Prepared (also subject responsible if other)		No.		
Marcus Haglund		19/FS-MAS0001		
Approved	Checked	Date	Rev	Reference
Per Berggren		2008-02-14	A	

FS - Charging Account Manager

History

Date	Rev	Description
2008-02-14	A	First version of the document.

1 Introduction

This document describes the function of the Charging Account Manager component. This component is used to handle the access to an Account Information and Refill server (AIR).

2 Function Requirements (Commercial)

This paragraph is intentionally left blank.

3 Function Specification (Design Related)

3.1 Introduction

The component uses the UCIP protocol to communicate with the AIR server. Via this protocol, clients can ask for account information for a specific user. The protocol is based on XML-RPC and sends XML data over HTTP.

Also see reference UCIP [1].

The Charging Account Manager also imports the ILog and IConfig interfaces. See picture below:

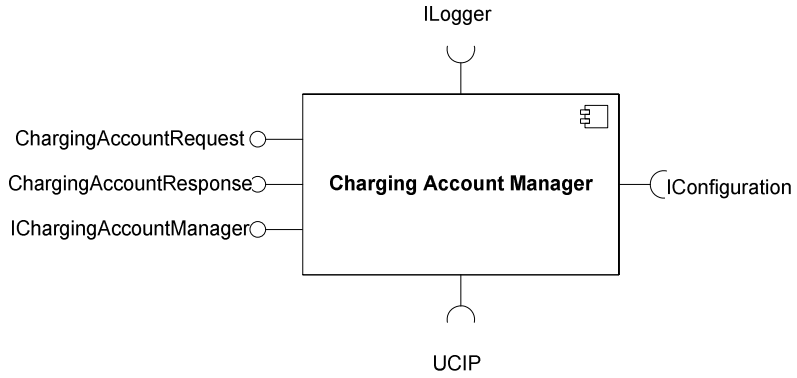

Figure 1 Charging Account Manager interfaces.

3.2 Exported Interfaces

3.2.1 IChargingAccountManager

This interface is used to send requests for charging account information. It has the following methods:

ChargingAccountResponse sendRequest(ChargingAccountRequest request, int clientId)
This method sends a charging request to an AIR server referenced by clientId and waits for a response.

Int getNextClientId()
Retrieves a client id for a specific assigned AIR node. This node can be used during a call.

3.2.2 ChargingAccountRequest

This is a class for data that is sent via the ChargingAccountManager interface. The following methods are available:

setName(String name)
Sets name on the request.

addParameter(String name, String value)
Adds a parameter to the request. If the parameter is located in a struct it will be added into the correct one according to the configuration.

3.2.3 ChargingAccountResponse

This is a class for data that is received via the ChargingAccountManager. It has the following methods:

int getResponseCode()
This method returns the response code from the AIR Server.

String getParameter(String name)
Retrieves a parameter from the response.

3.2.4 ChargingAccountException

This exception is thrown when something goes wrong in the Charging Account Manager.

3.3 Imported Interfaces

ChargingAccountManager uses the following external interfaces:

- ILogger for logging
- IConfiguration for configuration

3.4 Functions

3.5 Data type conversion

The UCIP protocol supports different data types for examples string, integer, boolean and dates. The Charging Account Manger converts the strings that are taken as input parameters to the correct data type depending on the parameter.

Same conversion is also made when a parameter is returned via the *getParameter* method.

A Boolean parameter returns the strings “*true*” or “*false*”

3.5.1 Request account information

This sequence diagram shows when an account request is sent:

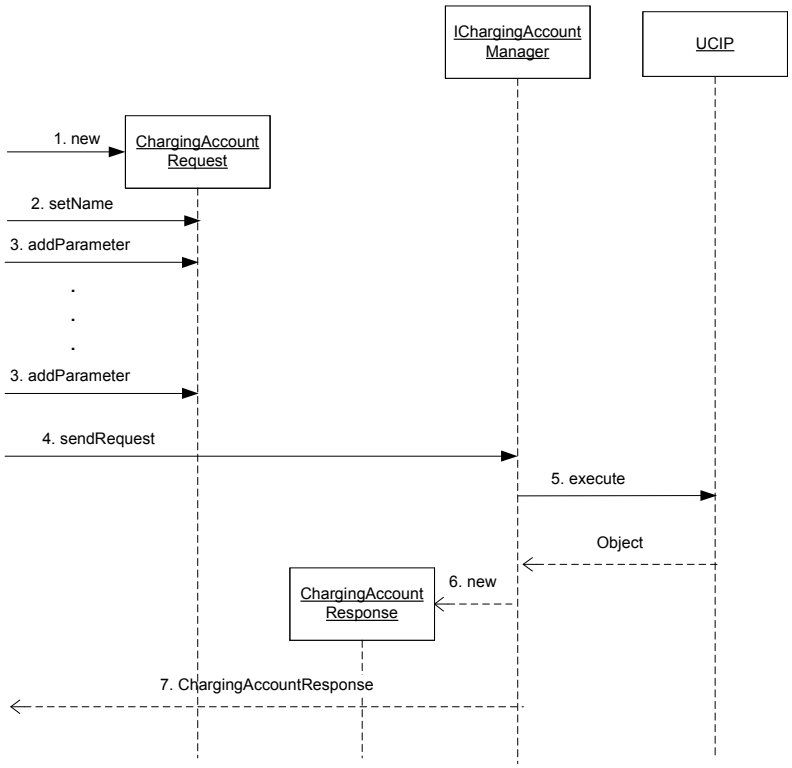

Figure 2. Account Request

1. A ChargingAccountRequest is created.
2. Name is set
3. A parameter is added (one or many times)
4. SendRequest is called.
5. The request is sent via the AIR (UCIP) interface.
6. Data from the response is added into a new ChargingAccountResponse.
7. The response is returned to the caller.

3.5.2 Thread Safe

The Charging Account Manager is thread safe.

3.6 Configuration

The following is configurable in Charging Account Manager:

- List of AIR server nodes. Nodes have an URI (host, port and path), username and password.
- Elements and groups in the UCIP protocol (see below).

3.6.1 Element and group configuration

In the UCIP protocol each element can be of type string, integer, boolean or date. The Charging Account Manager interface allows only strings to be set and get so a data-conversion must be made when fetching the data. To ease up this, the parameters that describe an element can be configurable.

Example:

```
<element name="accountActivationFlag" type="boolean"/>
<element name="accountGroupID" type="integer"/>
<element name="accumulatorEndDate" type="date"/>
```

Note that string is the default data type for a parameter, so if the parameter is not present it will be treated as a string.

Some parameters are also grouped together. The user of this interface has no option to set a structure of parameters, so when a parameter that belongs to a group is set via the *addParameter* method the Charging Account Manager puts it into the correct group.

The element's that belongs to a certain group is also configurable. See example below:

```
<elementgroup parent="messageCapabilityFlag"
structtype="struct">

     <member name="promotionNotificationFlag"/>
     <member name="firstIVRCallSetFlag"/>
     <member name="accountActivationFlag"/>
</elementgroup>
```

3.7 Logging

All calls to interface methods are logged at info level. Method name and parameter values are logged.

4 Capabilities

4.1 Authentication

Authentication towards the Air server is used via HTTP basic authentication. It is used by adding an Authorization header and base64 encoded username and password as value. Example:

Authorization: Basic dWlkOnB3ZA==

4.2 Load balancing and redundancy

The Air nodes (hosts) can be accessed in a *round-robin per call* fashion by using the clientId parameter. This means that one session (call) in MAS can use the same AIR node during the duration of the call.

If the communication with an AIR Node fails for some reason Charging Account Manager will throw an exception, and the failed node is set to disabled for a couple of minutes. After a while a new try is made to that node to see if it is back.

These errors are also logged to help debug the reasons of the failures.

4.3 Connection Timeout

Connection Timeout value (if the server does not respond in time) can be set in the Charging Account Manager.

5 References

[1] AIR Programmers Guide UCIP
6/1553-FAM 901 108/3 Uen

6 Terminology

AIR	Account Information and Refill system
UCIP	User Communication Integration Protocol