FD - Provision Manager

Content

History

Version	Date	Adjustments
PA1	2008-03-13	First version of the document. Original document was found in MAS. (EERITOR)

1 Introduction

The purpose of this document is to explain the internal structure and functions of the Provision Manager component. The Provision Manager functions are specified in [1].

2 Function Structure

2.1 Overview

This picture shows which external components the Provision Manager depends on.

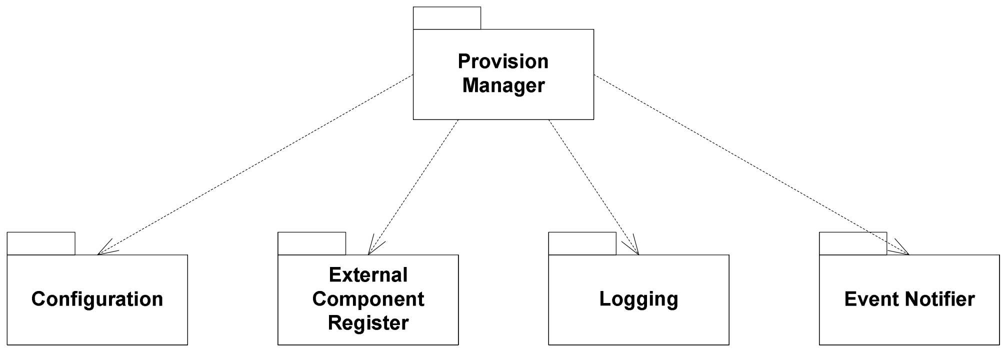

Figure 1 Component dependencies

- Configuration module is used to read the configuration.
- External Component Register is used to retrieve provisioning hosts.
- Logging is used for error and debug messages.
- Event Notifier is used to receive configuration changed events.

2.2 Modules

This chapter explains the different modules and classes and what functions they do. Note that not all classes are mentioned here, only the most important ones.

An overview of the main classes is shown in Figure 2.

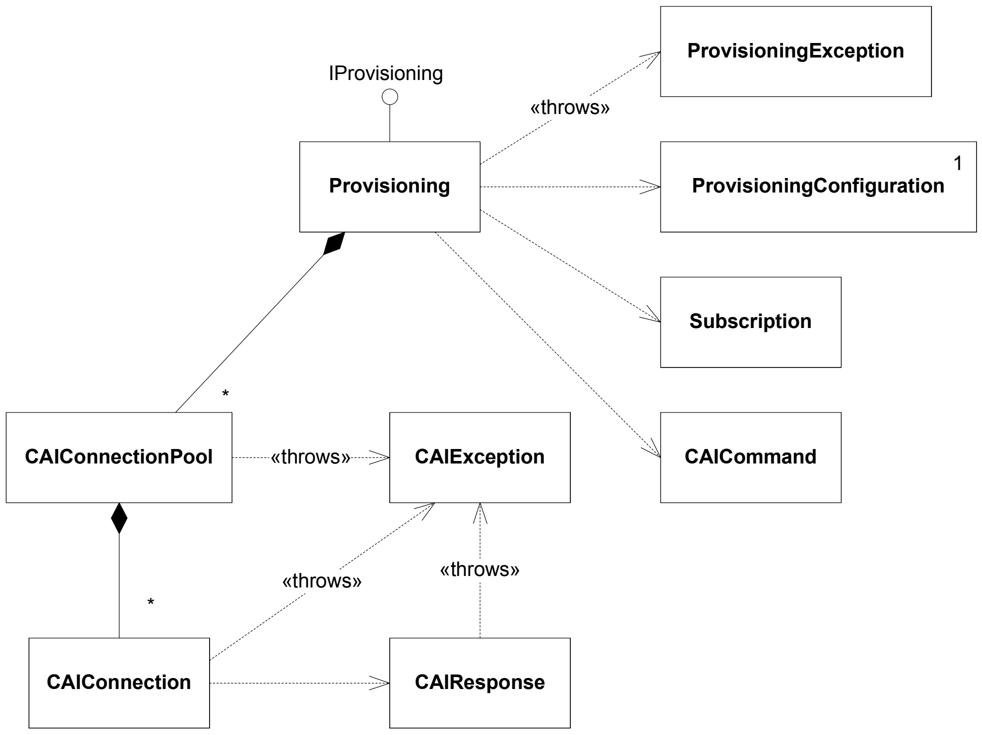

Figure 2 Overview Provision Manager

This component consists of two packages: *com.mobeon.masp.provisionmanager* and *com.mobeon.masp.provisionmanager.cai*

The CAI package is developed so that the provisionmanager package can use it as an API for the CAI protocol. By that it means that there are no "double aimed" dependencies between the packages. It is only the provisionmanager package that needs the CAI package and not the opposite.

2.2.1 com.mobeon.masp.provisionmanager

Main package for this component.

2.2.1.1 Subscription

This class models a subscriber. It contains a list of attribute-value pairs.

2.2.1.2 Provisioning

This is the class that implements the IProvisioning interface. It is created via the Spring framework at startup and contain references to the external interfaces that this component needs.

2.2.1.3 ProvisioningConfiguration

This is a class that contains functionality to read the configuration file used in the Provision Manager component.

2.2.1.4 *ProvisioningException*

This is an exception class that is thrown when some error occurred when calling the Provision Manager.

2.2.2 com.mobeon.masp.provisionmanager.cai

This package contains classes used to send CAI commands via the CAI protocol. See [2].

2.2.2.1 *CAIConnection*

This class handles the actual communication with a CAI server. It contains a connection (socket) to the server.

Have methods to send CAI commands and read the CAI response from the server.

The socket timeout value can be set.

Each time a connection is used the field *lastTouchedTime* is updated. This time can be requested (See below).

2.2.2.2 *CAIConnectionPool*

This is a class that has a pool of CAIConnection objects. Contains methods to get and return a connection from/back to the pool. A new connection is created if no free connections are available.

The connections are connected and "logged in" so they are ready for use.

These configurable parameters exist in the pool.

- Max pool size.
- Time to wait for a free connection if pool is full.
- Idle timeout limit for a connection (it will be physically closed if the time limit is reached). This is implemented by checking the *lastTouchedTime* in CAIConnection.

2.2.2.3 *CAICommand*

It is an abstract base class for CAI Commands. It has one method *toCommandString* that must be implemented by deriving classes.

Subclasses are listed here:

• LoginCommand	The LOGIN CAI command
• LogoutCommand	The LOGOUT CAI command
• CreateCommand	The CREATE CAI command
• DeleteCommand	The DELETE CAI command
• GetCommand	The GET CAI command

Each class returns the string used for the particular command it handles.

2.2.2.4 *CAIResponse*

Helper class used to parse the response that comes from the CAI server.

2.2.2.5 CAIException

This is an exception class for the CAI API. It contains an error message and an error code that comes in the response from the CAI server.

3 Function Behavior

This chapter describes more details of how the Provision Manager works.

3.1 Initialization

The Provision Manager is as all the components in MAS initialized via the Spring Framework. The external interfaces are injected into the Provisioning class when it is created.

These methods exist for this reason:

public void setConfiguration(IConfiguration configuration)

public void setEventDispatcher(IEventDispatcher eventDispatcher)

public void setServiceLocator(ILocateService locateService)

The Provisioning class exists as a singleton object in MAS.

3.2 Create

3.2.1 Successful create

This sequence diagram shows the flow for the create function.

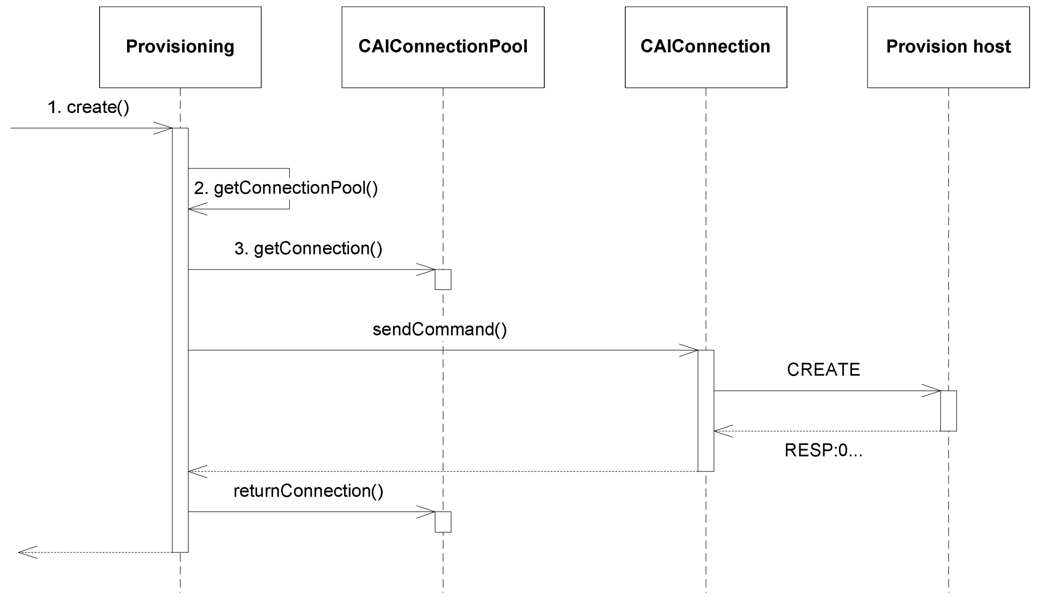

Figure 3 Sequence diagram for the create function

1. A client calls the create method on the Provisioning class.

2. Provisioning checks if there exists a connection pool for the adminuid that is passed as an argument. If not it is created. The pools are put in a map with adminuid as key.
3. A connection is requested from the connection pool. The pool checks if there are any free connections and if not a new connection is created. The connection is then returned.
4. A CAICreate command is created from the incoming Subscription object. This command is passed with the sendCommand call on the CAIConnection class.
5. The CAIConnection sends the CAI Create command on the socket to the CAIServer. The server responds with OK (code 0).
6. Provisioning returns the connection to the pool and the create call is returned.

3.2.2 Unsuccessful create

This sequence diagram shows the flow when the create function fails.

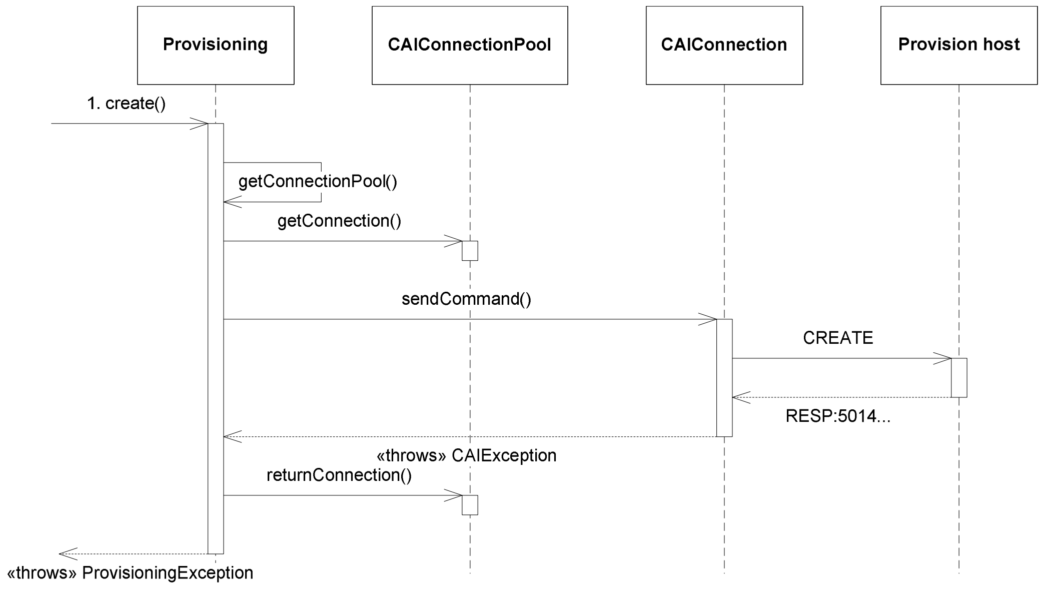

Figure 4 Sequence diagram for unsuccessful create

1. A client calls the create method on the Provisioning class.
2. Provisioning checks if there exists a connection pool for the adminuid that is passed as an argument. If not it is created. The pools are put in a map with adminuid as key.
3. A connection is requested from the connection pool. The pool checks if there are any free connections and if not a new connection is created. The connection is then returned.

4. A CAICreate command is created from the incoming Subscription object. This command is passed with the sendCommand call on the CAIConnection class.
5. The CAIConnection sends the CAI Create command on the socket to the CAIServer. The server responds with 5014 which means that an invalid cosdn was specified in the command. A CAIException is created and thrown.
6. Provisioning returns the connection to the pool and a ProvisioningException is thrown for the client to handle.

3.3 Delete

Same flow as for the Create function but a CAI Delete command instead of a CAI Create command.

3.4 Closed connection

An unused connection can be closed from the provisioning server end. To handle closed CAIConnections in the pool the CAIConnection class tries to reconnect to the server and retry the request when a SocketException occurs, see Figure 5.

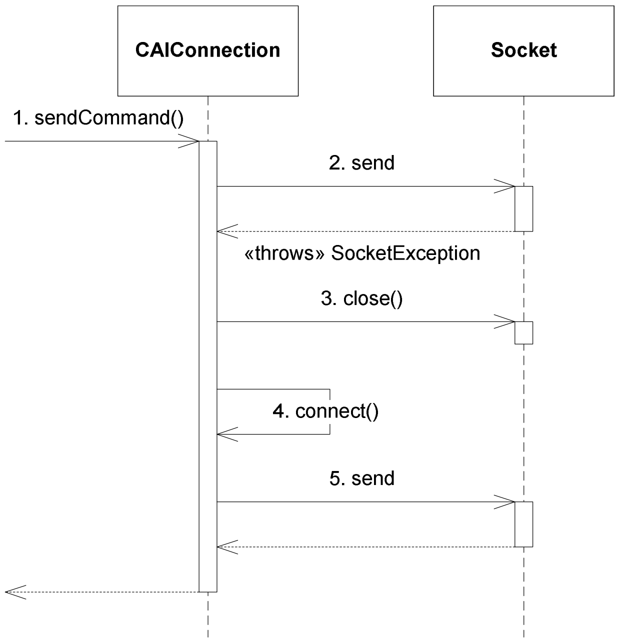

Figure 5 Closed connection

1. A command is sent to the CAIConnection.
2. When sending the command through the Socket, a SocketException occurs.
3. The socket is closed.
4. CAIConnection creates and connects to a new socket.

5. The command is resent successfully.

3.5 Configuration reload

MAS do not have to be restarted in order to update the configuration for this component.

Provision Manager is registered as an Event receiver in the EventDispatcher interface. When a ConfigurationChanged event is received the Provision Manager updates the configuration in the same way it did at start-up. The stored configuration parameters are cleared and the new ones are read from the new configuration.

4 References

[1] FS Provision Manager
18/FS-MAS0001 Uen

[2] IWD Provisioning Interface
2/155 19-CRH 109 086 Uen

5 Terminology

CAI	Customer Administration Interface
API	Application Programming Interface