Management Agent deployment

Confidential, Draft
RFC 27

17 Pages

Abstract

This document describes how a management agent may be deployed. A management agent is a function that resides within a Service Platform and that performs local functions on behalf of a management system. The management system is most often controlled by the Service Platform Operator. In order for the operator to control the Service Platform, the Service Platform must be equipped with a management agent and it must somehow be bound to the operator's management system. This document describes how a management agent may be deployed to the Service Platform and in particular how this is done when there is no prior relationship between the operator and the Service Platform. One of the most vital aspects of these procedures is that they are possible to perform in a secure way,

0 Document Information

0.1 Table of Contents

0.2 Status

This document specifies Management Agent deployment for the Open Services Gateway Initiative, and requests discussion and suggestions for improvements. Distribution of this document is unlimited within OSGi.

0.3 Acknowledgement

This document represents a joint effort of the Security and Remote Management Expert Groups. Among the contributors to this work you find the following persons; Frank Piessens (Acunia), Tomas Bornefall (Ericsson), Anders Joelson (Gatespace), Frank Seliger (IBM), Petri Niska (Nokia), Pavlin Dobrev (ProSyst), Jordan Simeonov (ProSyst) , James Jennings (Tivoli), Jonas Ekström (Gatespace) and Anna Tykesson (Gatespace).

0.4 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].

`Source code is shown in this typeface.`

0.5 Revision History

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

Revision	Date	Comments
1.00A	June 8, 2001	Derived from OSGi RFC 18 *Security Architecture*, §5.3 *Rendezvous and Bootstrap,* working draft of April 13, 2001. Author: William R Soley, Sun Microsystems <william.soley@sun.com>.
1.00B	June 19, 2001	1. Expanded explanation of details 2. Feedback from Petri Niska (Nokia) and Tomas Bornefall (Ericsson). 3. Changes for readability. Author: William R Soley, Sun Microsystems <william.soley@sun.com>.
1.00C	June 21, 2001	1. Added examples of HTTP transactions. 2. More changes for readability. Author: William R Soley, Sun Microsystems <william.soley@sun.com>.
1.00E	August 13,2001	Per June 27, 2001 RMEG/SEG discussion in Copenhagen: 1. Add description of end-state of each phase 2. No inbound connections to gateway 3. Multiple DNS mappings for rendezvous 4. Added sim card support 5. Added manual input support 6. Added non-IP support recommendations 7. Use java.util.Properties.load() syntax rather than XML. Other changes: 1. Generalized “rendezvous” to “binding” 2. Added recommendation for non-connected gateways 3. Added recommendation for branded gateways Author: William R Soley, Sun Microsystems <william.soley@sun.com>.
1.00F	September 21, 2001	1. New document scope 2. New title 3 … a whole lot of new things Author: Lars-Erik Helander, Gatespace <helander@gatespace.com>.
1.00G	October 19, 2001	Updates from meetings in San Francisco 1. Configuration data and certificates possible to exchange during SETUP. 2. Environment for execution of the Management agent Authors: Lars-Erik Helander, Gatespace <helander@gatespace.com> and Ben Reed, IBM <breed@almaden.ibm.com>.
1.00H	October 29, 2001	1. Minor correction. 2. API specification included. 3. Removed Bind ref. Authors: Lars-Erik Helander, Gatespace <helander@gatespace.com> and Ben Reed, IBM <breed@almaden.ibm.com>.

1.00I	November 1, 2001	Certificates made into X.509 variants Certificate details moved from RFC 36. Authors: Lars-Erik Helander, Gatespace <helander@gatespace.com> and Ben Reed, IBM <breed@almaden.ibm.com>.
1.00 J	November 30, 2001	1. Dictionary data transport format moved from RFC 36. 2. Dictionary data transport format changed from MIME to ZIP. 3. Minor adjustments and clarifications. Authors: Lars-Erik Helander, Gatespace <helander@gatespace.com> and Ben Reed, IBM <breed@almaden.ibm.com>.
1.00 K	January 04, 2002	1. Setup protocol generalized to Provisioning Data Assignment protocol and Load protocol removed. 2.Processes Staging, Bind and Setup not so very tied to the implementation 3.Some restructuring done. Adoption of RFC 28 terminology. 4.X509 issues moved to RFC 36 5. Unbind introduced. Authors: Lars-Erik Helander, Gatespace <helander@gatespace.com> and Ben Reed, IBM <breed@almaden.ibm.com>.
1.00 L	August 28, 2002	1. Text adjusted to current state of spec for R3. Authors: Lars-Erik Helander, Gatespace <helander@gatespace.com> and Ben Reed, IBM <breed@almaden.ibm.com>.
1.00 L	September, 2002	1. Text adjusted to current state of spec for R3.

1 Introduction

To allow freedom regarding the choice of management protocol, the OSGi, specifies an architecture to remotely manage a Service Platform with a Management Agent. The Management Agent is implemented with a Management Bundle that can communicate with an unspecified management protocol.

This specification defines how the Management Agent can make its way to the Service Platform, and gives a structured view of the problems and their corresponding resolution methods.

The purpose of this specification is to enable the management of a Service Platform by an Operator, and (optionally) to hand over the management of the Service Platform later to another Operator. This approach is in accordance with the OSGi remote management reference architecture.

This bootstrapping process requires the installation of a Management Agent, with appropriate configuration data, in the Service Platform.

This specification consists of a prologue, in which the principles of the Initial Provisioning are outlined, and a number of mappings to different mechanisms.

2 Motivation and Rationale

Policy Free – The proposed solution must be business model agnostic; none of the affected parties (Operators, SPS Manufacturers, etc.) should be forced into any particular business model.

Interoperability – The Initial Provisioning must permit arbitrary interoperability between management systems and Service Platforms. Any compliant Remote Manager should be able to manage any compliant Service Platform, even in the absence of a prior business relationship. Adhering to this requirement allows a particular Operator to manage a variety of makes and models of Service Platform Servers using a single management system of the Operator's choice. This rule also gives the consumer the greatest choice when selecting a Operator.

Flexible – The management process should be as open as possible, to allow innovation and specialization while still achieving interoperability.

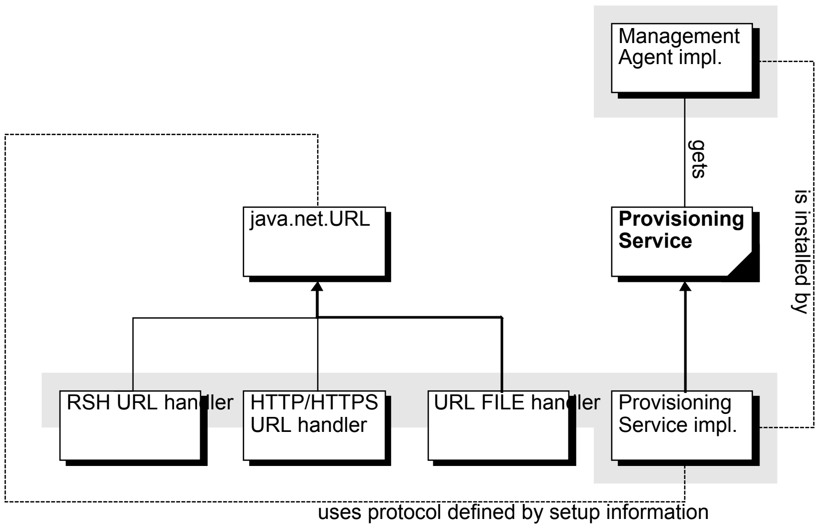
2.1 Entities

Provisioning Service – A service registered with the Framework that provides information about the initial provisioning to the Management Agent.

Provisioning Dictionary – A Dictionary object that is filled with information from the ZIP files that are loaded during initial setup.

RSH Protocol – An OSGi specific secure protocol based on HTTP (defined in RFC 36)

Management Agent – A bundle that is responsible for managing a Service Platform under control of a Remote Manager.



3 Technical Discussion

The following procedure should be executed by an OSGi Framework implementation that supports this Initial Provisioning specification.

When the Service Platform is first brought under management control, it must be provided with an initial request URL in order to be provisioned. Either the end user or the manufacturer may provide the initial request URL. How the initial request URL is transferred to the Framework is not specified, but a mechanism might, for example, be a command line parameter when the framework is started.

When asked to start the Initial Provisioning, the Service Platform will send a request to the management system. This request is encoded in a URL, for example:

http://osgi.acme.com/remote-manager

This URL may use any protocol that is available on the Service Platform Server. Many standard protocols exist, but it is also possible to use a proprietary protocol. For example, software could be present which can communicate with a smart card and could handle, for example, this URL:

smart-card://com1:0/7F20/6F38

Before the request URL is executed, the Service Platform information is appended to the URL. This information includes at least the Service Platform Identifier, but may also contain proprietary information, as long as the keys for this information do not conflict. Different URL schemes may use different methods of appending parameters; these details are specified in the mappings of this specification to concrete protocols

The result of the request must be a ZIP file (The content type must be application/zip). It is the responsibility of the underlying protocol to guarantee the integrity and authenticity of this ZIP file.

This ZIP file is unpacked and its entries (except bundle entries, described in the table below) are placed in a Dictionary object. This Dictionary object is called the Provisioning Dictionary. It must be made available from the Provisioning Service in the service registry. The names of the entries in the ZIP file must not start with a slash ("/").

Type	MIME	Type Description
text	MIME_STRING text/plain;charset=utf-8	Must be represented as a String object
binary	MIME_BYTE_ARRAY application/octet-stream	Must be represented as a byte array (byte[]).
bundle	MIME_BUNDLE application/x-osgi-bundle	Entries must be installed using BundleContext.installBundle(String,InputStream), with the InputStream object constructed from the contents of the ZIP entry. The location must be the name of the ZIP entry without leadingslash.
bundle-url	MIME_BUNDLE_URL text/x-osgi-bundle-ur l ; charset=utf-8	The content of this entry is a string coded in utf-8. Entries must be installed using BundleContext.installBundle(String,InputStream), with the InputStream object created from the given URL. The location must be the name of the ZIP entry without leading slash.

The ZIP file may contain only four types of dictionary entries: text, binary, bundle, or bundle-url, The types are specified in the ZIP entry's extra field, and must be a MIME type. The text and bundle-url entries are translated into a String object. All other entries must be stored as a byte[].

The Provisioning Service must install (but not start) all entries in the ZIP file that are typed in the extra field with bundle or bundle-url.

If an entry named PROVISIONING_START_BUNDLE is present in the Provisioning Dictionary, then its content type must be text as defined in the table above. The content of this entry must match the bundle location of a previously loaded bundle. This bundle must be given AllPermission and started. If no PROVISIONING_START_BUNDLE entry is present in the Provisioning Dictionary, the Provisioning Dictionary must contain a reference to another ZIP file under the PROVISIONING_REFERENCE key. If both keys are absent, no further action must take place.

If this key is present and holds a String object that can be mapped to a valid URL, then a new ZIP file must be retrieved from this URL. The PROVISIONING_REFERENCE link may be repeated multiple times in successively loaded ZIP files. Referring to a new ZIP file with such a URL allows a manufacturer to place a fixed reference inside the Service Platform Server (in a file or smart card) that will provide some platform identifying information and then also immediately load the information from the management system. The PROVISIONING_REFERENCE may be repeated multiple times in successively loaded ZIP files.The actual number of iterations must be maintained in the Provisioning Dictionary under the key PROVISIONING_UPDATE_COUNT.

Information retrieved while loading subsequent PROVISIONING_REFERENCE URLs may replace previous key/values in the Provisioning Dictionary, but must not erase unrecognized key/values. For example, if an assignment has assigned the key proprietary-x, with a value '3', then later assignments must not override this value, unless the later loaded ZIP file contains an entry with that name. All these updates to the provisioning information dictionary must stored persistently. At the same time each entry of type bundle or bundle-url must be installed ant not started.

Once the Management Agent has been started, the Initial Provisioning ser-vice has become operational. In this state, the Initial Provisioning service must react when the Provisioning Dictionary is updated with a new PROVISIONING_REFERENCE property. If this key is set, it should start the cycle again. For example, if the control of a Service Platform needs to be transferred to another Remote Manager, the Management Agent should set the PROVISIONING_REFERENCE to the location of this new Remote Manager's Initial Provisioning ZIP file. This process is called re-provisioning. If errors occur during this process, the Initial Provisioning service should try to notify the User of the problem.

The Management Agent may require configuration data that is specific to the Service Platform instance. If this data is transferred outside the Management Agent bundle, the merging of this data with the Management Agent may take place in the Service Platform. Transferring the data separately will make it possible to simplify the implementation on the server side, as it is not necessary to create personalized Service Platform bundles. The PROVISIONING_AGENT_CONFIG key is reserved for this purpose, but the Management Agent may use other keys if so desired.

The PROVISIONING_SPID key must contain the Service Platform Identifier.

3.1 Special Configurations

The next section shows some examples of specially configured types of Service Platform Servers and how they are treated with the respect to the specifications in this document.

3.1.1 Branded Service Platform Server

If a Service Platform Operator is selling Service Platform Servers branded exclusively for use with their service, the provisioning will most likely be performed prior to shipping the Service Platform Server to the User. Typically the Service Platform is configured with the Dictionary entry PROVISIONING_REFERENCE pointing at a location controlled by the Operator.

Up-to-date bundles and additional configuration data must be loaded from that location at activation time. The Service Platform is probably equipped with necessary security entities, like certificates, to enable secure downloads from the Operator's URL over open networks, if necessary.

3.1.2 Non-connected Service Platform

Circumstances might exist in which the Service Platform Server has no WAN connectivity, or prefers not to depend on it for the purposes covered by this specification.

The non-connected case can be implemented by specifying a file:// URL for the initial ZIP file (PROVISIONING_REFERENCE). That file:// URL would name a local file containing the response that would otherwise be received from a remote server.

The value for the Management Agent Reference (PROVISIONING_REFERENCE) found in that file will be used as input to the load process. The PROVISIONING_REFERENCE may point to a bundle file stored either locally or remotely. No code changes are necessary for the non-connected scenario. The file:// URLs must be specified, and the appropriate files must be created on the Service Platform.

3.2 The Provisioning Service

Provisioning information is conveyed between bundles using the Provisioning Service, as defined in the ProvsioningService interface. Any changes to the Provisioning Dictionary must be propagated directly to the Provisioning Service. The Provisioning Dictionary is retrieved from the ProvisioningService object using the getInformation() method.

3.3 Management Agent Environment

The Management Agent should be written with great care to minimize dependencies on other packages and services, as all services in OSGi are optional. Some Service Platforms may have other bundles pre-installed, so it is possible that there may be exported packages and services available. Mechanisms outside the current specification, however, must be used to discover these packages and services before the Management Agent is installed.

The Provisioning Service must ensure that the Management Agent is running with AllPermission. The Management Agent should check to see if the Permission Admin service is available, and establish the initial permissions as soon as possible to insure the security of the device when later bundles are installed. As the PermissionAdmin interfaces may not be present (it is an optional service), the Management Agent should export the PermissionAdmin interfaces to ensure they can be resolved.

Once started, the Management Agent may retrieve its configuration data from the Provisioning Service by getting the byte[] object that corresponds to the PROVISIONING_AGENT_CONFIG key in the Provisioning Dictionary. The structure of the configuration data is implementation specific.

The scope of this specification is to provide a mechanism to transmit the raw configuration data to the Management Agent. The Management Agent bundle may alternatively be packaged with its configuration data in the bundle, so it may not be necessary for the Management Agent bundle to use the Provisioning Service at all.

Most likely, the Management Agent bundle will install other bundles to provision the Service Platform. Installing other bundles might even involve downloading a more full featured Management Agent to replace the initial Management Agent.

3.4 Mapping To File Scheme

The file: scheme is the simplest and most completely supported scheme which can be used by the Initial Provisioning specification. It can be used to store the configuration data and Management Agent bundle on the Service Platform Server, and avoids any outside communication.

If the initial request URL has a file scheme, no parameters should be appended, because the file: scheme does not accept parameters.

3.4.1 Example With File Scheme

The manufacturer should prepare a ZIP file containing only one entry named PROVISIONING_START_BUNDLE that contains a location string of an entry of type application/x-osgi-bundle or application/x-osgi-bundle-URL.

For example, the following ZIP file demonstrates this:

provisioning.start.bundle	text	agent
agent	bundle	C0AF0E9B2AB

.The bundle may also be specified with a URL:

provisioning.start.bundle	text	http://acme.com/a.jar
agent	bundle-url	http://acme.com/a.jar

Upon startup, the framework is provided with the URL with the file: scheme that points to this ZIP file:

```
file:/opt/osgi/ma.zip
```

3.5 Other mappings

Mapping to other schemes are to be defined by other RFCs

4 Security Considerations

The security model for the Service Platform is based on the integrity of the Management Agent deployment. If any of the mechanisms used during the deployment of management agents are weak, or can be compromised, the whole security model becomes weak.

From a security perspective, one attractive means of information exchange would be a smart card. This approach enables all relevant information to be stored in a single place. The Operator could then provide the information to the Service Platform by inserting the smart card into the Service Platform.

4.1 Concerns

The major security concerns related to the deployment of the Management Agent are:

• The Service Platform is controlled by the intended Operator

• The Operator controls the intended Service Platform(s)

• The integrity and confidentiality of the information exchange that takes place during these processes must be considered

In order to address these concerns, an implementation of the OSGi Remote Management Architecture must assure that:

• The Operator authenticates itself to the Service Platform

• The Service Platform authenticates itself to the Operator

• The integrity and confidentiality of the Management Agent, certificates, and configuration data are fully protected if they are transported over public transports.

Each mapping of the Initial Provisioning specification to a concrete implementation must describe how these goals are met.

4.2 Service Platform Long-Term Security

Secrets for long-term use may be exchanged during the Initial Provisioning procedures. This way, one or more secrets may be shared securely, assuming that the Provision Dictionary assignments procedures used are implemented with the proper security characteristics. Permissions

The provisioning information may contain sensitive information. Also, the ability to modify provisioning information can have drastic consequences. Thus, only trusted bundles should be allowed to register, or get the Provisioning Service. This restriction can be enforced using ServicePermission[GETProvisioningService] .

No Permission classes guard reading or modification of the Provisioning Dictionary, so care must be taken not to leak the Dictionary object received from the Provisioning Service to bundles that are not trusted

5 API specification

5.1 org.osgi.service.provisioning Interface ProvisioningService

public interface **ProvisioningService**

Service for managing the initial provisioning information.
Initial provisioning of an OSGi device is a four step process that culminates with the installing and execution of the initial management agent. At each step of the process information is collected for the next step. Multiple bundles may be involved and this service provides a means for these bundles to exchange information. It also provides a means for the initial Management Bundle to get its initial configuration information.
The provisioning information is collected in a `Dictionary` object, called the Provisioning Dictionary. Any bundle that can access the service can get a reference to this object and read and update provisioning information. The key of the dictionary is a `String` object and the value is a `String` or `byte[]` object. The `provisioning` prefix is reserved for keys defined by OSGi, other key names may be used for implementation dependent provisioning systems.

Any changes to the provisioning information will be reflected immediately in all the dictionary objects obtained from the Provisioning Service. Any updates to a dictionary will cause corresponding provisioning information to be updated and made persistent.
Because of the specific application of the Provisioning Service, there should be only one Provisioning Service registered. This restriction will not be enforced by the Framework. Gateway operators or manufactures should ensure that a Provisioning Service bundle is not installed on a device that already has a bundle providing the Provisioning Service.
The provisioning information has the potential to contain sensitive information. Also, the ability to modify provisioning information can have drastic consequences. Thus, only trusted bundles should be allowed to register and get the Provisioning Service. The `ServicePermission` is used to limit the bundles that can gain access to the Provisioning Service. There is no check of `Permission` objects to read or modify the provisioning information, so care must be taken not to leak the Provisioning Dictionary received from `getInformation` method.

Field Summary

`static java.lang.String`	**MIME_BUNDLE** MIME type to be stored in the extra field of a `ZipEntry` object for an installable bundle file.
`static java.lang.String`	**MIME_BUNDLE_URL** MIME type to be stored in the extra field of a ZipEntry for a String that represents a URL for a bundle.
`static java.lang.String`	**MIME_BYTE_ARRAY** MIME type to be stored in the extra field of a `ZipEntry` object for `byte[]` data.
`static java.lang.String`	**MIME_STRING** MIME type to be stored in the extra field of a `ZipEntry` object for String data.
`static java.lang.String`	**PROVISIONING_AGENT_CONFIG** The key to the provisioning information that contains the initial configuration information of the initial Management Agent.
`static java.lang.String`	**PROVISIONING_REFERENCE** The key to the provisioning information that contains the location of the provision data provider.
`static java.lang.String`	**PROVISIONING_ROOTX509** The key to the provisioning information that contains the root X509 certificate used to esatblish trust with operator when using HTTPS.
`static java.lang.String`	**PROVISIONING_RSH_SECRET** The key to the provisioning information that contains the shared secret used in conjunction with the RSH protocol.
`static java.lang.String`	**PROVISIONING_SPID** The key to the provisioning information that uniquely identifies the Service Platform.
`static java.lang.String`	**PROVISIONING_START_BUNDLE** The key to the provisioning information that contains the location of the

	bundle to start with `AllPermission`.
`static java.lang.String`	**PROVISIONING_UPDATE_COUNT** The key to the provisioning information that contains the update count of the info data.

Method Summary

`java.util.Dictionary`	**getInformation**() Returns a reference to the Provisioning Dictionary.

Field Detail

5.1.1 PROVISIONING_SPID

`public static final java.lang.String` **`PROVISIONING_SPID`**

The key to the provisioning information that uniquely identifies the Service Platform. The value must be of type `String`.

5.1.2 PROVISIONING_REFERENCE

`public static final java.lang.String` **`PROVISIONING_REFERENCE`**

The key to the provisioning information that contains the location of the provision data provider. The value must be of type `String`.

5.1.3 PROVISIONING_AGENT_CONFIG

`public static final java.lang.String` **`PROVISIONING_AGENT_CONFIG`**

The key to the provisioning information that contains the initial configuration information of the initial Management Agent. The value will be of type `byte[]`.

5.1.4 PROVISIONING_UPDATE_COUNT

`public static final java.lang.String` **`PROVISIONING_UPDATE_COUNT`**

The key to the provisioning information that contains the update count of the info data. Each set of changes to the provisioning information must end with this value being incremented. The value must be of type `Integer`.

5.1.5 PROVISIONING_START_BUNDLE

`public static final java.lang.String` **`PROVISIONING_START_BUNDLE`**

The key to the provisioning information that contains the location of the bundle to start with `AllPermission`. The bundle must have be previously installed for this entry to have any effect.

5.1.6 PROVISIONING_ROOTX509

`public static final java.lang.String` **`PROVISIONING_ROOTX509`**

The key to the provisioning information that contains the root X509 certificate used to esatblish trust with operator when using HTTPS.

5.1.7 PROVISIONING_RSH_SECRET

`public static final java.lang.String` **`PROVISIONING_RSH_SECRET`**

The key to the provisioning information that contains the shared secret used in conjunction with the RSH protocol.

5.1.8 MIME_STRING

`public static final java.lang.String` **`MIME_STRING`**

MIME type to be stored in the extra field of a `ZipEntry` object for String data.

5.1.9 MIME_BYTE_ARRAY

`public static final java.lang.String` **`MIME_BYTE_ARRAY`**

MIME type to be stored in the extra field of a `ZipEntry` object for `byte[]` data.

5.1.10 MIME_BUNDLE

`public static final java.lang.String` **`MIME_BUNDLE`**

MIME type to be stored in the extra field of a `ZipEntry` object for an installable bundle file. Zip entries of this type will be installed in the framework, but not started. The entry will also not be put into the information dictionary.

5.1.11 MIME_BUNDLE_URL

`public static final java.lang.String` **`MIME_BUNDLE_URL`**

MIME type to be stored in the extra field of a ZipEntry for a String that represents a URL for a bundle. Zip entries of this type will be used to install (but not start) a bundle from the URL. The entry will not be put into the information dictionary.

Method Detail

5.1.12 getInformation

`public java.util.Dictionary` **`getInformation`**`()`

Returns a reference to the Provisioning Dictionary. Synchronizing on this object must prevent changes being made by other threads in the Framework. This allows for atomic updates of a group of values. before releasing the synchronization of the `Dictionary` object, the thread must increment the value with the key PROVISIONING_UPDATE_COUNT if updates have been made.

---

6 Document Support

6.1 References

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

[2]. Helander, L-E. OSGi Entities. OSGi RFC 28.

[3]. Jennings, J. Remote Management Architecture, OSGi RFC 20.

6.2 Author's Address

Name	Lars-Erik Helander
Company	Gatespace AB
Address	Stora Badhusgatan 18-20, Gothenburg, Sweden
Voice	+46 31 743 98 43
e-mail	helander@gatespace.com

Name	William R Soley
Company	Sun Microsystems
Address	901 San Antonio Road Palo Alto, California USA 94303-4900
Voice	+1 **650 653-2549**
e-mail	william.soley@sun.com

Name	Ben Reed
Company	IBM
Address	
Voice	
e-mail	breed@almaden.ibm.com

6.3 Acronyms and Abbreviations

6.4 End of Document