

4.1 Management agent making use of OSGi standardized Java interfaces

Currently, for the management of a bundle, the OSGi specifications define different Java objects with which a management application can interact. Using this approach, a management agent can implement extensive management of the OSGi framework, as well as any service standardized. Mapping the Java interfaces to the specific remote management protocol and data model tree is up to the management agent.

For runtime interaction with a bundle, a bundle can register a service in the service registry. However, this service interface is not standardized. Also, mapping the service interface to a general management model is not standardized. A current approach is to implement a proprietary service interface on all bundles to be managed. By tailoring this interface so that it easily maps to the management protocol primitives, it is simple for the management agent to map remote management commands to the bundle’s service interface. The disadvantage is the proprietary service interface, so that 3rd party bundles might not be compliant.

As a conclusion we can say that this current approach allows for extensive remote management of any aspect of the OSGi platform, but lacks a standardized service interface definition for bundles to implement.
4.2 Mobile specification approach

The Mobile Expert Group has provided its own solution based on the OMA [3] Device Management [4] specification to provide a remote management solution. The OSGi Mobile specification contains two chapters related to remote management:

- chapter 3: detailing the mobile management tree
- chapter 117: detailing the DMT Admin service, bundle plugin interface specification, the notification service

The Device Management Tree model of OMA-DM was chosen as meta-data model and operational model. However, it was intended to be mappable to other protocols.

An analysis of mapping the Mobile specification DMT model to TR-069, however, shows that the current DMT model approach (as defined in the OSGi R4 Mobile Specification) introduces some issues. For example:

- Limitations for active or passive notifications on any parameter in the object tree
- A limited number of services have been mapped to the DMT model
- The complexity of mapping a new protocol to the OMA-inspired DMT model, which could imply performance issues on limited devices.

4.2.1 Support for TR-069 notifications

TR-069 offers the feature of active and passive notifications. By setting a parameter’s notification attribute, a remote manager requests to be notified with the parameter’s new value at the time the value changes (active notification) or at the next periodic inform (passive notification). Notification can be configured on any parameter of the TR-069 object tree. This approach enables the remote manager to be informed not only of changes in status variables of the platform, but also of configuration changes performed by a local manager, e.g. through a local Web interface.

The Mobile specification offers a few features that could help to implement TR-069 notification support:

- The DMT Admin service sends events using the Event Admin service when operations have been performed on nodes (nodes added, removed or copied; node values changed etc.)
- The OSGi Notification service defines a way to alert a remote management server. Protocol adaptors on their turn have to implement a RemoteAlertSender interface (and register it) for use by the notification service. Notifications are sent by calling sendNotification on the notification service:

- The Monitor Admin service: A bundle can register a Monitorable service, to be used by the Monitor Admin service. By registering a Monitorable service, the bundle exposes access to a number of status variables. Notification can be implemented by the StatusVariable provider. If it does, it will call the update method on the Monitor Listener. The Monitor Admin service then generates an event on the Event Admin service. The Monitor service is currently also represented in the DMT tree.

Two problems arise when trying to map the current approach to TR-069:

- TR-069 defines that notification is applicable to any parameter in the object tree.

Currently, the DMT Admin service only send events for operations on DMT nodes that were performed using the DMT Admin API. For example: if configuration changes are performed by
using the Configuration Admin service API, no events will be sent. Most of the current implementations do not perform all changes via the DMT Admin service. Therefore, the events sent by the DMT Admin service are an only subset and thus not very reliable as single source of events (and thus as single source of TR-069 notifications).

The OSGi Monitor service only supports notification of changes on Status Variables, exposed through a Monitorable service, and enabled by the bundle to support on-change notification (i.e. dynamic Status Variables).

- Requesting notification is not fully under the control of the remote manager. In the case of a bundle using the notification service, there is no standardized way to configure the bundle to send alerts when the value of one of the implemented DMT nodes changes. In the case of the monitor service, the sending of events can be controlled, but is limited to dynamic Status Variables.

The current DMT Admin service has no attributes properties on its nodes to be used to configure notification behavior, such as active notification and passive notification defined in TR-069. Therefore, a remote manager cannot control the notification behavior of DMT nodes in a standardized way.

To conclude, the current options, as provided in the Mobile specification, limit notification of parameter changes to StatusVariables, explicitly enabled for monitoring. There is no standardized approach available to monitor changes on any node in the DMT.

### 4.2.2 Limitations in the number of services available in the DMT

The OSGi R4 Mobile specification mapped a number of services to the DMT tree. However, these services are limited to the services listed in the Mobile Specification. Other interesting services, as listed in the OSGi R4 Service Compendium are not yet mapped (standardized) in the DMT tree:

The DMT tree defined in the Mobile Specification contains objects for the following services:

- Configuration Admin service
- Log service
- Monitor Admin service
- Application Admin service
- Conditional Permission Admin service
- Deployment service

A number of areas that could be of interest to a remote manager are currently missing in the DMT tree:

- Startlevel management
- Bundle management: managing individual bundles as opposed to deployment packages (inventory, life-cycle management, exported services, …)
- Service management: getting a remote view on services registered in the service registry
- Permission Admin management
Home Gateway Core Function management: handling Home Gateway core functions, such as firewall configuration and port forwarding control, from bundles running on an OSGi framework

4.2.3 Mapping TR-069 to the OMA-DM inspired DMT model

Within the OSGi Mobile specification, the choice has been made to model the DMT after OMA-DM.

As a result, creating an OMA-DM protocol adapter is quite straightforward. Although no major hurdles have been identified in creating a TR-069 protocol adapter, it is less straightforward:

1. The TR-069 RPC primitives have to be translated to the DMT Admin service interface methods (which are OMA-DM RPC inspired).

2. The TR-069 tree has to be mapped to the DMT tree. Translating object model specific features like DMT meta nodes, or TR-069 attributes is not straightforward. It might require specific extensions to the DMT, e.g. to support TR-069 attributes, or a single node in the DMT might result in multiple objects in a TR-069 data model, etc.

3. The TR-069 data types have to be mapped to the DMT Admin data types. However, TR-069 data types, such as “unsignedint” and “dateTime” (ISO 8601), cannot be translated appropriately into DMT Admin data types defined in the current specification. Translating these data types might result a limitation of the available value range and a complex object that consists of multiple nodes, respectively.

4.3 Conclusion

The OSGi Mobile specification delivers a standardized data model (the DMT), and standardized interface (on the DMT Admin service) to enable remote management through a protocol adapter. However, the current specification lacks management objects for a number of interesting areas. Also, there is some support lacking for TR-069 notifications. Furthermore, since the DMT model is OMA-DM inspired, implementing a TR-069 protocol adapter is not straightforward, although not impossible.

5 Requirements

REQUIREMENT[9]: The solution should be able to handle the data types of “unsignedint” and “dateTime” (ISO 8601) to accommodate TR-069 protocol.
6 Technical Solution

This RFC describes how DMT Admin service is extended to accommodate the TR-069 protocol. Since DMT Admin service intends to use the OMA-DM protocol as a remote management protocol, some key features such as data types are missing which makes it difficult to support the TR-069 protocol.

This RFC describes the following changes to DMT Admin service.

- Addition of new data types to DMT Admin service

There is no need to modify the basic architecture.

6.1 Addition of data types

Two data types are missing in the DMT Admin specification and are needed to accommodate the TR-069 protocol. The following data types should be added to the format of the DmtData class.

- FORMAT_LONG – A long value. Since there is no suitable data type for unsignedInt in Java, this data type should be mapped as unsigned integer for use by the TR-069 protocol.

- FORMAT_DATETIME – A String object that is interpreted as the dateTime type defined in ISO 8601; it is used as the value of date and time in TR-069.

6.2 Javadoc

Note that only APIs that have been modified and added are included in this document.

6.3 info.dmtree

Class DmtData

java.lang.Object

\[\text{info.dmtree.DmtData}\]

public final class DmtData extends java.lang.Object

<table>
<thead>
<tr>
<th>Field Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int</td>
</tr>
<tr>
<td>The node holds a String object that is interpreted as a dateTime type defined in ISO 8601.</td>
</tr>
<tr>
<td>static int</td>
</tr>
<tr>
<td>The node holds a long value.</td>
</tr>
</tbody>
</table>
## Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DmtData(long ln)</code></td>
<td>Create a <code>DmtData</code> instance of long format and set its value.</td>
</tr>
<tr>
<td><code>DmtData(java.lang.String value, int format)</code></td>
<td>Create a <code>DmtData</code> instance of the specified format and set its value based on the given string.</td>
</tr>
</tbody>
</table>

## Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>java.lang.String getDateTime()</code></td>
<td>Gets the value of a node with dateTime format.</td>
</tr>
<tr>
<td><code>long getLong()</code></td>
<td>Gets the value of a node with long format.</td>
</tr>
<tr>
<td><code>java.lang.String toString()</code></td>
<td>Gets the string representation of the <code>DmtData</code>.</td>
</tr>
</tbody>
</table>

**Methods inherited from class java.lang.Object**

getClass, notify, notifyAll, wait, wait, wait

## Field Detail

### 6.3.1 FORMAT_LONG

```java
public static final int FORMAT_LONG
```

The node holds a long value. This data type should be mapped as unsigned integer if needed.

See Also:
Constant Field Values

### 6.3.2 FORMAT_DATETIME

```java
public static final int FORMAT_DATETIME
```

The node holds a String object that is interpreted as a dateTime type defined in ISO 8601.

See Also:
Constant Field Values
Constructor Detail

6.3.3 DmtData

```java
public DmtData(java.lang.String value,
               int format)
```

Create a DmtData instance of the specified format and set its value based on the given string. Only the following string-based formats can be created using this constructor:

- **FORMAT_STRING** - value can be any string
- **FORMAT_XML** - value must contain an XML fragment (the validity is not checked by this constructor)
- **FORMAT_DATE** - value must be parseable to an ISO 8601 calendar date in complete representation, basic format (pattern CCYMMDD)
- **FORMAT_TIME** - value must be parseable to an ISO 8601 time of day in either local time, complete representation, basic format (pattern hhmmss) or Coordinated Universal Time, basic format (pattern hhmmssZ)
- **FORMAT_DATETIME** - value must be parseable to an ISO 8601 calendar date-time in complete representation, basic format (pattern ccyy-mm-ddThh:mm:ssZ)

The null string argument is only valid if the format is string or XML.

**Parameters:**
- `value` - the string, XML, date, time or dateTime value to set
- `format` - the format of the DmtData instance to be created, must be one of the formats specified above

**Throws:**
- `java.lang.IllegalArgumentException` - if format is not one of the allowed formats, or value is not a valid string for the given format
- `java.lang.NullPointerException` - if a date or time is constructed and value is null

6.3.4 DmtData

```java
public DmtData(long ln)
```

Create a DmtData instance of long format and set its value.

**Parameters:**
- `ln` - the long value to set
6.3.5 getDateTime

public java.lang.String getDateTime()

Gets the value of a node with dateTime format. The returned dateTime string is formatted according to the ISO 8601 definition of a calendar date-time in complete representation, basic format (pattern ccyy-mm-ddThh:mm:ssZ).

Returns:
the dateTime value

Throws:
DmtIllegalStateException - if the format of the node is not dateTime

6.3.6 getLong

public long getLong()

Gets the value of a node with long format.

Returns:
the long value

Throws:
DmtIllegalStateException - if the format of the node is not long

6.3.7 toString

public java.lang.String toString()

Gets the string representation of the DmtData. This method works for all formats.

For string format data - including FORMAT_RAW_STRING - the string value itself is returned, while for XML, date, time, dateTime, integer, long, float, boolean and node formats the string form of the value is returned. Binary - including FORMAT_RAW_BINARY - and base64 data is represented by two-digit hexadecimal numbers for each byte separated by spaces. The NULL_VALUE data has the string form of "null". Data of string or XML format containing the Java null value is represented by an empty string.

Overrides:
toString in class java.lang.Object

Returns:
the string representation of this DmtData instance
7 Considered Alternatives

Nothing to be described.

8 Security Considerations

All security requirements follow the DMT Admin specification.

9 Document Support

9.1 References


[3]. OMA, Open Mobile Alliance. The mission of the Open Mobile Alliance is to facilitate global user adoption of mobile data services by specifying market driven mobile service enablers that ensure service interoperability across devices, geographies, service providers, operators, and networks, while allowing businesses to compete through innovation and differentiation. http://www.openmobilealliance.org/

[4]. OMA Device Management specification v1.2. The goal of the Device Management Working Group is to specify protocols and mechanisms that achieve management of mobile devices including the necessary configuration to access services and management of the software on mobile devices. http://www.openmobilealliance.org/release_program/dm_v1_2C.html


[6]. The Broadband Forum is a global consortium of nearly 200 leading industry players covering telecommunications, equipment, computing, networking and service provider companies. Established in 1994, originally as the ADSL Forum and later the DSL Forum, the Broadband Forum continues its
drive for a global mass market for broadband, to deliver the benefits of this technology to end users around the world over existing copper telephone wire infrastructures. [http://www.broadband-forum.org/about/forumhistory.php](http://www.broadband-forum.org/about/forumhistory.php)

[7]. Data model template for TR-069 enabled devices, TR-106 amendment 1, November 2006

### 9.2 Author's Address

<table>
<thead>
<tr>
<th>Name</th>
<th>Koya Mori</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>NTT Corporation</td>
</tr>
<tr>
<td>Address</td>
<td>Y320C, 1-1 Hikari-no-oka, Yokosuka, Kanagawa, Japan</td>
</tr>
<tr>
<td>Voice</td>
<td>+81-46-859-3446</td>
</tr>
<tr>
<td>e-mail</td>
<td><a href="mailto:mori.kouya@lab.ntt.co.jp">mori.kouya@lab.ntt.co.jp</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Ikuo Yamasaki</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>NTT Corporation</td>
</tr>
<tr>
<td>Address</td>
<td>Y320C, 1-1 Hikari-no-oka, Yokosuka, Kanagawa, Japan</td>
</tr>
<tr>
<td>Voice</td>
<td>+81-46-859-8537</td>
</tr>
<tr>
<td>e-mail</td>
<td><a href="mailto:yamasaki.ikuo@lab.ntt.co.jp">yamasaki.ikuo@lab.ntt.co.jp</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Shigekuni Kondo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>NTT Corporation</td>
</tr>
<tr>
<td>Address</td>
<td>Y320C, 1-1 Hikari-no-oka, Yokosuka, Kanagawa, Japan</td>
</tr>
<tr>
<td>Voice</td>
<td>+81-46-859-3444</td>
</tr>
<tr>
<td>e-mail</td>
<td><a href="mailto:kondo.shigekuni@lab.ntt.co.jp">kondo.shigekuni@lab.ntt.co.jp</a></td>
</tr>
</tbody>
</table>

### 9.3 Acronyms and Abbreviations

### 9.4 End of Document
RFC144: Configuration Admin Extension

Draft

35 Pages

Abstract

This document proposes technical solution for fine grained access control to use Configuration Admin Service. To realize it, binding mechanism between bundles and Configuration objects and Configuration Permission are extended.
0 Document Information

0.1 Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 Document Information</td>
<td>2</td>
</tr>
<tr>
<td>0.1 Table of Contents</td>
<td>2</td>
</tr>
<tr>
<td>0.2 Terminology and Document Conventions</td>
<td>3</td>
</tr>
<tr>
<td>0.3 Revision History</td>
<td>4</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>5</td>
</tr>
<tr>
<td>2 Application Domain</td>
<td>5</td>
</tr>
<tr>
<td>3 Problem Description</td>
<td>6</td>
</tr>
<tr>
<td>3.1 Binding between Configuration objects and bundles</td>
<td>6</td>
</tr>
<tr>
<td>3.2 Coarse grain control for configuring Configuration objects</td>
<td>6</td>
</tr>
<tr>
<td>3.3 Terminologies used in this document</td>
<td>6</td>
</tr>
<tr>
<td>4 Requirements</td>
<td>6</td>
</tr>
<tr>
<td>5 Technical Solution</td>
<td>7</td>
</tr>
<tr>
<td>5.1 Configuration Permission Extension</td>
<td>7</td>
</tr>
<tr>
<td>5.1.1 Parameters</td>
<td>7</td>
</tr>
<tr>
<td>5.1.2 Wildcard in location String</td>
<td>8</td>
</tr>
<tr>
<td>5.2 Extended Binding Mechanism between Bundles and Configuration objects</td>
<td>8</td>
</tr>
<tr>
<td>5.2.1 Format of location String of Configuration object</td>
<td>8</td>
</tr>
<tr>
<td>5.2.2 Callback Operation of newly defined Configuration Admin</td>
<td>10</td>
</tr>
<tr>
<td>5.2.3 Callback Operation Lists</td>
<td>11</td>
</tr>
<tr>
<td>5.2.4 Configurations with null location</td>
<td></td>
</tr>
<tr>
<td>5.3 Permission Check Operations of the caller</td>
<td></td>
</tr>
<tr>
<td>5.3.1 Permission Check Operation to get configuration by location</td>
<td>11</td>
</tr>
<tr>
<td>5.3.2 Permission Check Operation to get configuration by only pid</td>
<td>12</td>
</tr>
<tr>
<td>5.3.3 Permission Check Operation to get factory configuration by location</td>
<td>13</td>
</tr>
<tr>
<td>5.3.4 Permission Check Operation to get factory configuration by only factoryPid</td>
<td>14</td>
</tr>
<tr>
<td>5.3.5 Permission Check Operation to list configurations</td>
<td></td>
</tr>
<tr>
<td>5.3.6 Permission Check Operation to get bundle location of a Configuration</td>
<td>14</td>
</tr>
<tr>
<td>5.3.7 Permission Check Operation to set bundle location of a Configuration</td>
<td>14</td>
</tr>
<tr>
<td>5.4 Clarification of the spec</td>
<td>15</td>
</tr>
<tr>
<td>5.5 Use cases</td>
<td>15</td>
</tr>
<tr>
<td>5.5.1 Use case 1: General</td>
<td>15</td>
</tr>
<tr>
<td>5.5.2 Multiple Service Provider Model Use Case 1</td>
<td>16</td>
</tr>
<tr>
<td>5.5.3 Multiple Service Provider Model Use Case2</td>
<td>17</td>
</tr>
<tr>
<td>5.5.4 Multiple Service Provider Model Use Case3</td>
<td>18</td>
</tr>
<tr>
<td>5.5.5 Use Case 4: Traditional Location Usage with Limited Access to Configure</td>
<td>19</td>
</tr>
</tbody>
</table>
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.
<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
</table>
| Initial  | Nov 10, 2008 | Initial draft created.  
Ikuo Yamasaki, NTT Corporation, yamasaki.ikuo@lab.ntt.co.jp  
Shigekuni Kondo, NTT Corporation, kondo.shigekuni@lab.ntt.co.jp |
| 2nd draft | Dec 05, 2008 | Completely renewed based on the discussion on REG F2F in Nov.  
multiple binding is introduced.  
Ikuo Yamasaki, NTT Corporation, yamasaki.ikuo@lab.ntt.co.jp  
Shigekuni Kondo, NTT Corporation, kondo.shigekuni@lab.ntt.co.jp |
| 3rd draft | Jan 14, 2009 | Revised based on the discussion on CPEG F2F, Jan 13, 2009.  
All new methods introduced by 2nd draft are eliminated. Section 3.3 is added for terminologies.  
Ikuo Yamasaki, NTT Corporation, yamasaki.ikuo@lab.ntt.co.jp |
| 4th draft | Jan 29, 2009 | Revised based on the discussion on REG F2F, Jan 23, 2009.  
Whole document is totally renewed.  
Simplify ConfigurationPermission extension, just added prefix match and new action “BOUND”.  
Binding mechanism has been changed.  
Ikuo Yamasaki, NTT Corporation, yamasaki.ikuo@lab.ntt.co.jp |
| 5th draft | April 1, 2009 | Revised for wrong descriptions and add Use Case 4 and descriptions of considerations for the case that the Java Runtime Environment does not support permissions.  
Ikuo Yamasaki, NTT Corporation, yamasaki.ikuo@lab.ntt.co.jp |
| 6th draft | April 16, 2009 | Revised based on the discussion in REG F2F on April 2, 2009.  
Descriptions gets detailed.  
Ikuo Yamasaki, NTT Corporation, yamasaki.ikuo@lab.ntt.co.jp |
| 7th draft | July 23, 2009 | Revised for clarifying descriptions.  
Ikuo Yamasaki, NTT Corporation, yamasaki.ikuo@lab.ntt.co.jp |
| 8th draft | Sep 21, 2009 | Revised for clarifying descriptions.  
Ikuo Yamasaki, NTT Corporation, yamasaki.ikuo@lab.ntt.co.jp |
| 9th draft | Oct 16, 2009 | Revised based on the discussion in Palo Alto CPEG F2F.  
Javadoc is newly added.  
Ikuo Yamasaki, NTT Corporation, yamasaki.ikuo@lab.ntt.co.jp |
1 Introduction

This RFC defines how to realize security control in a finer granularity regarding Configuration objects. The requirements are described in RFP-110.

Though OSGi is integrated with Java2 Security Model, some important resources are not protected by permissions. There is a permission with too coarse-grained scope to use in some application domains. This RFP clarifies the resources to be protected and a permission to be finer-grained. The resources are related to Configuration Admin Service.

To realize it, the ConfigurationPermission is extended and the binding mechanism between Configuration objects and bundles.

2 Application Domain

R4.1 spec defines Configuration Admin Service specifications, which provides unified way for privileged bundles to set configurations for other bundles and for a bundle to use those configurations. The privileged bundles are bundles given ConfigurationPermission.

3 Problem Description

ConfigurationPermission introduced in R4.1 doesn't realize fine-grained control for access to Configuration objects. Current granularity of the permission allows independent bundles to interfere each others through changing configuration.

3.1 Binding between Configuration objects and bundles

In Configuration Admin Service Specification 1.2, one Configuration object is bound to either only one bundle or non. Therefore, common configuration information between multiple bundles must be set into multiple Configurations.
3.2 Coarse grain control for configuring Configuration objects

In Configuration Admin Service Specification 1.2, a bundle granted ConfigurationPermission can get Configuration objects bound to any bundles' locations and a bundle not granted ConfigurationPermission can get no Configuration objects except the ones bound to own bundle location. In other words, the following scenario is impossible to realize: bundle 1 is allowed to get Configuration objects of bundle1 and bundle 2, but not to get ones of bundle 3. In order to realize this, ConfigurationPermission must be changed to support a filter expression for its name as similarly as change of AdminPermission in Core Specification 1.3 (release 4.0).

3.3 Terminologies used in this document

Configuring bundle: a bundle which gets a Configuration object by calling a method of ConfigurationAdmin interface and sets a Dictionary object to the Configuration object by calling update() method of Configuration interface.

Target bundle: a bundle which registers a ManagedService or a ManagedFactoryService with a pid.

Configuration Manager(CM): a bundle which implements Configuration Admin Service Spec. It registers Configuration Admin Service.

4 Requirements

[REQ01] New Spec MUST provide a solution to finer grained access control regarding getting Configuration objects.

[REQ02] New spec MUST allow one Configuration object to be bound to multiple bundles.

[REQ03] New spec MUST be backwards compatible.

5 Technical Solution

This proposal consists of two changes;

1. new ConfigurationPermission will support not only "*" but also arbitrary String in its name argument for fine granularity control and will support new action “BOUND” and “READ”.
2. binding mechanism between bundles and Configuration objects will be extended.
Each of them is related with each other.

### 5.1 Configuration Permission Extension

In Configuration Admin Service Spec Version 1.2, when the following methods are called, whether the caller bundle has the appropriate ConfigurationPermission is checked:

- `ConfigurationAdmin#getConfiguration()` / `createFactoryConfiguration()` / `listConfigurations()`
- `Configuration#getBundleLocation()` / `setBundleLocation()`.

We introduce ConfigurationPermission extension to realize the requirement. Let us assume that the Java Runtime Environment supports permissions.

#### 5.1.1 Parameters

New ConfigurationPermission has the following parameters.

- **Name**
  
  The name should be bundle location of configuration target bundle, "*", or opaque String.

- **Action**
  
  Supported actions are “CONFIGURE”, “READ” and “BOUND”. “CONFIGURE” implies both “BOUND” and “READ”. The * wildcard for the actions parameter is supported.

ConfigurationPermission#implies() returns true if the specified permissions is an ConfigurationPermission AND

- By prefix match, this object’s name matches the name of the specified permissions,
- AND this object’s actions include all of the specified permission’s actions.

CONFIGURE action allows a bundle to get a Configuration object with the location which matches the specified name of this Permission and to set a location of a Configuration object with the location which matches the specified name of this Permission.

BOUND action allows a bundle to be bound to a Configuration object with the location which matches the specified name of this Permission. “a bundle is bound to a Configuration object” means : if the bundle registers a ManagedService with the pid of the Configuration object, CM will call back the service.

READ action allows a bundle to get a location of the Configuration object with the location which matches the specified name of this Permission.

#### 5.1.2 Wildcard in location String

Location String can contain wildcard "*". When permission is checked in this case, "*" is regarded just a character. For instance, when `ConfigurationAdmin#getConfiguration(pid, “opaque*”)` is called, if caller bundle has ConfigurationPermission(“opa*”, CONFIGURE), it will succeed. On the other hand, if caller bundle has ConfigurationPermission(“opaque1*”, CONFIGURE), it will fail.

### 5.2 Extended Binding Mechanism between Bundles and Configuration objects

In Configuration Admin Service Spec Version 1.2, a Configuration object is bound to one bundle or none. The key element of the binding is a location of a Configuration object which represents bundle location of the configuration target bundle. In other words, CM judges which bundle a target Configuration object needs to be bound to by checking whether a target bundle has the same bundle location to the location of the Configuration object.
Our proposal enables a Configuration object to any of one bundle, none and multiple bundles who registered the Configuration object with the target pid.

5.2.1 Format of location String of Configuration object

The new spec does NOT restrict the format of location String at all. Location String is opaque for Configuration Manager. Therefore, the developers of configuring bundles have fully freedom to set any String to location.

5.2.2 Callback Operation of newly defined Configuration Admin

There are various callback operation listed in 5.2.3. This section take one of them to explain how CM judges the bindings between ManagedService(Factory)-s and configurations.

There are the following operations we have to think of;
1: Callback Operation for Configuration object whose location is NOT null, and
2: Callback Operation for Configuration object whose location is null.

5.2.2.1 Callback Operation for Configuration object whose location is NOT null
Assume that there already exists a Configuration object whose location is NOT null and who has service.pid. When the update() method of the Configuration object is called, CM must do the followings:

1. CM updates the configuration directory.
2. CM lists up all registered ManagedService-s with target service.pid.
3. For each services listed,
   a Identify a target bundle, who registered the service.
   b If the any of the following conditions is met, CM calls back updated method of the ManagedService;
      1. If the Java Runtime Environment does not support permissions,
      2. If the target bundle has the exact same bundle location as the Configuration object's location.
      3. If the configuration has null location.
      4. If the target bundle has appropriate ConfigurationPermission to be bound to the location,
         • In other words, if Bundle#hasPermission(new ConfigurationPermission( the location of the Configuration object, BOUND) ) returns true.

The Fig 2 shows how it works for the case of 3. b. 4.

Remark that it is backward compatible due to 3. b. 2. In that case, no Permission check will be required.
Fig 1: Sequence for configuration object bound to multiple bundles (Configuration.update() is called after ManagedServices are registered)

Fig 3: Sequence for configuration object bound to multiple bundles (ManagedServices are registered after Configuration object is created)
Same thing can be defined by a Configuration object is created by the call of ConfigurationAdmin#createFactoryConfiguration(factoryPid, location).

5.2.2.2 Thread Issue for calling back ManagedService(Factory)#updated()

Fig 1 and Fig 3 show a sequence of process including registering ManagedService object and calling back updated() method of ManagedService.

Fig 1 shows the case that a Configuration object created by getConfiguration(pid, location) is updated after two target ManagedServices have been registered. In this case, Configuration Admin must start another thread and call updated() method of all target ManagedServices on othin the thread. If multiple target ManagedServices exist, the calling order of those updated method is Configuration Admin MUST call updated() methods in the following order; firstly, descending order of SERVICE_RANKING, and if multiple service with the same SERVICE_RANKING exists, among them, secondly ascending order of SERVICE_ID.

Fig 3 shows the case that ManagedService 1 and Managed Service 2 are registered respectively after the Configuration object has been created. In this case, Configuration Admin must run the thread calling updated() method and calls back, respectively.

In case of using ManagedServiceFactory, the process is done in the same way.

5.2.3 Callback Operation Lists

ManagedService(MS) callbacks will be done only in the following cases:
1. When Configuration#update() of the configuration is called after MS-s are registered, ONLY and each updated() of MS-s registered with the pid by target bundles, which the configuration can be bound to, will be called back.

2. When bundleTB1 registers a MS with pid1, the MS MUST be called back regardless of the permissions and props.

3. When service pid of registered MS is modified, the MS MUST be called back regardless of the permissions and props as 2.

4. When Configuration#delete() is called, ONLY and each updated() of the MS-s registered with the pid by target bundles, which the configuration can be bound to, will be called back with null.

5. When CM starts, do as 2 for each MS-s registered at the moment.

ManagedServiceFactory(MSF) callbacks will be done only in the following cases:

1. When Configuration#update() of the factory configuration is called after MSF-s are registered, ONLY and each updated() of MSF-s registered with the factory pid by target bundles, which the factory configuration can be bound to, will be called back.

2. When bundleTB1 registers MSF with the factory pid, the MSF MUST be called back for the factory configuration stored at the moment regardless of permissions and props.

3. When service pid (which represents factory pid for factory configuration) of registered MSF is modified, the MSF MUST be called back regardless of the permissions and props.

4. When Configuration#deleted() of the factory configuration is called, ONLY and each deleted() of MSF-s registered with the factory pid by target bundles, which the factory configuration can be bound to, will be called back.

5. When CM starts, do as 2 for each MSF-s registered at the moment.

For MS and MSF, CM judges whether the configuration can be bound to the target bundle" by whether any of the following conditions is met:

- if the Java Runtime does not support permissions,
- if it supports permissions and the location of the Configuration is null, or
- if it supports permissions, the location of the Configuration is not null, and if the target bundle has appropriate ConfigurationPermission[the location of the Configuration, "BOUND"].

[Remark1] Any of the followings will NOT trigger callback of ManagedService(Factory) at all, at the moment.

- ConfigurationAdmin#getConfiguration and #createFactoryConfiguration.
- Configuration#setBundleLocation.
- Permissions change of target bundles.

5.2.4 Configurations with null location

Dynamic binding defined in version 1.3, or earlier, is assumed as old way in the new Spec. In the proposal, a configuration with null location is dealt with as follows:

In case that firstly a configuration with null location appears and then ManagedService(Factory) is registered,

- The ManagedService(Factory) will be called back with the properties, which might be null, regardless of permissions granted to the target bundle which registers the service.
• The location of the configuration will keep null.

In case that firstly there exists one or more ManagedService(Factory)s registered and then a configuration with null location appears,
• All ManagedService(Factory) will be called back with the properties, which might be null, regardless of permissions granted to the target bundle which registers the service.
• The location of the configuration will keep null.

For both cases, “a configuration with null location appears” can be caused by calling any of
• ConfigurationAdmin#getConfiguration(pid, null),
• ConfigurationAdmin#createFactoryConfiguration(pid, null), or
• Configuration#setBundleLocation(null).

Description in 5.2.3 includes how the configurations with null location should be dealt with.

5.3 Permission Check Operations of the caller

5.3.1 Permission Check Operation to get configuration by location

Let us assume that bundle A calls ConfigurationAdmin#getConfiguration(pid, location).

<table>
<thead>
<tr>
<th>Case</th>
<th>location is null</th>
<th>location is NOT null</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case1</td>
<td>A Configuration with the PID has already existed. Assume the location of it is locationOld (is NOT null)</td>
<td>Case2</td>
</tr>
<tr>
<td>Case3</td>
<td>A Configuration with the PID has already existed. Assume the location of it is null</td>
<td>Case4</td>
</tr>
<tr>
<td>Case5</td>
<td>A Configuration with the PID did NOT exist yet.</td>
<td>Case6</td>
</tr>
</tbody>
</table>

Table 1: Cases to be taken into account for getConfiguration(pid, location).

5.3.1.1 Case1 and Case2
If the Java Runtime Environment does not support permissions or, if it supports permissions and the caller bundle has ConfigurationPermission(locationOld, CONFIGURE), the existed Configuration object with locationOld will be returned. The specified location in the method will be ignored.

Otherwise, SecurityException will be thrown.

5.3.1.2 Case3 and Case4
If the Java Runtime Environment does not support permissions or, if it supports permissions and the caller bundle has ConfigurationPermission("*", CONFIGURE), the existed Configuration object with null will be returned. The specified location in the method will be ignored.

Otherwise, SecurityException will be thrown.

[REMARK] Even in Case4 (location is NOT null), the location is ignored as the current spec (version 1.3).

5.3.1.3 Case5
If the Java Runtime Environment does not support permissions or, if it supports permissions and the caller bundle has ConfigurationPermission("*", CONFIGURE), a new Configuration object with null location will be returned.
Otherwise, SecurityException will be thrown.

5.3.1.4 Case6
If the Java Runtime Environment does not support permissions or, if it supports permissions and the caller bundle has ConfigurationPermission(\texttt{location}, CONFIGURE), a new Configuration object with the specified location \texttt{location} will be returned.
Otherwise, SecurityException will be thrown.

5.3.2 Permission Check Operation to get configuration by only pid
Let us assume that and bundle A, who has bundle location “\texttt{locationA}”, calls ConfigurationAdmin\#getConfiguration(pid).

<table>
<thead>
<tr>
<th>Case</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case1</td>
<td>A Configuration with the PID has already existed. Assume the location of it is null</td>
</tr>
<tr>
<td>Case2</td>
<td>A Configuration with the PID has already existed. Assume the location of it equals \texttt{locationOld}</td>
</tr>
<tr>
<td>Case3</td>
<td>A Configuration with the PID has already existed. Assume the location of it is neither null nor \texttt{locationA} but \texttt{locationOld}</td>
</tr>
<tr>
<td>Case4</td>
<td>A Configuration with the PID does NOT exist yet.</td>
</tr>
</tbody>
</table>

Table 2: Cases to be taken into account for getConfiguration(pid).

5.3.2.1 Case1
Either if the Java Runtime Environment supports permissions or not, CM will return the Configuration object, whose location still keeps null.

5.3.2.2 Case2
Either if the Java Runtime Environment supports permissions or not, the existed Configuration object will be returned.

5.3.2.3 Case3
1. If the Java Runtime Environment does not support permissions or, if it supports permissions and the caller bundle has ConfigurationPermission(\texttt{locationOld}, CONFIGURE), the existed Configuration object with \texttt{locationOld} will be returned.
Otherwise, SecurityException will be thrown.

5.3.2.4 Case4
Either if the Java Runtime Environment supports permissions or not, CM will create new Configuration object with the location which is set to the caller bundle’s bundle location “\texttt{locationA}” and return the Configuration object.

5.3.3 Permission Check Operation to get factory configuration by location
Let us assume that bundle A calls createFactoryConfiguration(factoryPid, location).
### 5.3.3.1 Case1
If the Java Runtime Environment does not support permissions or, if it supports permissions and the caller bundle has ConfigurationPermission("*", CONFIGURE), new Configuration object will be created and returned. The returned Configuration object will have the automatic property of the specified factoryPid and will have the location of \texttt{null}.
Otherwise, SecurityException will be thrown.

### 5.3.3.2 Case2
If the Java Runtime Environment does not support permissions or if it supports permissions and the caller bundle has ConfigurationPermission("location", CONFIGURE), new Configuration object will be created and returned. The returned Configuration object will have the automatic property of the specified factoryPid and will have the location of \texttt{location}.
Otherwise, SecurityException will be thrown.

### 5.3.4 Permission Check Operation to get factory configuration by only factoryPid
Let us assume that bundle \texttt{A} calls ConfigurationAdmin\#createFactoryConfiguration(factoryPid).

CM will create a new Configuration object with a new PID and return it. The Configuration object will have the automatic property of the specified factoryPid and be bound to the calling bundle's location. (The Configuration object will have the location of bundle \texttt{A}.)

### 5.3.5 Permission Check Operation to list configurations
Let us assume that bundle \texttt{A} calls ConfigurationAdmin\#listConfigurations(\texttt{filter}).

Firstly CM will list up a set of all Configuration objects which the filter matches.

Secondly, for each Configuration object of them, CM will check whether the Configuration object can be returned by ConfigurationAdmin\#getConfiguration(pid). If not possible, the object will be removed from the set.

Finally, return the set.

It is similar that BundleContext\#getServiceReferences(String clazz, String filter) returns filtered ServiceReference set.

c.f) Section 6.1.6.14 in core specification version 1.4 describes this method operation. It says:

*If the Java Runtime Environment supports permissions, the set of ServiceReference objects produced by the previous step is reduced by checking that the caller has the ServicePermission to get at least one of the class names under which the service was registered. If the caller does not have the correct permission for a particular ServiceReference object, then it is removed from the set.*

### 5.3.6 Permission Check Operation to get bundle location of a Configuration
Let us assume that a Bundle \texttt{A} calls Configuration\#getBundleLocation() for the Configuration object of which bundle location equals \texttt{location}.
If the Java Runtime Environment does not support permissions, or if it supports permissions and the caller bundle has appropriate ConfigurationPermission to read the Configuration object, CM will return location of the configuration object. The returned value might be null.

Otherwise, SecurityException will be thrown.

The appropriate Permission to read it is:

- ConfigurationPermission(\textit{location}, \textit{READ}), if \textit{location} is NOT null.
- ConfigurationPermission(\textit{**}, \textit{READ}), if \textit{location} is null.

### 5.3.7 Permission Check Operation to set bundle location of a configuration

Let us assume that Bundle A calls Configuration#setBundleLocation(\textit{location}) for the Configuration object of which bundle location equals \textit{locationOld}.

If the Java Runtime Environment does not support permissions, or if it supports permissions and the caller bundle has appropriate ConfigurationPermission to configure ONLY the Configuration object with \textit{location}, CM will set \textit{location} to the location of the Configuration object.

Otherwise, SecurityException will be thrown.

[REMARK] In order to get Configuration object, the caller bundle must be required to have appropriate ConfigurationPermission for \textit{locationOld}. Therefore, setBundleLocation does not check permission for \textit{locationOld}.

The appropriate ConfigurationPermission for each case are as follows.

<table>
<thead>
<tr>
<th>\textit{locationOld} is null</th>
<th>\textit{location is null}</th>
<th>\textit{location is NOT null}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case1</td>
<td></td>
<td>Case2</td>
</tr>
<tr>
<td>LocationOld is NOT null</td>
<td>Case3</td>
<td>Case4</td>
</tr>
</tbody>
</table>

Case1 and Case3: ConfigurationPermission(\textit{**}, \textit{CONFIGURE}).

Case2 and Case4: ConfigurationPermission(\textit{location}, \textit{CONFIGURE}).

### 5.4 Clarification of the spec

ConfigurationAdmin#CreateFactoryConfiguration() always returns newly created factory configuration.

### 5.5 Use cases

#### 5.5.1 Use case 1: General

Let us assume that the Java Runtime Environment supports permissions and bundle A calls ConfigurationAdmin#getConfiguration(pid, "opaque1.a.").

If bundle A has ConfigurationPermission("opaque1.a.", \textit{CONFIGURE}), it will succeed.

If bundle A has ConfigurationPermission("opaque1.a.b", \textit{CONFIGURE}), it will fail.

If bundle A has ConfigurationPermission("opaque1.a.", \textit{CONFIGURE}), it will fail.
If bundle A has ConfigurationPermission("opaque1.c.*", CONFIGURE), it will fail.

Then bundle B registers a ManageService with the pid.

If bundle B has ConfigurationPermission("opaque1.a.*", BOUND), the ManagedService will be called back.

If bundle B has ConfigurationPermission("opa*", BOUND), the ManagedService will be called back.

If bundle B has ConfigurationPermission("opaque1.a.b", BOUND), the ManagedService will NOT be called back.

If bundle B has ConfigurationPermission("opaque1.a.", BOUND), the ManagedService will NOT be called back.

If bundle B has ConfigurationPermission("opaque1.c.*", BOUND), the ManagedService will NOT be called back.

Here, bundle A passes the Configuration object to bundle C. Then bundle C calls getBundleLocation() method of the Configuration object.

If bundle C has ConfigurationPermission("opaque1.a.*", READ), the method will return the location.

If bundle B has ConfigurationPermission("opa*", READ), the method will return the location.

If bundle B has ConfigurationPermission("opaque1.a.b", READ), the method will throw SecurityException.

If bundle B has ConfigurationPermission("opaque1.a.", READ), the method will throw SecurityException.

If bundle B has ConfigurationPermission("opaque1.c.*", READ), the method will throw SecurityException.
5.5.2 Multiple Service Provider Model Use Case 1

Let us assume that Multiple Service Provider Model [3]. In the model, bundles developed by multiple service providers run on one framework. Fig 7 shows an example of typical case.

TB1 and CB1 are developed by service provider “jp.co.ntt”. Therefore, MA(Management Agent) decides both belong to “segment:jp.co.ntt.”.

The other bundles are developed by service provider “com.acme”. All of them belong to “segment:com.acme.”.

The names of each segment are used as location of a Configuration object. C1 and C2 has location of “segment:jp.co.ntt.” and “segment:com.acme.”, respectively. MA gives each bundle the following ConfigurationPermission(CP), respectively.

CB1:CP("segment:jp.co.ntt.",CONFIGURE)
TB1:CP("segment:jp.co.ntt.",BOUND)
CB2:CP("segment:com.acme.",CONFIGURE)
TB2, TB3, TB4:CP("segment:com.acme.",BOUND)

In this case, complete separation are realized between segment.
5.5.3 Multiple Service Provider Model Use Case2

Fig 5 shows another example of Multiple Service Provider Model.

In addition to previous case, the following permissions are given to each bundle:

- TB1:CP("segment:com.acme.*",BOUND) and
- CB3:CP("segment:jp.co.ntt.*",CONFIGURE).

Due to the additional permissions, CB3 can configure both C1 and C3, and TB1 can be bound to C2 beyond the segment.

5.5.4 Multiple Service Provider Model Use Case3

In the case of Fig 5, both Configuration object C1 and C3 can be configured CB3. Fig 6 shows another example of Multiple Service Provider Model, which realizes more fine grained control beyond segment.

While CB1 and TB1 has CP("segment:jp.co.ntt.*", CONFIGURE/BOUND), set different location for C1 and C3: "segment:jp.co.ntt.a" and "segment:jp.co.ntt.b". CB1 succeeds in configuring both C1 and C3, and TB1 succeeds in being bound to both C1 and C2 because CP's name "segment:jp.co.ntt.*" matches both "segment:jp.co.ntt.a" and "segment:jp.co.ntt.b".
Here, MA gives CB3 not CP("segment:jp.co.ntt.b",CONFIGURE) but only CP("segment:jp.co.ntt.a",CONFIGURE). Therefore, CB3 can configure not C3 but C1. TB4 can be bound to C2 as well.

As you see, this is an effective use case.

- Default separation is realized by simple ConfigurationPermission management: giving ConfigurationPermission with name starting with each segment's prefix and ending with "*".
- Each Configuring bundle in a segment creates a Configuration object with the prefix of the segment and finer grained suffix.
- If a platform manager would like to allow limited access between segments, giving ConfigurationPermission with fine grained name to a bundle, which is allowed to configure or be bound beyond segments, realizes it.

5.5.5 Use Case 4: Traditional Location Usage with Limited Access to Configure

In the case of Fig 7, a location of a Configuration object is set in traditional way: location is set to the bundle location of the bundle to be bound.

- CB1 can configure Both C1 and C2.
- CB2 can configure not C2 but only C1.
- CB3 can configure neither C1 nor C2.
- Although TB1 does not have any ConfigurationPermission, TB1 can configure C1 and be bound to C1 because the location of C1 equals the bundle location of TB1.

Fig 7: Case 4 Traditional Location Usage with Limited Access to Configure
5.6 In case that security is disabled

In this section, let us assume that the Java Runtime Environment does not support permissions.

Regarding configure side, there is no difference between the proposal and Spec 1.3: any bundle can configure any Configurations by only specifying service (factory) pid.

On the other hand, regarding bound side, there is a difference between two.

Think about there exists a Configuration object with location except null. In the proposal, any bundle can be bound to the Configuration as long as the specified service pid matches (Fig 8). In Spec 1.3, only bundle whose bundle location equals the location of the Configuration object can be bound to it (Fig 9).

c.f. if a bundle registers multiple ManagedService with the pid, all of them will be called back in both the proposal and Spec 1.3 (no difference between two).

Fig 8: No Permission Supported Case in the Proposal
5.7 Java doc

6 Javadoc

There is no new interfaces, classes, and methods. Only exception is new actions of ConfigurationPermission.

The javadocs of the following interfaces and class will be updated:

- ConfigurationPermission
- ConfigurationAdmin
- Configuration
- ManagedService
- ManagedServiceFactory

The only changed part are written in this section, especially in red.

6.1 Package org.osgi.service.cm Description
Configuration Admin Package Version 1.4.

Bundles wishing to use this package must list the package in the Import-Package header of the bundle's manifest. For example:
6.2  org.osgi.service.cm

Interface Configuration

public interface Configuration

The configuration information for a ManagedService or ManagedServiceFactory object. The Configuration Admin service uses this interface to represent the configuration information for a ManagedService or for a service instance of a ManagedServiceFactory.

A Configuration object contains a configuration dictionary and allows the properties to be updated via this object. Bundles wishing to receive configuration dictionaries do not need to use this class - they register a ManagedService or ManagedServiceFactory. Only administrative bundles, and bundles wishing to update their own configurations need to use this class.

The properties handled in this configuration have case insensitive String objects as keys. However, case is preserved from the last set key/value.

A configuration can be bound to multiple bundles since Version 1.4. If the Java Runtime supports permissions, the binding is determined by permissions a bundle have. If the bundle has appropriate ConfigurationPermission to be bound to, the configuration will be passed to the updated method of the corresponding ManagedService object or ManagedServiceFactory object.

If a configuration's location is null, it is not yet bound to a location. While version 1.3 supports dynamic binding, version 1.4 does not. A location of an existing configuration can be changed only by setBundleLocation(java.lang.String) method.

The same Configuration object is used for configuring both a Managed Service Factory and a Managed Service. When it is important to differentiate between these two the term "factory configuration" is used.

Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.lang.String getBundleLocation()</td>
<td>Get the bundle location.</td>
</tr>
<tr>
<td>java.lang.String getFactoryPid()</td>
<td>For a factory configuration return the PID of the corresponding Managed Service Factory, else return null.</td>
</tr>
<tr>
<td>java.lang.String getPid()</td>
<td>Get the PID for this Configuration object.</td>
</tr>
<tr>
<td>java.util.Dictionary getProperties()</td>
<td>Return the properties of this Configuration object.</td>
</tr>
<tr>
<td>int hashCode()</td>
<td>Hash code is based on PID.</td>
</tr>
<tr>
<td>void setBundleLocation(java.lang.String bundleLocation)</td>
<td>Set the location of this Configuration object to the specified location.</td>
</tr>
<tr>
<td>void update()</td>
<td>Update the Configuration object with the current properties.</td>
</tr>
<tr>
<td>void update(java.util.Dictionary properties)</td>
<td></td>
</tr>
</tbody>
</table>
Method Detail

6.2.1 setBundleLocation

void setBundleLocation(java.lang.String bundleLocation)

Set the location of this Configuration object to the specified location. The specified location might be null. If the Java Runtime Environment supports permissions and the caller does not have appropriate ConfigurationPermission, SecurityException will be thrown. Here, the appropriate permission to be required is ConfigurationPermission[*,CONFIGURE] if the specified bundleLocation is null. Otherwise, it is ConfigurationPermission[the specified bundleLocation, CONFIGURE].

Parameters:
  bundleLocation - a location or null

Throws:
  java.lang.IllegalArgumentException - If this configuration has been deleted.
  java.lang.SecurityException - If the caller does not have appropriate ConfigurationPermission.

6.2.2 getBundleLocation

java.lang.String getBundleLocation()

Get the bundle location. Returns the bundle location to which this configuration has. Returned value might be null.

If the Java Runtime Environment supports permissions and the caller bundle does not have appropriate ConfigurationPermission, SecurityException will be thrown. Here, the appropriate permission to be required is ConfigurationPermission[*,READ] if the location of this Configuration object is null. Otherwise, it is ConfigurationPermission[the location of this Configuration object, READ].

Returns:
  location to which this configuration is bound, or null.

Throws:
  java.lang.IllegalArgumentException - If this Configuration object has been deleted.
  java.lang.SecurityException - If the caller does not have appropriate ConfigurationPermission.

6.3 org.osgi.service.cm

Interface ConfigurationAdmin
public interface ConfigurationAdmin

Service for administering configuration data.

The main purpose of this interface is to store bundle configuration data persistently. This information is represented in Configuration objects. The actual configuration data is a Dictionary of properties inside a Configuration object.

There are two principally different ways to manage configurations. First there is the concept of a Managed Service, where configuration data is uniquely associated with an object registered with the service registry.

Next, there is the concept of a factory where the Configuration Admin service will maintain 0 or more Configuration objects for a Managed Service Factory that is registered with the Framework.

The first concept is intended for configuration data about "things/services" whose existence is defined externally, e.g. a specific printer. Factories are intended for "things/services" that can be created any number of times, e.g. a configuration for a DHCP server for different networks.

Bundles that require configuration should register a Managed Service or a Managed Service Factory in the service registry. A registration property named service.pid (persistent identifier or PID) must be used to identify this Managed Service or Managed Service Factory to the Configuration Admin service.

When the ConfigurationAdmin detects the registration of a Managed Service, it checks its persistent storage for a configuration object whose PID matches the PID registration property (service.pid) of the Managed Service. If found and the registering bundle of the service has appropriate permissions to be bound to, it calls ManagedService.updated(java.util.Dictionary) method with the new properties. Otherwise, ManagedService.updated(java.util.Dictionary) method with null. The implementation of a Configuration Admin service must run these call-backs asynchronously to allow proper synchronization.

When the Configuration Admin service detects a Managed Service Factory registration, it checks its storage for configuration objects whose factoryPid matches the PID of the Managed Service Factory. For each such Configuration objects, if the registering bundle has appropriate permissions to be bound to, it calls the ManagedServiceFactory.updated method asynchronously with the new properties. The calls to the updated method of a ManagedServiceFactory must be executed sequentially and not overlap in time.

In general, bundles having permission to use the Configuration Admin service can only access and modify their own configuration information. Accessing or modifying the configuration of another bundle requires appropriate ConfigurationPermission.

The method descriptions of this class refer to a concept of "the calling bundle". This is a loose way of referring to the bundle which obtained the Configuration Admin service from the service registry. Implementations of ConfigurationAdmin must use a org.osgi.framework.ServiceFactory to support this concept.

---

**Method Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration.createFactoryConfiguration(java.lang.String factoryPid)</td>
<td>Create a new factory Configuration object with a new PID.</td>
</tr>
<tr>
<td>Configuration.createFactoryConfiguration(java.lang.String factoryPid, java.lang.String location)</td>
<td>Create a new factory Configuration object with a new PID.</td>
</tr>
<tr>
<td>Configuration.getConfiguration(java.lang.String pid)</td>
<td>Get an existing or new Configuration object from the persistent store.</td>
</tr>
<tr>
<td>Configuration.getConfiguration(java.lang.String pid, java.lang.String location)</td>
<td>Get an existing Configuration object from the persistent store, or create a new Configuration object.</td>
</tr>
<tr>
<td>Configuration.listConfigurations(java.lang.String filter)</td>
<td></td>
</tr>
</tbody>
</table>
6.3.1 createFactoryConfiguration

Configuration createFactoryConfiguration(java.lang.String factoryPid)
throws java.io.IOException

Create a new factory Configuration object with a new PID. The properties of the new Configuration object are null until the first time that its Configuration#update(Dictionary) method is called.

It is not required that the factoryPid maps to a registered Managed Service Factory.

The location of the Configuration object will be set to the location of the calling bundle.

Parameters:
   factoryPid - PID of factory (not null).

Returns:
   A new Configuration object.

Throws:
   java.io.IOException - if access to persistent storage fails.

6.3.2 createFactoryConfiguration

Configuration createFactoryConfiguration(java.lang.String factoryPid, java.lang.String location)
throws java.io.IOException

Create a new factory Configuration object with a new PID. The properties of the new Configuration object are null until the first time that its Configuration#update(Dictionary) method is called.

It is not required that the factoryPid maps to a registered Managed Service Factory.

If the Java Runtime Environment does not support permissions or, if it supports permissions and the caller bundle has appropriate ConfigurationPermission, a new Configuration object will be created and returned. The returned Configuration object will have the automatic property of the specified factoryPid and will have the specified location. Otherwise, SecurityException will be thrown. Here, the appropriate permission to be required is ConfigurationPermission[*,CONFIGURE] if the specified location is null. Otherwise, it is ConfigurationPermission[the specified location,CONFIGURE].

Parameters:
   factoryPid - PID of factory (not null).
   location - A location string, or null.

Returns:
   a new Configuration object.

Throws:
   java.io.IOException - if access to persistent storage fails.
6.3.3 getConfiguration

Configuration getConfiguration(java.lang.String pid, java.lang.String location) throws java.io.IOException

Get an existing Configuration object from the persistent store, or create a new Configuration object.

If a Configuration with this PID already exists in Configuration Admin service, and if the Java Runtime Environment does not support permissions or if it supports permissions and the caller bundle has appropriate ConfigurationPermission, the existing Configuration will be returned. The location parameter is ignored in this case. Otherwise, SecurityException will be thrown. Here, the appropriate permission to be required is ConfigurationPermission[*\*,CONFIGURE] if the location of the existing Configuration is null. Otherwise, it is ConfigurationPermission[location of the existing Configuration,CONFIGURE].

If a Configuration with this PID does not exist in Configuration Admin service, and if the Java Runtime Environment does not support permissions or if it supports permissions and the caller bundle has appropriate ConfigurationPermission, a new Configuration object will be created and returned. The location of the new object is set to the specified location and the properties are set to null. Otherwise, SecurityException will be thrown. Here, if the specified location is null, appropriate permission to be required is ConfigurationPermission[*\*,CONFIGURE]. Otherwise, it is ConfigurationPermission[the specified location,CONFIGURE].

Parameters:
- pid - Persistent identifier.
- location - The location string, which might be null.

Returns:
An existing or new Configuration object.

Throws:
- java.io.IOException - if access to persistent storage fails.
- java.lang.SecurityException - if the specified location the caller does not have appropriate ConfigurationPermission to configure the Configuration object.

6.3.4 getConfiguration

Configuration getConfiguration(java.lang.String pid) throws java.io.IOException

Get an existing or new Configuration object from the persistent store. If the Configuration object for this PID does not exist, create a new Configuration object for that PID and return it, where properties are null. Set its location to the calling bundle’s location.

Otherwise, if any of the following conditions is satisfied, the existing Configuration will be returned without changing its location:
• the location of the existing Configuration object is null,
• the location of the existing Configuration object equals the caller’s bundle location,
• the Java Runtime Environment does not support permissions, or
• the Java Runtime supports permissions and the caller bundle has 
  ConfigurationPermission[location of the existing Configuration ,CONFIGURE].

Otherwise, SecurityException will be thrown.

**Parameters:**
  - pid - persistent identifier.

**Returns:**
an existing or new Configuration matching the PID.

**Throws:**
  - java.io.IOException - if access to persistent storage fails.
  - java.lang.SecurityException - if the existing Configuration object has a location which is 
    not null and different from one of the calling bundle and it has no 
    ConfigurationPermission[location of the Configuration object,CONFIGURE].

---

6.3.5 listConfigurations

Configuration[] listConfigurations(java.lang.String filter) 
  throws java.io.IOException, 
  java.lang.SecurityException

List the current Configuration objects which match the filter.

Only Configuration objects with non-null properties are considered current. That is, 
Configuration.getProperties() is guaranteed not to return null for each of the returned 
Configuration objects.

Only Configuration objects that the caller bundle can get by getConfiguration(String) are 
returned.

The syntax of the filter string is as defined in the org.osgi.framework.Filter class. The filter can test 
any configuration parameters including the following system properties:

  • service.pid-String- the PID under which this is registered
  • service.factoryPid-String- the factory if applicable
  • service.bundleLocation-String- the bundle location

The filter can also be null, meaning that all Configuration objects should be returned.

**Parameters:**
  - filter - A filter string, or null to retrieve all Configuration objects.

**Returns:**
  - All matching Configuration objects, or null if there aren't any.

**Throws:**
  - java.io.IOException - if access to persistent storage fails
  - java.lang.SecurityException - if the existing Configuration object has a location which is 
    not null and different from one of the calling bundle and it has no 
    ConfigurationPermission[location of the Configuration object,CONFIGURE].
  - java.lang.SecurityException - if the existing Configuration object has a location which is 
    not null and different from one of the calling bundle and it has no 
    ConfigurationPermission[location of the Configuration object,CONFIGURE].
6.4  org.osgi.service.cm

Class ConfigurationPermission

java.lang.Object
   java.security.Permission
      java.security.BasicPermission
         org.osgi.service.cm.ConfigurationPermission

All Implemented Interfaces:
   java.io.Serializable, java.security.Guard

public final class ConfigurationPermission extends java.security.BasicPermission

Indicates a bundle’s authority to configure a configuration, read location information from a configuration, and be bound a configuration to.
Wildcards can be used in its name.
Examples:
   a.b.c
   a.b.*
   *

ConfigurationPermission has three actions; configure, read, and bound.
- CONFIGURE action allows a bundle to configure a configuration with the specified location.
- READ action allows a bundle to read bundle location of a configuration with the specified location.
- BOUND action allows a bundle to be bound a configuration with the specified location to.

CONFIGURE action implies both of READ action and BOUND action.

Since: 1.4
See Also: Serialized Form

---

Field Summary

<table>
<thead>
<tr>
<th>static java.lang.String</th>
<th>BOUND</th>
</tr>
</thead>
<tbody>
<tr>
<td>The action string bound.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static java.lang.String</th>
<th>CONFIGURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>The action string configure.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static java.lang.String</th>
<th>READ</th>
</tr>
</thead>
<tbody>
<tr>
<td>The action string read.</td>
<td></td>
</tr>
</tbody>
</table>

Constructor Summary

ConfigurationPermission(java.lang.String name, java.lang.String actions)
Create a new ConfigurationPermission.
### Method Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean equals (obj)</td>
<td>Determines the equality of two ConfigurationPermission objects. This method returns true if: the specified object is ConfigurationPermission, the name of the specified permission has the same name, and the action of the specified permission has the same action.</td>
</tr>
<tr>
<td>java.lang.String getActions()</td>
<td>Returns the canonical string representation of the ConfigurationPermission actions.</td>
</tr>
<tr>
<td>boolean implies (p)</td>
<td>Determines if the specified permission is implied by this object.</td>
</tr>
</tbody>
</table>

### Field Detail

#### 6.4.1 CONFIGURE

```java
public static final java.lang.String CONFIGURE
```

The action string `configure`.

**See Also:**
- [Constant Field Values](#)

#### 6.4.2 BOUND

```java
public static final java.lang.String BOUND
```

The action string `bound`.

**See Also:**
- [Constant Field Values](#)

#### 6.4.3 READ

```java
public static final java.lang.String READ
```

The action string `read`.

**See Also:**
- [Constant Field Values](#)
Constructor Detail

6.4.4 ConfigurationPermission

```java
public ConfigurationPermission(java.lang.String name,
                               java.lang.String actions)
```

Create a new ConfigurationPermission.

**Parameters:**
- `name` - name, which represents location.
- `actions` - bound, configure, or read (canonical order).

Method Detail

6.4.5 implies

```java
public boolean implies(java.security.Permission p)
```

Determines if the specified permission is implied by this object.

More specifically, this method returns true if:

- `p`'s class is `ConfigurationPermission`,
- `p`'s name equals or (in the case of wildcards) is implied by this object's name. For example, "a.b.*" implies "a.b.c", and
- `p`'s action is implied by this object's action.

**CONFIGURE** action implies both **READ** action and **BOUND** action.

**Overrides:**
- `implies` in class `java.security.BasicPermission`

**Parameters:**
- `p` - The requested permission.

**Returns:**
- `true` if the specified permission is implied by this object; `false` otherwise.

6.4.6 equals

```java
public boolean equals(java.lang.Object obj)
```

Determines the equality of two `ConfigurationPermission` objects. This method returns true if:

- the specified object is `ConfigurationPermission`,
- the name of the specified permission has the same name, and
- the action of the specified permission has the same action.

**Overrides:**
- `equals` in class `java.security.BasicPermission`
Parameters:
  obj - The object being compared for equality with this object.

Returns:
  true if obj is equivalent to this ConfigurationPermission; false otherwise.

6.4.7 getActions

public java.lang.String getActions()

Returns the canonical string representation of the ConfigurationPermission actions.

Always returns present ConfigurationPermission actions in the following order: BOUND, CONFIGURE, READ,

Overrides:
  getActions in class java.security.BasicPermission

Returns:
  Canonical string representation of the ConfigurationPermission actions.

6.5 org.osgi.service.cm

Interface ManagedService

public interface ManagedService

A service that can receive configuration data from a Configuration Admin service.

A Managed Service is a service that needs configuration data. Such an object should be registered with the Framework registry with the service.pid property set to some unique identifier called a PID.

If the Configuration Admin service has a Configuration object corresponding to this PID, it will callback the updated() method of the ManagedService object, passing the properties of that Configuration object, if the bundle who registered the ManagedService object has appropriate permissions to be bound to.

If it has no such Configuration object, then it calls back with a null properties argument. Registering a Managed Service will always result in a callback to the updated() method provided the Configuration Admin service is, or becomes active. This callback must always be done asynchronously.

Else, every time that either of the updated() methods is called on that Configuration object, the ManagedService.updated() method with the new properties is called, if the bundle who registered the ManagedService object has appropriate permissions to be bound to. When the delete() method is called on that Configuration object, ManagedService.updated() is called with a null for the properties parameter if the bundle who registered the ManagedService object has appropriate permissions to be bound to. All these callbacks must be done asynchronously.

The following example shows the code of a serial port that will create a port depending on configuration information.

    class SerialPort implements ManagedService {
ServiceRegistration registration;
Hashtable configuration;
CommPortIdentifier id;

synchronized void open(CommPortIdentifier id, BundleContext context) {
    this.id = id;
    registration = context.registerService(
        ManagedService.class.getName(),
        this,
        getDefaults()
    );
}

Hashtable getDefaults() {
    Hashtable defaults = new Hashtable();
    defaults.put( "port", id.getName() );
    defaults.put( "product", "unknown" );
    defaults.put( "baud", "9600" );
    defaults.put( Constants.SERVICE_PID,
        "com.acme.serialport." + id.getName() );
    return defaults;
}

public synchronized void updated(
    Dictionary configuration ) {
    if ( configuration ==
        null
    )
        registration.setProperties( getDefaults() );
    else {
        setSpeed( configuration.get("baud") );
        registration.setProperties( configuration );
    }
} ...

As a convention, it is recommended that when a Managed Service is updated, it should copy all the properties it does not recognize into the service registration properties. This will allow the Configuration Admin service to set properties on services which can then be used by other applications.

6.6 org.osgi.service.cm

Interface ManagedServiceFactory

public interface ManagedServiceFactory

Manage multiple service instances. Bundles registering this interface are giving the Configuration Admin service the ability to create and configure a number of instances of a service that the implementing bundle can provide.
For example, a bundle implementing a DHCP server could be instantiated multiple times for different interfaces using a factory.

Each of these service instances is represented, in the persistent storage of the Configuration Admin service, by a factory Configuration object that has a PID. When such a Configuration is updated, the Configuration Admin service calls the ManagedServiceFactory updated method with the new properties, if the bundle who registered the ManagedServiceFactory object has appropriate permissions to be bound to. When updated is called with a new PID, the Managed Service Factory should create a new factory instance based on these configuration properties. When called with a PID that it has seen before, it should update that existing service instance with the new configuration information.

In general it is expected that the implementation of this interface will maintain a data structure that maps PIDs to the factory instances that it has created. The semantics of a factory instance are defined by the Managed Service Factory. However, if the factory instance is registered as a service object with the service registry, its PID should match the PID of the corresponding Configuration object (but it should not be registered as a Managed Service).

An example that demonstrates the use of a factory. It will create serial ports under command of the Configuration Admin service.

class SerialPortFactory
  implements ManagedServiceFactory {
    ServiceRegistration registration;
    Hashtable ports;
    void start(BundleContext context) {
      Hashtable properties = new Hashtable();
      properties.put( Constants.SERVICE_PID,
                     "com.acme.serialportfactory ");
      registration = context.registerService( 
        ManagedServiceFactory.class.getName(),
        this,
        properties
      );
    }
    public void updated( String pid,
                         Dictionary properties ) {
      String portName = (String) properties.get("port");
      SerialPortService port =
        (SerialPort) ports.get( pid );
      if ( port == null ) {
        port = new SerialPortService();
        ports.put( pid, port );
        port.open();
      }
      if ( port.getPortName().equals(portName) )
        return;
      port.setPortName( portName );
    }
    public void deleted( String pid ) {
      SerialPortService port =
        (SerialPort) ports.get( pid );
      port.close();
      ports.remove( pid );
    }
    ...
  }

Copyright © NTT Corporation 2009 All Rights Reserved
7 Considered Alternatives

Nothing to write.

8 Security Considerations

The proposal heavily depends on permissions. If the Java Runtime does not support permissions, grouping by

9 Document Support

9.1 References


9.2 Author’s Address
9.3 Acronyms and Abbreviations

9.4 End of Document
RFC0145 - Home Gateway Administration

Draft

25 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

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 Document Information</td>
<td>2</td>
</tr>
<tr>
<td>0.1 Table of Contents</td>
<td>2</td>
</tr>
<tr>
<td>0.2 Terminology and Document Conventions</td>
<td>3</td>
</tr>
<tr>
<td>0.3 Revision History</td>
<td>3</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>4</td>
</tr>
<tr>
<td>2 Application Domain</td>
<td>5</td>
</tr>
<tr>
<td>3 Problem Description</td>
<td>5</td>
</tr>
<tr>
<td>4 Requirements</td>
<td>7</td>
</tr>
<tr>
<td>4.1 Functional Requirements</td>
<td>7</td>
</tr>
<tr>
<td>4.2 Non-Functional Requirements</td>
<td>7</td>
</tr>
<tr>
<td>5 Technical Solution</td>
<td>8</td>
</tr>
<tr>
<td>5.1 Scope</td>
<td>8</td>
</tr>
<tr>
<td>5.2 General Considerations</td>
<td>8</td>
</tr>
<tr>
<td>5.3 Roles and Functional Blocks</td>
<td>10</td>
</tr>
<tr>
<td>5.1 DMT Subtree Admin</td>
<td>11</td>
</tr>
<tr>
<td>5.1.1 DMT Admin restrictions</td>
<td>11</td>
</tr>
<tr>
<td>5.1.2 Managing and Mapping DMT Admin subtrees</td>
<td>12</td>
</tr>
<tr>
<td>5.1.3 Mapping Rules Examples</td>
<td>15</td>
</tr>
<tr>
<td>5.1.4 Additional expected behaviour when using DMT Subtree Admin</td>
<td>16</td>
</tr>
<tr>
<td>5.1.5 Raising Events</td>
<td>17</td>
</tr>
<tr>
<td>5.1.6 Event Mapping Examples</td>
<td>19</td>
</tr>
<tr>
<td>5.2 Mapping for an Internet Gateway Device</td>
<td>19</td>
</tr>
<tr>
<td>5.2.1 Registering IGD related Data Plugins</td>
<td>19</td>
</tr>
<tr>
<td>5.2.2 Remote Management Events and Notifications</td>
<td>20</td>
</tr>
<tr>
<td>5.3 Mapping rules to DMT Admin Management Tree</td>
<td>20</td>
</tr>
<tr>
<td>5.3.1 Mapping from Broadband Forum TR-069 and TR-098 to DMT</td>
<td>20</td>
</tr>
<tr>
<td>5.3.2 Mapping from UPnP IGD 1.0</td>
<td>21</td>
</tr>
<tr>
<td>5.3.3 Examples</td>
<td>22</td>
</tr>
<tr>
<td>6 Considered Alternatives</td>
<td>22</td>
</tr>
<tr>
<td>6.1 Not using DMT Admin</td>
<td>22</td>
</tr>
<tr>
<td>7 Security Considerations</td>
<td>22</td>
</tr>
</tbody>
</table>
8 Document Support
8.1 References
8.1 Author’s Address
8.2 Acronyms and Abbreviations
8.3 End of Document

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 8.1.

Source code is shown in this typeface.

0.3 Revision History

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

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial 0.5</td>
<td>Mai 29th 2009</td>
<td>Initial version for review. Andreas Kraft</td>
</tr>
<tr>
<td>0.6</td>
<td>June 3rd 2009</td>
<td>Included changes according to the audio conference on June 2nd 2009 and the first review of the document. Andreas Kraft</td>
</tr>
<tr>
<td>0.7</td>
<td>June 4th 2009</td>
<td>Included changes according to comments in the document and e-mails. Clarified session handling. Clarified event handling. Changed security considerations. Andreas Kraft</td>
</tr>
<tr>
<td>0.9</td>
<td>June 5th 2009</td>
<td>Updated illustration 5. Fixed typos. Rephrased description of the use of session.id property. Andreas Kraft</td>
</tr>
<tr>
<td>0.10</td>
<td>October 10th 2009</td>
<td>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</td>
</tr>
<tr>
<td>0.11</td>
<td>Februry 3rd 2010</td>
<td>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</td>
</tr>
</tbody>
</table>
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
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.
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 notified 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.
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] is 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.
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 5.3 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.

Illustration 2: Internet Gateway Device Administration components relationship
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.

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 an 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.

## 5.1 DMT Subtree Admin

This section specifies the operation of the DMT Subtree Admin service

### 5.1.1 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](image)

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.

5.1.2 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:

  - 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, i.e. it starts from the beginning of the DMT tree. Example: “./InternetGatewayDevice/Devices/SomeVendorDevice”. The DMT Subtree Admin takes the first path element after the root node (here: InternetGatewayDevice) as an identifier and associates the tracked Data Plugin to one of its own registered Data Plugins with the property dataRootURIs set to the same value. 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 (then) ignored tracked Data Plugin. In this case the tracked Data Plugin must be associated now.

  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.
Overlapping paths are allowed. 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 Strings. 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 drivers.

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 an overlapping is detected by the DMT Subtree admin, a DmtException with the cause `NODE_ALREADY_EXISTS` is thrown. 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 5.1.4). 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` property. There is currently only one leaf node defined that must be supported by a Vendor Data Plugin that sets this property. See also illustration 7 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 5.1.4 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.
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.

5.1.3 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`.

<table>
<thead>
<tr>
<th>Vendor Data Plugin</th>
<th>Resulting Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataRootURIs</td>
<td>configurationPaths</td>
</tr>
<tr>
<td>./vendor_x/firewall</td>
<td>./InternetGatewayDevice/service/firewall</td>
</tr>
<tr>
<td>./vendor_x/interface/WAN</td>
<td>./InternetGatewayDevice/interfance/WAN</td>
</tr>
</tbody>
</table>

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.

**Illustration 7: Example subtree mapping**

### 5.1.4 Additional expected behaviour 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 a 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.

5.1.5 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 with (the `dataRootURIs` property).

- The `newnodes` property is not set.

The DMT Subtree Admin must handle events as follows:

---

² An example for an ADDED event could be when a DHCP server in the IGD registers a new client's MAC address.
• The DMT Subtree Admin is catching all DMT Admin events with the topic of either "info/dmtree/DmtEvent/ADDED", "info/dmtree/DmtEvent/DELETED", or "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, 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 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.
    • The new value is constructed as follows:
      “./” + <the first path element of the mapping DMT Subtree Admin's Data Plugin> + <according value of configurationPaths for that Data Plugin> + <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.
    • The new value is constructed as follows:
      “./” + <the first path element of the mapping DMT Subtree Admin's Data Plugin> + <according value of configurationPaths for that Data Plugin> + <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 8 presents the general relationships between the involved components.
5.1.6 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 5.1.3.

<table>
<thead>
<tr>
<th>Node value of the vendor's driver event</th>
<th>Mapped node value</th>
</tr>
</thead>
<tbody>
<tr>
<td>./vendor_x/firewall/enabled</td>
<td>./InternetGatewayDevice/firewall/enabled</td>
</tr>
<tr>
<td>./vendor_x/interface/WAN/disconnected</td>
<td>./InternetGatewayDevice/interface/WAN/1234/disconnected</td>
</tr>
</tbody>
</table>

*Table 2: Event Mapping Examples*

5.2 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.

5.2.1 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.

5.2.2 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.

5.3 Mapping rules to DMT Admin Management Tree

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

5.3.1 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.
### Table 3: Mapping of data types between TR-069 and DMT Admin

<table>
<thead>
<tr>
<th>TR-069 data type</th>
<th>DMT Admin data type</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>FORMAT_STRING</td>
<td>Created by DmtData(String)</td>
</tr>
<tr>
<td>int</td>
<td>FORMAT_INTEGER</td>
<td>Created by DmtData(int)</td>
</tr>
<tr>
<td>unsignedInt</td>
<td>FORMAT_STRING</td>
<td>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.</td>
</tr>
<tr>
<td>boolean</td>
<td>FORMAT_BOOLEAN</td>
<td>Created by DmtData(boolean)</td>
</tr>
<tr>
<td>dateTime</td>
<td>FORMAT_STRING</td>
<td>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.</td>
</tr>
<tr>
<td>base64</td>
<td>FORMAT_BASE64</td>
<td>Created by DmtData(byte[], boolean)</td>
</tr>
</tbody>
</table>

### 5.3.2 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.

### 5.3.3 Examples

The following table presents an example mapping between TR-098, UPnP IGD 1.0, and the 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 bundle which has enough TopicPermission to receive DMT Events.

8 Document Support

8.1 References

[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
[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
[9]. OSGi Companion Specification, DMT Admin, Events, Section 117.10
[10]. OSGi Companion Specification, DMT Admin, Security, Section 117.12
[12]. OSGi Service Platform Core Specification, Permission Admin Service, Chapter 10

8.1 Author’s Address

<table>
<thead>
<tr>
<th>Name</th>
<th>Vivien Helmut, Andreas Kraft, Andreas Sayegh</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>Deutsche Telekom AG, Laboratories</td>
</tr>
<tr>
<td>Address</td>
<td>Ernst-Reuter-Platz 7, D-10587 Berlin</td>
</tr>
<tr>
<td>Voice</td>
<td>+49 30 8353-58185</td>
</tr>
</tbody>
</table>
### 8.2 Acronyms and Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B2BUA</td>
<td>Back-to-back User Agent</td>
</tr>
<tr>
<td>BBF</td>
<td>Broadband Forum</td>
</tr>
<tr>
<td>CPE</td>
<td>Customer-Premises Equipment</td>
</tr>
<tr>
<td>DMT</td>
<td>Device Management Tree</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HG</td>
<td>Home Gateway</td>
</tr>
<tr>
<td>HGI</td>
<td>Home Gateway Initiative</td>
</tr>
<tr>
<td>IGD</td>
<td>Internet Gateway Device</td>
</tr>
<tr>
<td>ISP</td>
<td>Internet Service Provider</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>NAT</td>
<td>Network Address Translation</td>
</tr>
<tr>
<td>PAT</td>
<td>Port Address Translation</td>
</tr>
<tr>
<td>RMS</td>
<td>Remote Management System</td>
</tr>
<tr>
<td>RTSP</td>
<td>Real-Time Streaming Protocol</td>
</tr>
<tr>
<td>SSID</td>
<td>Service Set Identifier</td>
</tr>
<tr>
<td>VoD</td>
<td>Video on Demand</td>
</tr>
<tr>
<td>VoIP</td>
<td>Voice over IP</td>
</tr>
<tr>
<td>WAN</td>
<td>Wide Area Network</td>
</tr>
<tr>
<td>Wi-Fi</td>
<td>Wireless Fidelity</td>
</tr>
</tbody>
</table>
8.3 End of Document
RFC-149 TR-069 Protocol Implementation Guideline

Draft

22 Pages

Abstract

Different industries are interested in applying OSGi to advance their businesses, in which remote management is a key issue. In the residential area, the TR-069 is the one of the de-facto standard protocol for remote management. The best way to realize remote management based on the TR-069 on OSGi is utilizing DMT Admin service, which has been defined in the OSGi Alliance for the mobile device management. In this case, TR-069 is implemented as a protocol adapter of the DMT Admin. The DMT Admin service, however, has the interfaces inspired by OMA-DM, which is the de-facto standard protocol for mobile area. Although these two protocols, the TR-069 and the OMA-DM, have the similar objectives and functionality, there are several differences between them. This RFC introduces the guideline of the mapping between TR-069 RPCs and the DMT Admin interfaces, and the guideline of data type translation.
# 1 Document Information

## 1.1 Table of Contents

1 Document Information .................................................................................................................. 2  
   1.1 Table of Contents .................................................................................................................. 2  
   1.2 Terminology and Document Conventions .............................................................................. 3  
   1.3 Revision History ................................................................................................................... 3  

2 Introduction ................................................................................................................................. 5  

3 Application Domain ....................................................................................................................... 6  

4 Problem Description ...................................................................................................................... 6  
   4.1 Management agent making use of OSGi standardized Java interfaces ................................. 7  
   4.2 Mobile specification approach .............................................................................................. 7  
   4.2.1 Support for TR-069 notifications ...................................................................................... 7  
   4.3 Conclusion ............................................................................................................................ 9  

5 Requirements ................................................................................................................................ 9  

6 Technical Solution .......................................................................................................................... 9  
   6.1 Basic architecture of TR-069 protocol implementation ......................................................... 10  
   6.2 RPC mapping for TR-069 ..................................................................................................... 11  
   6.2.1 TR-069 CPE RPCs to DMT Admin Interfaces .................................................................. 11  
   6.2.2 DMT Admin Interfaces to TR-069 ACS RPCs ................................................................. 13  
   6.3 Session management ............................................................................................................. 14  
   6.3.1 Opening Session ............................................................................................................... 14  
   6.3.2 Commit for Atomic Session ............................................................................................ 15  
   6.4 Mapping of data types .......................................................................................................... 16  
   6.5 Error code and SOAP Fault ................................................................................................. 17  
   6.5.1 Error code and Fault code mapping ................................................................................. 17  
   1.1 SOAP Fault expression ........................................................................................................ 19  

7 Considered Alternatives ............................................................................................................... 20  
   7.1 Future work for notification handling ................................................................................. 20  
   7.2 Future work for forced inform parameters .......................................................................... 20  

8 Security Considerations ................................................................................................................. 20  

9 Document Support .......................................................................................................................... 20  
   9.1 References ......................................................................................................................... 20  
   9.2 Author's Address ................................................................................................................ 21  
   9.3 Acronyms and Abbreviations .............................................................................................. 21  

Copyright © NTT Corporation 2010 All Rights Reserved
1.2 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.

1.3 Revision History

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

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
</table>
| Initial  | Sep. 16 2008 | Initial Draft
Koya Mori, NTT Corporation, mori.kouya@lab.ntt.co.jp |
| 2nd      | Jan. 20 2009 | 2nd Draft
Generic setter/getter methods are added.
Limitation of notification is described.
Koya Mori, NTT Corporation, mori.kouya@lab.ntt.co.jp |
| 3rd      | Jun. 2 2009 | 3rd Draft
The technical solution has been changed to describe guideline of RPC mapping for TR-069.
Koya Mori, NTT Corporation, mori.kouya@lab.ntt.co.jp
Ikuo Yamasaki, NTT Corporation, yamasaki.ikuo@lab.ntt.co.jp |
| 4th      | Jul. 24 2009 | 4th Draft
The detail of the Notification handling and the RPC mapping is defined.
And the Session Management is added.
Koya Mori, NTT Corporation, mori.kouya@lab.ntt.co.jp
Ikuo Yamasaki, NTT Corporation, yamasaki.ikuo@lab.ntt.co.jp |
<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>5th</td>
<td>Sep. 18 2009</td>
<td>5th Draft</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The number of this RFC has been changed to 148.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The behavior for notification and session management is changed based on the REG F2F discussion.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The error code mapping is added.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Koya Mori, NTT Corporation, <a href="mailto:mori.kouya@lab.ntt.co.jp">mori.kouya@lab.ntt.co.jp</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ikuo Yamasaki, NTT Corporation, <a href="mailto:yamasaki.ikuo@lab.ntt.co.jp">yamasaki.ikuo@lab.ntt.co.jp</a></td>
</tr>
<tr>
<td>1.0.0</td>
<td>Oct. 16 2009</td>
<td>6th Draft</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The number of this RFC has been changed to 149.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Notification handling is changed based on REG discussion.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The operation for forced inform parameter is added.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Koya Mori, NTT Corporation, <a href="mailto:mori.kouya@lab.ntt.co.jp">mori.kouya@lab.ntt.co.jp</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ikuo Yamasaki, NTT Corporation, <a href="mailto:yamasaki.ikuo@lab.ntt.co.jp">yamasaki.ikuo@lab.ntt.co.jp</a></td>
</tr>
<tr>
<td>1.1.0</td>
<td>Jan. 21 2010</td>
<td>7th Draft</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Two implementation models are added to allow the RFC-145.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Forced inform parameters handling mechanism is added.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Notification event definition is changed to use multiple events.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The data mapping is fixed to accommodate the new data types.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Koya Mori, NTT Corporation, <a href="mailto:mori.kouya@lab.ntt.co.jp">mori.kouya@lab.ntt.co.jp</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ikuo Yamasaki, NTT Corporation, <a href="mailto:yamasaki.ikuo@lab.ntt.co.jp">yamasaki.ikuo@lab.ntt.co.jp</a></td>
</tr>
</tbody>
</table>
2 Introduction

Traditionally, fixed telecommunication operators don’t have knowledge about what runs in the customer’s local area network (LAN). They provide connectivity and manage the wide area network (WAN) that provides this connectivity, but they do not know anything about the devices and networks behind the gateway (xDSL mainly) that interconnect WAN and LAN. Recently the need for management of customer networks and devices is increasing in order to make the deployment of new complex services at home (home automation, tele-health, VoIP, IPTV, surveillance, etc) feasible with reasonable costs.

There are two main kinds of devices that need to be managed in operator’s business: those which come from the fixed business managed through TR-069 and standardized by the Broadband FORUM, and those which come from the mobile business, managed through OMA-DM and standardized by the Open Mobile Alliance. The DMT Admin Service specification of OSGi covers the OMA-DM ones. In addition, the design of DMT Admin Service specification potentially allows adaption of other remote management protocols other than the OMA-DM.

The best way to realize remote management based on TR-069 on OSGi is utilizing the DMT Admin service. In this case, TR-069 is implemented as a protocol adapter of the DMT Admin. However, there are several architectural differences between the TR-069 and the OMA-DM, although these two protocols have the similar objectives and functionality.
Therefore, this RFC defines the mapping guideline between the TR-069 and the DMT Admin to use them together. In addition, this RFC defines the handling of notification mechanism and data type translation.

3 Application Domain

Driven by triple play service delivery in the home network, fixed line access service providers have the need to configure home devices to ensure the proper service delivery. Broadband Forum’s CPE WAN Management Protocol (CWMP, alias TR-069) enables them to do this. By using a remote management server (Auto Configuration Server, ACS), they are able to manage TR-069 enabled devices. TR-069 provides them with possibilities to configure parameters, be informed of parameter changes (notification), start diagnostic tools, update firmware, etc.

Similarly, for the mobile world, the OMA defined the OMA-Device Management specification for remote management of mobile devices. OMA-DM offers similar tools to the mobile service providers as TR-069 to fixed line service provider, but OMA-DM is of course tailored to the specifics of the mobile environment.

As OSGi technology offers a flexible software environment for all these different devices, the remote management of the platform is of interest for both fixed and mobile service providers. As such, it should be possible to integrate the remote management of the OSGi platform, and the applications running on top of it, in the existing management infrastructure.

The DMT Admin service with its mobile management tree in the Mobile specification for OSGi R4 standardizes the remote management of an OSGi platform. As it is largely inspired by OMA-DM, it needs to be evaluated for multi protocol support.

4 Problem Description

In a scenario in which service providers offer a growing number of services, to use specific solutions for the management of these services is not the most suitable option. To speed up the deployment of these services, such as triple play, home automation or tele-health, it is essential to offer general management solutions that allow for the management of a large number of services and the flexible life-cycle management of applications.

These devices usually are already managed by a standard protocol, so it makes sense that an OSGi framework, which hosts the services, running on a device could be managed in the same way as the other resources of the device. Of course, the remote management should be fully integrated in the existing remote management solutions of the service provider to avoid duplicating management infrastructure and to increase performance on the devices.
Currently, there are two options in OSGi for remote management:

- create a management agent bundle making use of the Java interfaces,
- create a protocol adapter bundle that interacts with the DMT Admin service, as defined in the OSGi Mobile specification.

### 4.1 Management agent making use of OSGi standardized Java interfaces

Currently, for the management of a bundle, the OSGi specifications define different Java objects with which a management application can interact. Using this approach, a management agent can implement extensive management of the OSGi framework, as well as any service standardized. Mapping the Java interfaces to the specific remote management protocol and data model tree is up to the management agent.

For runtime interaction with a bundle, a bundle can register a service in the service registry. However, this service interface is not standardized. Also, mapping the service interface to a general management model is not standardized. A current approach is to implement a proprietary service interface on all bundles to be managed. By tailoring this interface so that it easily maps to the management protocol primitives, it is simple for the management agent to map remote management commands to the bundle’s service interface. The disadvantage is the proprietary service interface, so that 3rd party bundles might not be compliant.

As a conclusion we can say that this current approach allows for extensive remote management of any aspect of the OSGi platform, but lacks a standardized service interface definition for bundles to implement.

### 4.2 Mobile specification approach

The Mobile Expert Group has provided its own solution based on the OMA [3] Device Management [4] specification to provide a remote management solution. The OSGi Mobile specification contains two chapters related to remote management:

- chapter 3: detailing the mobile management tree
- chapter 117: detailing: the DMT Admin service, bundle plugin interface specification, the notification service

The Device Management Tree model of OMA-DM was chosen as meta-data model and operational model. However, it was intended to be mappable to other protocols.

An analysis of mapping the Mobile specification DMT model to TR-069, however, shows that the current DMT model approach (as defined in the OSGi R4 Mobile Specification) introduces some issues. For example:

- Limitations for active or passive notifications on any parameter in the object tree
- The complexity of mapping a new protocol to the OMA-inspired DMT model, which could imply performance issues on limited devices.

#### 4.2.1 Support for TR-069 notifications

TR-069 offers the feature of active and passive notifications. By setting a parameter’s notification attribute, a remote manager requests to be notified with the parameter’s new value at the time the value changes (active notification) or at the next periodic inform (passive notification). Notification can be configured on any parameter of the TR-069 object tree. This approach enables the remote manager to be informed not only of changes in status variables of the platform, but also of configuration changes performed by a local manager, e.g. through a local Web interface.
The Mobile specification offers a few features that could help to implement TR-069 notification support:

- The DMT Admin service sends events using the Event Admin service when operations have been performed on nodes (nodes added, removed or copied; node values changed etc.)
- The OSGi Notification service defines a way to alert a remote management server. Protocol adapters on their turn have to implement a RemoteAlertSender interface (and register it) for use by the notification service. Notifications are sent by calling sendNotification on the notification service:
- The Monitor Admin service: A bundle can register a Monitorable service, to be used by the Monitor Admin service. By registering a Monitorable service, the bundle exposes access to a number of status variables. Notification can be implemented by the StatusVariable provider. If it does, it will call the update method on the Monitor Listener. The Monitor Admin service then generates an event on the Event Admin service. The Monitor service is currently also represented in the DMT tree.

Two problems arise when trying to map the current approach to TR-069:

- TR-069 defines that notification is applicable to any parameter in the object tree.

  Currently, the DMT Admin service only send events for operations on DMT nodes that were performed using the DMT Admin API. The DMT Admin Spec does not require Data Plugins to send events when DMT nodes are changed through except the DMT Admin API. It is called internal changes hereafter. For example: if configuration changes are performed by using the Configuration Admin service API, there might be no events sent. Such internal changes should be supported for TR-069 notification. However, it cannot be supported.

  The OSGi Monitor service only supports notification of changes on Status Variables, exposed through a Monitorable service, and enabled by the bundle to support on-change notification (i.e. dynamic Status Variables).

- Requesting notification is not fully under the control of the remote manager. In the case of a bundle using the notification service, there is no standardized way to configure the bundle to send alerts when the value of one of the implemented DMT nodes changes. In the case of the monitor service, the sending of events can be controlled, but is limited to dynamic Status Variables.

  The current DMT Admin service has no attributes properties on its nodes to be used to configure notification behavior, such as active notification and passive notification defined in TR-069. Therefore, a remote manager cannot control the notification behavior of DMT nodes in a standardized way.

To conclude, the current options, as provided in the Mobile specification, limit notification of parameter changes to StatusVariables, explicitly enabled for monitoring. There is no standardized approach available to monitor changes on any node in the DMT.

### 4.2.2 Mapping TR-069 to the OMA-DM inspired DMT model

Within the OSGi Mobile specification, the choice has been made to model the DMT after OMA-DM.

As a result, creating an OMA-DM protocol adapter is quite straightforward. Although no major hurdles have been identified in creating a TR-069 protocol adapter, it is less straightforward:
● The TR-069 RPC primitives have to be translated to the DMT Admin service interface methods (which are OMA-DM RPC inspired).

● The TR-069 data types have to be mapped to the DMT Admin data types. However, TR-069 data types, such as “unsignedint” and “dateTime” (ISO 8601), cannot be translated appropriately into DMT Admin data types defined in the current specification. Translating these data types might result a limitation of the available value range and a complex object that consists of multiple nodes, respectively.

### 4.3 Conclusion

The OSGi Mobile specification delivers a standardized data model (the DMT), and standardized interface (on the DMT Admin service) to enable remote management through a protocol adapter. However, in the current specification there is some support lacking for TR-069 notifications. Furthermore, since the DMT model is OMA-DM inspired, implementing a TR-069 protocol adapter is not straightforward, although not impossible.

### 5 Requirements

**REQUIREMENT[8]:** The solution should specify a guideline of RPC mapping between DMT Admin service interfaces and remote management protocols, such as TR-069.

### 6 Technical Solution

This RFC provides a guideline for RPC mapping between DMT Admin service interfaces and TR-069 protocol, which is defined by Broadband Forum [6]. Since the DMT Admin service focuses on the OMA-DM protocol for remote management, some features, such as RPC mapping and notification mechanism, are inadequate in terms of supporting the TR-069 protocol.

This RFC makes the following recommendations to support TR-069.

● Guideline mapping between the DMT Admin interfaces and TR-069 protocol RPCs
- Management of sessions including read-only and transactional session
- Data type mapping not defined in the DMT Admin specification version 1.0
- Error code mapping and SOAP Fault handling for TR-069 protocol

The basic architecture of DMT Admin service does not need to be modified to support the TR-069 protocol. Therefore, only the specifications and APIs that are related to the above recommendations are included in this document.

### 6.1 Basic architecture of TR-069 protocol implementation

The basic architecture of a system using TR-069 follows that defined in DMT Admin specification (see Figure 1). In the architecture, there must be a protocol adapter that uses the TR-069 protocol to communicate with remote manager. On the other hand, each data model, which is intended to be managed through the protocol adapter, must be implemented as a data plugin for DMT Admin service.

![Fig.1 Basic Architecture of TR-069 Protocol Implementation](image)

The TR-069 protocol adapter should call `DmtSession` interface provided by DMT Admin service when the remote manager calls some particular protocol adapter's RPCs specified by TR-069. Data plugins are used to implement the data models that are to be manipulated via the interfaces defined in the DMT Admin specification. Therefore, a remote manager's operation through the TR-069 protocol can be propagated to the data plugins.

On the other hand, notifications from data plugins to remote managers are propagated through DMT Admin and TR-069 protocol adapter. There are two possible ways to send notifications to the TR-069 protocol adapter; the first one is using the Notification Service registered by the DMT Admin, and the second one is using the Event Admin. In the first case, the data plugin gets the services and calls the method to send a notification. The Remote Alert Sender service registered by the TR-069 protocol adapter will receive it. In the second case, the data plugin gets the Event Admin service and calls the method to fire a event and the Event Handler service registered by the...
TR-069 protocol adapter will receive it. Then, the TR-069 protocol adapter should send the TR-069 notification to the remote manager properly.

### 6.2 RPC mapping for TR-069

DMT Admin interface is not restricted to any specific protocol but is inspired by the RPCs of OMA-DM, because OMA-DM is a primary remote management protocol for the mobile industry.

In TR-069 specification, there are 17 RPCs for handling client devices, which include basic setter / getter methods and methods modifying tree-structured data. Although these RPCs are very similar to OMA-DM RPCs, there is a slight mismatch between TR-069 RPCs and the methods defined by the DMT Admin specification.

Therefore, mapping rules between them should be defined in order to manipulate data models properly such as the Residential Management Model defined in RFC-140.

This RFC provides the guidelines;

- how an implementation of TR-069 protocol adapter should call one or more DMT Admin methods, when it receives TR-069 RPCs from a remote manager, and
- how an implementation of TR-069 protocol adapter should call one or more TR-069 RPCs, when it receives DMT Admin methods callback, originated by Data Plugins.

#### 6.2.1 TR-069 CPE RPCs to DMT Admin Interfaces

Table 4.1 shows the mapping guideline of TR-069 CPE RPCs to DMT Admin interfaces. TR-069 CPE RPCs are the RPCs implemented in the client side program and called by a remote server. TR-069 protocol adapter, therefore, calls DMT Admin service through the DmtSession interface.

**Table 4.1 Mapping of TR-069 CPE RPCs to DMT Admin interfaces**

<table>
<thead>
<tr>
<th>Type</th>
<th>TR-069 RPCs</th>
<th>DMT Admin methods</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>TR-069 CPE required RPC</td>
<td>GetRPCMethods</td>
<td>None</td>
<td>There is no recommendation for these RPCs. These RPCs’ action depends on the implementation of the protocol adapter.</td>
</tr>
<tr>
<td>SetParameterValues</td>
<td>DmtSession.setNodeValue()</td>
<td></td>
<td>This RPC is to set the values of the specified nodes. Therefore, DmtSession.setNodeValue() should be called as the corresponding method. When the RPC contains multiple ParameterValueStructs, the method should be called for each name-value pair. CPE should return the SetParameterValuesResponse, which is the corresponding response of the SetParameterValues, after completion of all method calls for the ParameterValueStructs.</td>
</tr>
</tbody>
</table>
| GetParameterValues          | DmtSession.getNodeValue() |                         | This RPC gets the values from the specified nodes. Therefore, DmtSession.getNodeValues() should be called as the corresponding method. When the RPC contains multiple strings indicating path names, the method should be called for each path name. Moreover, if the indicated string is
<table>
<thead>
<tr>
<th>Type</th>
<th>TR-069 RPCs</th>
<th>DMT Admin methods</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GetParameterNames</td>
<td>DmtSession.getEffectiveNodeAcl()</td>
<td>This RPC is used to obtain parameters of the specified path. Therefore, DmtSession.getChildNodeNames() should be called to get the node name as a parameter name. Because the obtained information includes the permission for overwriting the specified node, DmtSession.getEffectiveNodeAcl() should be also used to get the information. These methods should be called recursively for all of the nodes in the sub-tree indicated by the parameter path. CPE should return the GetParameterNamesResponse after completion of all method calls for the arguments. If the protocol adapter encounters the DmtException with PERMISSION_DENIED code, it should continue operation for getting the next node name and should discard the DmtException.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DmtSession.getChildNodeNames()</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SetParameterAttributes</td>
<td>None</td>
<td>There is no recommendation for these RPCs. These RPCs’ action depends on the implementation of the protocol adapter.</td>
</tr>
<tr>
<td></td>
<td>GetParameterAttributes</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>AddObject</td>
<td>DmtSession.createInteriorNode()</td>
<td></td>
<td>This RPC is used to create an object sub-tree including interior nodes and leaf nodes, and only interior node indicating object name, which is usually the node locating at the top of sub-tree hierarchy, can be specified. At first, the protocol adapter should call DmtSession.getChildNodeNames() to get instance IDs currently existing to check the instance IDs which is created by the data plugin internally. Second, the protocol adapter chooses one instance ID that has been never used before,</td>
</tr>
<tr>
<td>Type</td>
<td>TR-069 RPCs</td>
<td>DMT Admin methods</td>
<td>Remarks</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>which means the instance ID must not be reused for the same object beyond reboots of an OSGi framework. The protocol adapter should then call the DmtSession.createInteriorNode() with the specified path name and the selected instance ID, and the data plugin tries to create the desired object under the indicated instance ID. If the indicated ID is not acceptable for the data plugin, for example the ID is already used by the data plugin internally or the ID is reserved to create instance in the future, the data plugin must throw DmtException with COMMAND_FAILED code to the protocol adapter. If the DmtException is thrown, the protocol adapter should change the instance ID and should retry the creation of the object by calling DmtSession.createInteriorNode() until the prescribed number of times has been exceeded. Finally, the protocol adapter must return the AddObjectResponse with the instance ID to the remote manager if the operation succeeded. [Remarks] A data plugin may give the same instance ID to the same target in order to keep consistency of the node path, if the instance is created by not the operation from the protocol adapter but by the data plugin internally.</td>
</tr>
<tr>
<td></td>
<td>DeleteObject</td>
<td>DmtSession.deleteNode()</td>
<td>This RPC is to delete the specified object including interior and leaf nodes. Therefore, the DmtSession.DeleteNode() should be called with the instance ID of the object, which is specified as the parameter of DeleteObject RPC.</td>
</tr>
<tr>
<td></td>
<td>Reboot</td>
<td>None</td>
<td>There is no recommendation for these RPCs. These RPCs' action depends on the implementation of the protocol adapter.</td>
</tr>
<tr>
<td></td>
<td>Download</td>
<td>None</td>
<td>There is no recommendation for these RPCs. These RPCs' action depends on the implementation of the protocol adapter.</td>
</tr>
<tr>
<td>TR-069 CPE optional RPC</td>
<td>ScheduleInform</td>
<td>None</td>
<td>This RPC is not related to DMT Admin method. Therefore, there is no recommendation for this RPC.</td>
</tr>
<tr>
<td></td>
<td>FactoryReset</td>
<td>None</td>
<td>There is no recommendation for these RPCs. These RPCs' action depends on the implementation of the protocol adapter.</td>
</tr>
<tr>
<td></td>
<td>GetQueuedTransfers</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Upload</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SetVouchers</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GetOptions</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

### 6.2.2 DMT Admin Interfaces to TR-069 ACS RPCs

Table 4.2 shows the mapping guideline of DMT Admin interfaces to TR-069 ACS RPCs. TR-069 ACS RPCs are the RPCs implemented in the server side program and called by a managed client. Therefore, the TR-069
A protocol adapter must register the RemoteAlertSender service, which is defined in the DMT Admin specification, with the "principals" service property that represents the associated principals of the protocol adapter.

Table 4.2 Mapping of TR-069 ACS RPCs to DMT Admin interfaces

<table>
<thead>
<tr>
<th>Type</th>
<th>DMT Admin methods</th>
<th>TR-069 RPCs</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>TR-069 ACS required RPC</td>
<td>RemoteAlertSender.sendAlert()</td>
<td>Inform</td>
<td>A protocol adapter must register the RemoteAlertSender service with the &quot;principals&quot; property that represents the associated principals of the protocol adapter. When the RemoteAlertSender.sendAlert() implemented by the protocol adapter is called, the Inform RPC should be fired to the associated ACS. The &quot;code&quot; specified as the parameter of the method should be interpreted as the EventCode included in the EventStruct defined in the TR-069 specification. And the ParameterList that is contained in the Inform RPC should include the ParameterValueStruct consisting of the Name and Value, which should be the &quot;source&quot; and the &quot;data&quot; included in the AlertItem, respectively. If the AlertItem does not contain any &quot;source&quot; as the node that is the cause of the alert, the protocol adapter should discard the AlertItem and does not have to send Inform RPC. Note that this definition intends to implement scenarios in which an alert emerges from data plugins. Therefore, other use cases, such as Periodic, Boot and TransferComplete, should be implemented on an user-defined way.</td>
</tr>
<tr>
<td>None</td>
<td>TransferComplete</td>
<td>None</td>
<td>There is no recommendation for these RPCs. This RPC's action depends on the implementation of the protocol adapter.</td>
</tr>
<tr>
<td>TR-069 ACS optional RPC</td>
<td>None</td>
<td>GetRPCMethods</td>
<td>There is no recommendation for these RPCs. These RPCs' action depends on the implementation of the protocol adapter.</td>
</tr>
<tr>
<td>None</td>
<td>AutonomousTransfer</td>
<td>AutonomousTransfer</td>
<td>None</td>
</tr>
</tbody>
</table>

6.3 Session management

6.3.1 Opening Session

The DMT Admin specification defines three types of locking modes of sessions for protocol adapters to open a DmtSession for reading or writing node values; LOCK_TYPE_SHARED, LOCK_TYPE_EXCLUSIVE and LOCK_TYPE_ATOMIC, which correspond to the ReadableDataSession, the ReadWriteDataSession and the TransactionalDataSession respectively.
The protocol adapter adaptively chooses either LOCK_TYPE_ATOMIC or LOCK_TYPE_EXCLUSIVE as the lock mode based on the RPC, when it receives the SOAP request from the remote manager. If the received RPC needs to write node values, the protocol adapter must use LOCK_TYPE_ATOMIC session. If there already exists an opened LOCK_TYPE_ATOMIC session, the session will continue to be used. Otherwise, the protocol adapter should open a new session as LOCK_TYPE_ATOMIC. If the received RPC does not need to write node values, the protocol adapter should use LOCK_TYPE_SHARED session. If there already exists an opened LOCK_TYPE_SHARED session, the session will continue to be used. Otherwise, the protocol adapter should open a new session as LOCK_TYPE_SHARED.

On the other hand, the timing of closing session is not defined in this RFC. The close of the session should be conducted in a reasonable way depends on the implementation. In general, the protocol adapter should avoid opening too many sessions concurrently.

Note: in case that the SOAP request requires to read many node values simultaneously in one RPC operation such as GetParameterValues and GetParameterNames, there is risk that the consistency of data may not be kept because of value changes during the operation. Therefore, the protocol adapter is recommended to open the session as LOCK_TYPE_EXCLUSIVE for those RPCs, although these RPCs do not need to write any node value.

Note: if the received RPC needs to write only one node value, the protocol adapter may use LOCK_TYPE_EXCLUSIVE because there is no reason to support transactions.

If a data plugin does not support the TransactionalDataSession but does support the ReadWriteDataSession, the protocol adapter may use the LOCK_TYPE_EXCLUSIVE instead of the LOCK_TYPE_ATOMIC. In this case, the atomicity of the data is not ensured by the DMT Admin service. If a data plugin does not support either the TransactionalDataSession or ReadWriteDataSession, the protocol adapter cannot change values for any nodes and should return the corresponding SOAP Fault response to the remote manager.

The protocol adapter may open or close the session every time it receives a SOAP Request or sends the corresponding SOAP Response, respectively.

### 6.3.2 Commit for Atomic Session

The DMT Admin supports the transaction management of the operations conducted through the DmtSession interface. However, there is no explicit RPC in the TR-069 specification to handle atomicity of those operations. Therefore, this RFC recommends the following implementation regarding the management of the transactional session.

Let us assume that the protocol adapter receives the SOAP request from the remote manager and the protocol adapter needs to use a LOCK_TYPE_ATOMIC session as described in Section 6.4.1. In that case, the session should be opened at the start of processing the request that needs to write node values, if the session has not been opened. If the corresponding RPC operation has been completed normally, the protocol adapter should call the DmtSession.commit() method before returning the SOAP response to the remote manager. After calling commit(), it can close the session.

Note: even if the corresponding SOAP response fails to send to the remote manager, the changed data will be stored in the data plugin because the protocol adapter cannot rollback the operation after calling commit().

If the protocol adapter encounters any error conditions during the operation, such as DmtExceptions, it should call the DmtSession.rollback() to restore data consistency and should return SOAP Fault response to the remote manager.
6.4 Mapping of data types

There are 9 data types defined in the TR-106 [7]. Table 4.3 shows the mapping of these data types to DmtData types defined in DMT Admin specification.

Five data types, unsignedInt, Long, unsignedLong, HexBinary and dataTime, are missing in the DMT Admin specification version 1.0 as regards accommodating TR-069 protocol. Regarding Long and dataTime, extended DMT Admin specification described in the RFC-141 introduces new data types: FORMAT_LONG and FORMAT_DATETIME. See also RFC-141.

Here, these data types should be handled in the following way.

Table 4.3 Mapping of TR-069 data types to DMT data types

<table>
<thead>
<tr>
<th>Type</th>
<th>TR-069 data type</th>
<th>DMT Admin data type</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Type</td>
<td>String</td>
<td>FORMAT_STRING</td>
<td>Created by DmtData(String)</td>
</tr>
<tr>
<td></td>
<td>int</td>
<td>FORMAT_INTEGER</td>
<td>Created by DmtData(int)</td>
</tr>
<tr>
<td></td>
<td>long</td>
<td>FORMAT_LONG</td>
<td>Created by DmtData(long), newly defined in RFC-141.</td>
</tr>
<tr>
<td></td>
<td>unsignedInt</td>
<td>FORMAT_RAW_STRING</td>
<td>The unsignedInt is an unsigned integer in the range 0 to 4294967295, inclusive. Since there is no suitable data type defined in Java, this data type should be mapped as FORMAT_RAW_STRING. The format name must be &quot;unsignedInt&quot;. This data type is</td>
</tr>
</tbody>
</table>

Copyright © NTT Corporation 2010 All Rights Reserved
<table>
<thead>
<tr>
<th>Data Type</th>
<th>Format Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsignedLong</td>
<td>FORMAT_RAW_STRING</td>
<td>The unsignedLong is an unsigned long in the range 0 to 18446744073709551615, inclusive. Since there is no suitable data type defined in Java, this data type should be mapped as FORMAT_RAW_STRING. The format name must be &quot;unsignedLong&quot;. This data type is created by DmtData(String, String).</td>
</tr>
<tr>
<td>boolean</td>
<td>FORMAT_BOOLEAN</td>
<td>Created by DmtData(boolean)</td>
</tr>
<tr>
<td>dateTime</td>
<td>FORMAT_DATETIME</td>
<td>The dateTime is a String object that is interpreted as an dateTime type defined in ISO 8601; it is used as a value of date and time in TR-069. This data type is created by DmtData(String value, int format), where value is format is FORMAT_DATETIME, newly defined in RFC-141.</td>
</tr>
<tr>
<td>base64</td>
<td>FORMAT_BASE64</td>
<td>Created by DmtData(byte[], boolean)</td>
</tr>
<tr>
<td>hexBinary</td>
<td>FORMAT_RAW_BINARY</td>
<td>Hex encoded binary defined in TR-106. Since there is no suitable data type defined in Java, this data type should be mapped as FORMAT_RAW_Binary. The format name must be &quot;hexBinary&quot;. This data type is created by DmtData(String, byte[]).</td>
</tr>
</tbody>
</table>

### 6.5 Error code and SOAP Fault

#### 6.5.1 Error code and Fault code mapping

The DMT Admin specification defines The DmtIllegalStateException and the DmtException containing 17 codes to represent the cause of an error. On the other hand, TR-069 defines 20 fault codes as standard errors in its specification. Therefore, this RFC defines the mapping between these error codes and fault codes.
### Table 4.4 Mapping of Exception code to TR-069 Fault Code

<table>
<thead>
<tr>
<th>Class</th>
<th>DmtException code</th>
<th>TR-069 Fault code</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALERT_NOT_ROUTED</td>
<td>None</td>
<td>9002</td>
<td>This error never happens against a protocol adapter.</td>
</tr>
<tr>
<td>COMMAND_FAILED</td>
<td></td>
<td>9003</td>
<td>For the CWMP fault element.</td>
</tr>
<tr>
<td>COMMAND_NOT_ALLOWED</td>
<td></td>
<td>9008</td>
<td>For the SetParameterValuesFault element.</td>
</tr>
<tr>
<td>CONCURRENT_ACCESS</td>
<td></td>
<td>9003</td>
<td>For the CWMP fault element.</td>
</tr>
<tr>
<td>DATA_STORE_FAILURE</td>
<td></td>
<td>9008</td>
<td>For the SetParameterValuesFault element.</td>
</tr>
<tr>
<td>FEATURE_NOT_SUPPORTED</td>
<td></td>
<td>9002</td>
<td>Only for the CWMP fault due to the SetParameterValues RPC.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9008</td>
<td>For the SetParameterValuesFault element.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9009</td>
<td>Only for the CWMP fault due to the SetParameterAttributes RPC.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9002</td>
<td>For the CWMP fault due to the other RPCs.</td>
</tr>
<tr>
<td>INVALID_URI</td>
<td></td>
<td>9003</td>
<td>Only for the CWMP fault due to the SetParameterValues RPC.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9005</td>
<td>For the SetParameterValuesFault element.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9005</td>
<td>For the CWMP fault due to the other RPCs.</td>
</tr>
<tr>
<td>METADATA_MISMATCH</td>
<td></td>
<td>9003</td>
<td>For the CWMP fault element.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9007</td>
<td>For the SetParameterValuesFault element.</td>
</tr>
<tr>
<td>NODE_ALREADY_EXISTS</td>
<td></td>
<td>9004</td>
<td></td>
</tr>
<tr>
<td>NODE_NOT_FOUND</td>
<td></td>
<td>9003</td>
<td>Only for the CWMP fault due to the SetParameterValues RPC.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9005</td>
<td>For the SetParameterValuesFault element.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9005</td>
<td>For the CWMP fault due to the other RPCs.</td>
</tr>
<tr>
<td>PERMISSION_DENIED</td>
<td></td>
<td>9003</td>
<td>Only for the CWMP fault due to the SetParameterValues RPC.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9008</td>
<td>For the SetParameterValuesFault element.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9001</td>
<td>For the CWMP fault due to the other RPCs.</td>
</tr>
</tbody>
</table>
### 1.1 SOAP Fault expression

In this section, the expression of the SOAP Fault response is defined based on the error code and fault code mapping.

If the SOAP fault does not contain SetParameterValuesFault, the protocol adapter must set a code and a message contained in a DmtException instance to a FaultCode element and a FaultString element in the SOAP Fault response, respectively. The code in the DmtException must change the Fault code according to Table 4.4. Other information such as a path and causes included in the DmtException instance may be added to the FaultString element.

If the SOAP fault is raised by the SetParameterValues and needs to include SetParameterValuesFault element, the protocol adapter must set path, code and message contained in a DmtException instance to ParameterName, FaultCode element and FaultString element in the SetParameterValuesFault element, respectively. The code in the DmtException must change the Fault code according to Table 4.4. If there are multiple paths in the DmtException, each path must be described as one SetParameterValuesFault element. Other information such as cause information included in the DmtException instance may be added to the FaultString element included in the SetParameterValuesFault element.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMOTE_ERROR</td>
<td>None</td>
<td>This error never happens against a protocol adapter.</td>
</tr>
<tr>
<td>ROLLBACK_FAILED</td>
<td>9002</td>
<td></td>
</tr>
<tr>
<td>SESSION_CREATION_TIMEOUT</td>
<td>9002</td>
<td></td>
</tr>
<tr>
<td>TRANSACTION_ERROR</td>
<td>9003</td>
<td>For the CWMP fault element.</td>
</tr>
<tr>
<td></td>
<td>9008</td>
<td>For the SetParameterValuesFault element.</td>
</tr>
<tr>
<td>UNAUTHORIZED</td>
<td>9001</td>
<td></td>
</tr>
<tr>
<td>URI_TOO_LONG</td>
<td>9003</td>
<td>Only for the CWMP fault due to the SetParameterValues RPC.</td>
</tr>
<tr>
<td></td>
<td>9005</td>
<td>For the SetParameterValuesFault element.</td>
</tr>
<tr>
<td></td>
<td>9005</td>
<td>For the CWMP fault due to the other RPCs.</td>
</tr>
<tr>
<td>DmtIllegalStateException</td>
<td>None</td>
<td>9002</td>
</tr>
<tr>
<td>SecurityException</td>
<td>None</td>
<td>9001</td>
</tr>
<tr>
<td>Other Exceptions</td>
<td>None</td>
<td>9002</td>
</tr>
</tbody>
</table>
7 Considered Alternatives

7.1 Future work for notification handling
As described in Section 4.2.1, in order to realize TR-069 based notification especially regarding internal changes in OSGi, what Data Plugins and TR-069 protocol adapter, respectively, needs to do should be clarified. However, those are out of scope of this RFC, because the much further discussion will be required for the clarification.

7.2 Future work for forced inform parameters
Some data model specified by BBF, such as TR-098, defines some nodes as forced inform parameters. The values of the nodes must be included in the ParameterList argument of every Inform message to the ACS. How to realize it in OSGi should be clarified. However, those are out of scope of this RFC, because the much further discussion will be required for the clarification.

8 Security Considerations

All security requirements follow the DMT Admin specification.

9 Document Support

9.1 References
[3]. OMA, Open Mobile Alliance. The mission of the Open Mobile Alliance is to facilitate global user adoption of mobile data services by specifying market driven mobile service enablers that ensure service interoperability across devices, geographies, service providers, operators, and networks, while

Copyright © NTT Corporation 2010 All Rights Reserved
allowing businesses to compete through innovation and differentiation.

http://www.openmobilealliance.org/

[4]. OMA Device Management specification v1.2. The goal of the Device Management Working Group is to specify protocols and mechanisms that achieve management of mobile devices including the necessary configuration to access services and management of the software on mobile devices. http://www.openmobilealliance.org/release_program/dm_v1_2C.html


[6]. The Broadband Forum is a global consortium of nearly 200 leading industry players covering telecommunications, equipment, computing, networking and service provider companies. Established in 1994, originally as the ADSL Forum and later the DSL Forum, the Broadband Forum continues its drive for a global mass market for broadband, to deliver the benefits of this technology to end users around the world over existing copper telephone wire infrastructures. http://www.broadband-forum.org/about/forumhistory.php

[7]. Data model template for TR-069 enabled devices, TR-106 amendment 1, November 2006

9.2 Author's Address

<table>
<thead>
<tr>
<th>Name</th>
<th>Koya Mori</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>NTT Corporation</td>
</tr>
<tr>
<td>Address</td>
<td>Y320C, 1-1 Hikari-no-oka, Yokosuka, Kanagawa, Japan</td>
</tr>
<tr>
<td>Voice</td>
<td>+81-46-859-3446</td>
</tr>
<tr>
<td>e-mail</td>
<td><a href="mailto:mori.kouya@lab.ntt.co.jp">mori.kouya@lab.ntt.co.jp</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Ikuo Yamasaki</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company</td>
<td>NTT Corporation</td>
</tr>
<tr>
<td>Address</td>
<td>Y320C, 1-1 Hikari-no-oka, Yokosuka, Kanagawa, Japan</td>
</tr>
<tr>
<td>Voice</td>
<td>+81-46-859-8537</td>
</tr>
<tr>
<td>e-mail</td>
<td><a href="mailto:yamasaki.ikuo@lab.ntt.co.jp">yamasaki.ikuo@lab.ntt.co.jp</a></td>
</tr>
</tbody>
</table>

9.3 Acronyms and Abbreviations
9.4 End of Document
The OSGi Alliance and its members specify, create, advance, and promote wide industry adoption of an open delivery and management platform for application services in home, commercial buildings, automotive and industrial environments. The OSGi Alliance serves as the focal point for a collaborative ecosystem of service providers, developers, manufacturers, and consumers. The OSGi specifications define a standardized, component oriented, computing environment for networked services. OSGi technology is currently being delivered in products and services shipping from several Fortune 100 companies. The OSGi Alliance's horizontal software integration platform is ideal for both vertical and cross-industry business models within home, vehicle, mobile and industrial environments. As an independent non-profit corporation, the OSGi Alliance also provides for the fair and uniform creation and distribution of relevant intellectual property – including specifications, reference implementations, and test suites – to all its members.

HOW TO REACH US:

OSGi Alliance
Bishop Ranch 6
2400 Camino Ramon, Suite 375
San Ramon, CA 94583 USA

Phone: +1.925.275.6625
E-mail: marketinginfo@osgi.org
Web: http://www.osgi.org

OSGi is a trademark of the OSGi Alliance in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

All other marks are trademarks of their respective companies.