RFC 111

Extending the MEG DMT Admin API to non-OSGi Platforms

Draft

11 Pages

Abstract

The current OSGi MEG DMT Admin API is in a single package that is only applicable to classic OSGi environments which include an OSGi Framework. This RFC proposes a way to extend the Mobile Expert Group Device Management Tree (MEG DMT) Admin API to non-OSGi platforms like CLDC. It involves minimal changes to the MEG DMT Admin API.

0 Document Information

0.1 Table of Contents

0 Document Information 2
0.1 Table of Contents 2
0.2 Terminology and Document Conventions 2
0.3 Revision History 3

1 Introduction 3
1.1 Terminology 3
1.2 History/Prior Work 3

2 Application Domain 4

3 Problem Description 4

4 Requirements 5

5 Technical Solution 6
5.1 Overview 6
5.2 Comparision to issues list 8
5.3 Javadoc 9

6 Considered Alternatives 9

7 Security Considerations 9

8 Document Support 10
8.1 References 10
8.2 Author's Address 10
8.3 Acronyms and Abbreviations 10
8.4 End of Document 11

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 [1].

All Page Within This Box

`Source code is shown in this typeface.`

0.3 Revision History

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

Revision	Date	Comments
Initial	*19 August 2005*	*Initial version.* *Joe Rusnak, IBM, jgrusnak@us.ibm.com*
1.1	*4 November 2005*	*Updating the document to address all the issues identified during the 246/232 alignment calls.* *Gábor Pécsy, Nokia, gabor.pecsy@nokia.com*
1.2	*7 November 2005*	*Last minute suggestion from BJ Hargrave.* *Joe Rusnak, IBM, jgrusnak@us.ibm.com*

1 Introduction

The MEG DMT Admin API[1] provides a set of Java APIs to manage an OMA DM Device Management Tree (DMT). As currently defined, it only works in an OSGi Service Platform environment.

The industry has expressed interest in a more general Java API to navigate and manipulate OMA DM DMTs, one that works on CDC (with or without the OSGi Service Platform) and CLDC platforms. This RFC proposes a minor extension to the current MEG DMT Admin API so that it can be used on the OSGi Service Platform, CDC (without the OSGi Service Platform), and CLDC.

1.1 Terminology

For conciseness, this document will refer to “non-OSGi platforms”. This includes J2ME CLDC (which cannot support a fully compliant OSGi Service Platform), J2ME CDC (without the OSGi Service Platform), and J2SE (without the OSGi Service Platform).

1.2 History/Prior Work

This RFC is based on a large body of earlier work in the OSGi Mobile Expert Group (MEG). Specifically:

[1] Chapter 118 of the OSGi Service Platform Mobile Specification R4 (draft) [3]

All Page Within This Box

- The MEG Device Management work stream created the DMT Admin Service which is documented in Chapter 118 of the OSGi Service Platform Mobile Specification Release 4 (draft) [3].

- Primarily in response to Issue 393, a re-factoring of the DMT Admin API was proposed. Part of this re-factoring involved splitting the monolithic DMT Admin API into an API that general Java developers can use to access the DMT, and an SPI that specialized DMT plug-in developers can use. As a stepping stone towards a universal "core" DMT Admin API that could be used on CLDC, it also factored out the parts of DMT Admin that require Java classes that are provided on CDC, but not CLDC.

- This RFC extends the re-factoring proposal to provide functions that are required for DMT administration which are available on the OSGi Service Platform, but not on non-OSGi platforms.

- This RFC addresses the issues identified during the alignment work set up between JSR 232 and JSR 246 expert groups.

2 Application Domain

This RFC provides a Java API to navigate, manipulate, and administer an OMA DMT. The API is applicable to all J2ME devices (CDC with the OSGi Service Platform, CDC without the OSGi Service Platform, and CLDC). The application domain is all J2ME devices that conform to OMA DM device management standards.

3 Problem Description

The current OSGi MEG DMT Admin API is specifically targeted to the OSGi Service Platform environment. In particular, there are two problems in making work on non-OSGi platforms:

1. Portions of the current DMT Admin API rely on Java capabilities (notably Java 2 permissions) that are not supported on CLDC.

2. The current OSGi MEG DMT Admin API does not specify some of the functions required for full DMT administration. This is because it gets these capabilities from other parts of the OSGi Service Platform. These functions need to be specified for non-OSGi platforms (CLDC, or CDC without OSGi MEG). The "missing" functions are:

All Page Within This Box

- The ability to register and unregister DMT plugins (the current MEG DMT Admin API uses the OSGi service registry for this).
- A mechanism to generate and subscribe to DMT events (the current MEG DMT Admin API uses the OSGi Event Admin service for this).

3. A number of structural differences between the two existing proposals (OSGi DMT Admin API and JSR 246 API proposal):
    a. Goal is to have the same single entry point class (Engine/DmtAdmin). It can be used to create session/context objects which provide access for all following management operations.
    b. The tree manipulation concepts are different regarding the usage of temporary objects for node access.
    c. getSession()/getContext(): JSR 246 has no getContext() method with explicit principal.
    d. Event listener registration: JSR 232 does not define such a mechanism because it uses the OSGi Event Admin Service (this is also mentioned in 2. above).
    e. setTimeout(): JSR 232 does not have one.
    f. Possible deadlock situations when locking the DMT -> both JSRs check again: JSR 246 may limit locking for lower or no DM permissions. JSR 232 has a fine grained permission mechanism which might be sufficient.
    g. ACL enforcement: JSR 246 has strict ACL enforcement, whereas JSR 232 has no ACL enforcement if no principal is given.
    h. Meta Data access: JSR 246 made a proposal for accessing meta information by string parameter which seems more future proof

The re-factoring of the DMT Admin API in response to Issue 393 addressed the first problem. This RFC addresses the second and third problems.

4 Requirements

1. There must be a core Java API for manipulating DMTs that is the same across all J2ME platforms (CDC with the OSGi Service Platform, CDC without the OSGi Service Platform, and CLDC).
2. The core API must have extensions so that additional functions (e.g., Java 2 security) that are available on CDC can be exploited.
3. The core API must have extensions so that DMT-related functions that are present in the OSGi Service Platform, but not on non-OSGi platforms, are made available on these platforms.

All Page Within This Box

5 Technical Solution

5.1 Overview

The existing MEG DMT Admin API of three packages:

org.osgi.service.dmt

- Acl
- AlertItem
- Data
- *DmtAdmin*
- *DmtException*
- *MetaNode*
- *Session*
- *RemoteAlertSender*

org.osgi.service.dmt.spi

- *DataPluginFactory*
- *ExecPlugin*
- *ReadableDataSession*
- *ReadWriteDataSession*
- *TransactionalDataSession*

org.osgi.service.dmt.security

- DmtAlertPermission
- DmtPermission
- DmtPrincipalPermission

The package org.osgi.service.dmt.spi provides a common SPI for DMT plug-in developers to use. The package org.osgi.service.dmt provides a core API for navigating and manipulating a DMT. These packages work on the

All Page Within This Box

OSGi Service Platform, on CLDC, and on CDC without the OSGi Service Platform. On the latter two platforms, however, these packages are necessary but not sufficient. To remedy this we propose:

1. Extending the package **org.osgi.service.dmt** as follows:

   - DmtEvent
     ```
     DmtEvent(String type, String[ ] nodes, String[ ] newNodes)
     //on OSGi platforms this object maps directly to
     //org.osgi.service.event.Event as used for DMT events
     ```

   - *DmtEventHandler*
     ```
     void handleDmtEvent(DmtEvent e)
     //on OSGi platforms this object maps directly to
     //org.osgi.service.event.EventHandler as used for DMT events
     ```

2. Extending org.osgi.service.dmt.DmtAdmin interface with the following methods:

   ```
   public DmtSession getSession( String subtreeUri, long timeout )
                  throws DmtException

   public DmtSession getSession( String subtreeUri, int lockMode, long timeout )
                  throws DmtException

   public DmtSession getSession( String principal, String subtreeUri,
                  int lockMode, long timeout ) throws DmtException
   ```

   These method would work exactly like the corresponding method with no timeout parameter, but they would throw an exception if the session cannot be opened within the specified timeframe. The timeout value 0 would mean infinite wait and this value would be used in those variants of the getSession method which have no timeout parameter.

3. Adding the following package and class:

org.osgi.service.dmt.registry

DmtRegistry

```
//method to get the DmtRegistry object.
static DmtRegistry getDmtRegistry()

//method to get the object implementing the DmtAdmin interface on non-
//OSGi platforms – maps directly to the service registry on OSGi
//platforms
public DmtAdmin getDmtAdmin()

//On OSGi platforms the following methods are “convenience methods” for
//the general OSGi Event Admin Service and the DMT Events defined as
//part of DMT Admin
public void addEventHandler(String type, DmtEventHandler e)
public void removeEventHandler(DmtEventHandler e)
```

These packages provide a consistent Java API for navigating, manipulating, and administering a DMT, as illustrated in the following table:

All Page Within This Box

OSGi Service Platform	J2ME CDC w/o OSGi Service Platform	J2ME CLDC[2]
org.osgi.service.dmt org.osgi.service.dmt.spi org.osgi.service.dmt.security (org.osgi.service.dmt.registry)[3]	org.osgi.service.dmt org.osgi.service.dmt.spi org.osgi.service.dmt.security org.osgi.service.dmt.registry	org.osgi.service.dmt org.osgi.service.dmt.spi org.osgi.service.dmt.registry

5.2 Comparision to issues list

The following list evaluates the proposal by checking how it addresses the identified open issues.

a. *Goal is to have the same single entry point class (Engine/DmtAdmin) to use as a factory for all following management operations.*

   We propose to separate the registry-like functions from the session management. This way DmtAdmin can remain an interface which gives much more flexibility in the implementation. In OSGi environments, the getDmtAdmin() methods can be mapped directly to the OSGi Service Registry by means of a "thin wrapper". For maximum efficiency in OSGi environments where portability is not an issue (e.g. in case of bundles), the OSGi Service Registry can be used directly.

b. *The tree manipulation concepts are different regarding the usage of temporary objects for node access..*

   We suggest not to use temporary objects for node access because this would result in intensive heap usage (lots of small objects created and destroyed) and can have a negative performance impact on the device.

c. *getSession()/getContext(): 246 has no getContext() method with explicit principal.*

   getSession() with explicit principal is needed to satisfy the use case of creating protocol adapters which can act on behalf of a remote management server. We suggest to keep this method.

d. *Event listener registration: 232 does not define such a mechanism because it uses the OSGi EventAdmin Sservice.*

   The proposed DmtRegistry class includes the listener registration capability. In an OSGi environment, the registration methods map directly to the OSGi Service registry. For maximum efficiency in OSGi environments where portability is not an issue (e.g. in case of bundles), the OSGi Service Registry can be used directly.

e. *setTimeout(): JSR 232 does not have one.*

   Instead of having a global setTimeout() method we suggest adding new variants of the getSession() methods with an additional timeout parameter. This solution reduces the interference between different clients of the API in a shared environment like OSGi.

f. *Possible deadlock situations when locking the DMT.*

   JSR 246 may limit locking for lower or no DM permissions. JSR 232 has a fine grained permission mechanism which is sufficient. However, this does not affect the API.

[2] The corresponding MIDP 2 permissions need to be defined for these API. However, these are JSR 246 specific.
[3] Use of this package is optional on OSGi. Using it promotes portability to non-OSGi platforms.

All Page Within This Box

g. *ACL enforcement: JSR 246 has strict ACL enforcement, whereas JSR 232 has no ACL enforcement if no principal is given*

The alignment groups agreed that ACL handling can be left JSR specific. However, this difference doesn't affect the API, neither the code using that API. The differences are visible for the policy administrator only.

h. *Meta Data access: JSR 246 made a proposal for accessing meta information by string parameter which seems more future proof*

We propose to keep the current MetaData interface, as it makes the use of standard OMA metadata easier and it provides useful functions for creating local management applications.

In addition, the use of well-known keys for accessing metadata attribute assumes that the keys are standardized and are known by the application. Application also needs to know the associated semantics. Therefore, the "future compatibility" gain seems to be quite minor and the API usage becomes more cumbersome and error prone.

5.3 Javadoc

(to be added)

6 Considered Alternatives

None

7 Security Considerations

The security aspects of DMT administration is discussed in Section 118.11 of [3].

All Page Within This Box

8 Document Support

8.1 References

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

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

[3]. OSGi Service Platform Mobile Specification Release 4 (draft), http://membercvs.osgi.org/sp-r4/drafts/meg.pdf

8.2 Author's Address

Name	Joe Rusnak
Company	IBM
Address	
Voice	919-543-3317
e-mail	jgrusnak@us.ibm.com

Name	Gábor Pécsy
Company	Nokia
Address	
Voice	+36-20-984-9052
e-mail	gabor.pecsy@nokia.com

8.3 Acronyms and Abbreviations

CDC: J2ME Connected Device Configuration (http://java.sun.com/j2me)

CLDC: J2ME Connected Limited Device Configuration (http://java.sun.com/j2me)

DMT: Device Management Tree

J2ME: Java 2 Micro Edition (http://java.sun.com/j2me)

All Page Within This Box

MEG: OSGi Mobile Expert Group

OMA: Open Mobile Alliance (http://www.openmobilealliance.org)

8.4 End of Document