Prepared (also subject responsible if other)		No.		
Marcus Haglund		19/FD-MAS0001		
Approved	Checked	Date	Rev	Reference
Per Berggren		2008-03-26	A	

FD - Charging Account Manager

History

Date	Rev	Description
2008-03-26	A	First version of the document.

1 Introduction

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

2 Function Structure

2.1 Overview

This picture shows which external components the Charging Account Manager on.

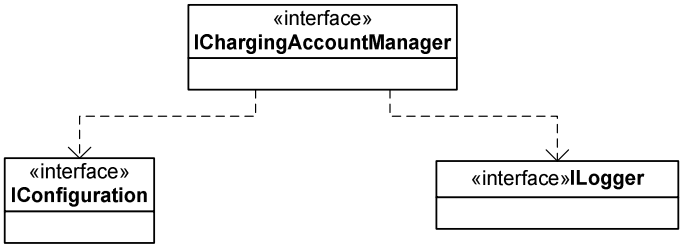

Figure 1 Component dependencies

- Configuration module is used to read the configuration.
- Logging is used for error and debug messages.

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.

This component consists of one package:
com.mobeon.masp.chargingaccountmanager

2.2.1 ChargingAccountManager

This is the class that implements the IChargingAccountManager interface. It is created via the *Spring* framework at startup and contain references to the external interfaces that this component needs. The class exists as a singleton object in MAS.

The method *sendRequest* takes a ChargingAccountRequest as parameter and sends the data via the AirClient class; the response data is then put into a ChargingAccountResponse.

This class also is responsible for error handling in case there is some error with the Air communication.

2.2.2 ChargingAccountRequest

The ChargingAccountRequest class holds the parameters that are going to be sent in a request. It also has a name which is the same name as the message name in the UCIP protocol.

Depending on the configuration each parameter is converted from the incoming string (in the *addParameter* function) to the specific XML-RPC data type for the parameter.

2.2.3 ChargingAccountResponse

The ChargingAccountResponse class holds the parameters that are present in a response from the Air server.

The *getParameter* method converts the parameter value from the XML-RPC data type to a string.

2.2.4 ChargingAccountException

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

2.2.5 ChargingAccountConfiguration

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

2.2.6 AirClient

This class implements the IAir interface and handles the communication with an AIR server. It uses the Apache XML-RPC API for this.

XML-RPC uses HTTP as transport protocol so there are no persistent connections to the AIR server. A new connection is opened and closed for each request.

3 Function Behavior

This chapter describes in detail how the Charging Account Manager works.

3.1 Initialization

The Charging Account Manager is as all the components in MAS initialized via the Spring Framework. The external interface IConfiguration is injected into the ChargingAccountManager class when it is created.

This method exists for this reason:

public void setConfiguration(IConfiguration configuration)

3.2 Load balancing and failover

The ChargingAccountManager has a list of AirClient objects that is created from the configuration. The clientId parameter is used to get a client from the list. The *getNextClientId* method can be used to get another client.

If the communication with an AIR Node fails the AirClient that is assigned to this node is set to disabled and an exception is thrown. The *getNextClientId* method will not use a disabled client when returning an id.

3.3 Send Account Request

3.3.1 Successful request

This sequence diagram shows the flow for the when an account request is sent.

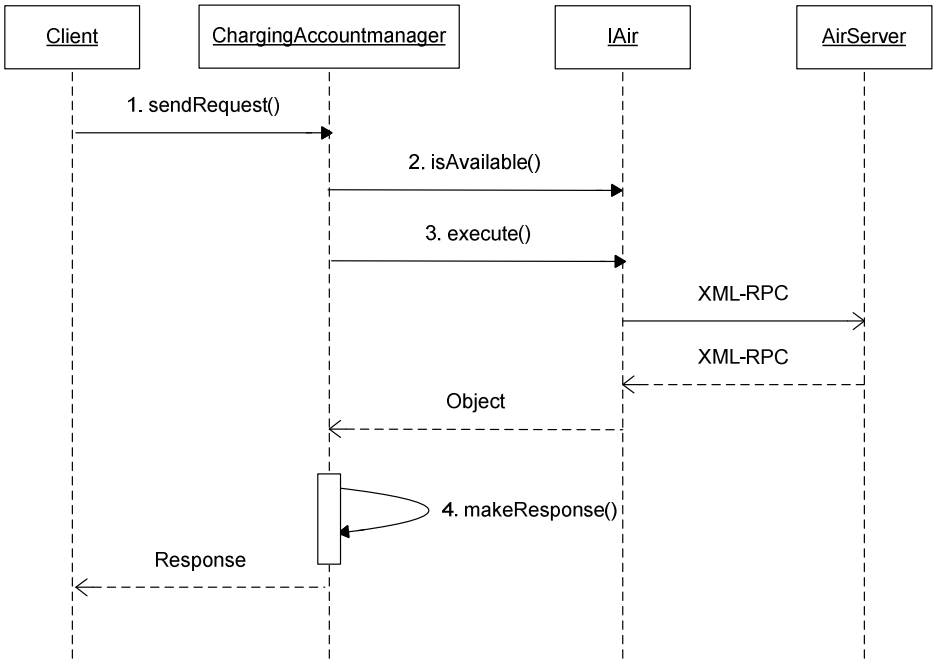

Figure 2 Sequence diagram for the sent account request function.

1. A client calls the sendRequest method on the ChargingAccountManager class. A ChargingAccountRequest and the clientId are sent as argument.

2. ChargingAccountManager checks if the AirClient referenced by clientId is available.

3. The data from the request is added to a List and execute is called on the AirClient. The data is sent with help of the Apache XML-RPC API to the AirServer. The AirServer responds OK, and the response is put into an object by the API.

4. Data from the object is put into a ChargingAccountResponse and is returned to the client.

3.3.2 Unsuccessful request

This sequence diagram shows the flow when an account request fails.

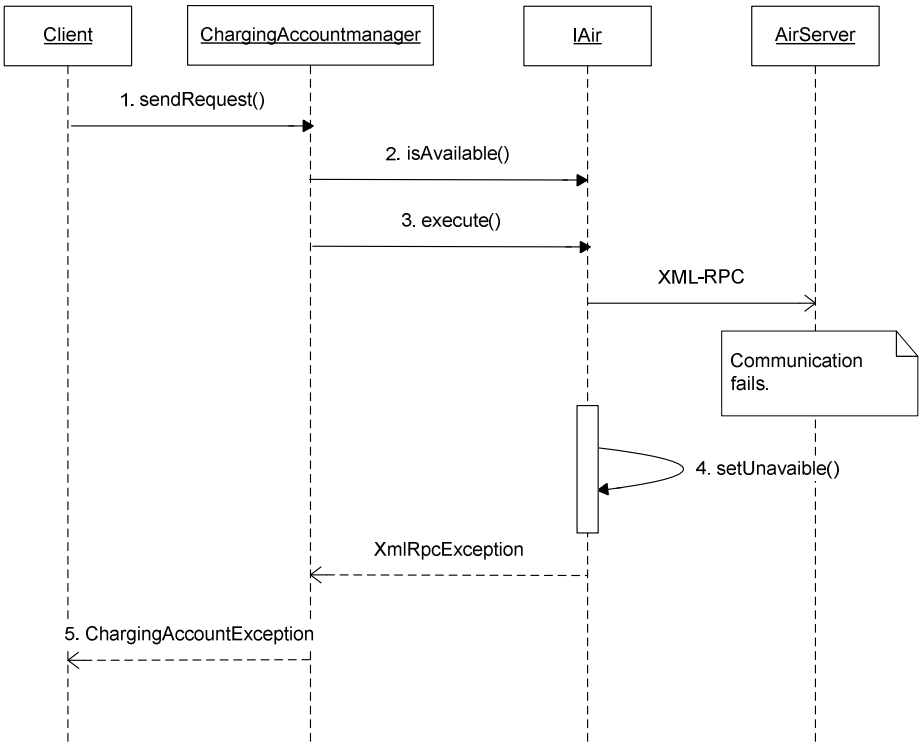

Figure 3 Sequence diagram when the sent account request fails.

The steps 1, 2, 3 are same as the successful flow.

4. The communication towards the AirServer fails for some reason so the client is set to unavailable and an XmlRpcException is thrown.

5. A ChargingAccountException is thrown to the client.

4 References

[1] FS – Charging Account Manager
19/FS-MAS0001

5 Terminology

AIR	Account Information and Refill system
UCIP	User Communication Integration Protocol