AIR-UCIP Document

Mobilink Pakistan

Given By Intelligent Network Planning Team

Mobilink

Pakistan’s Favourite Cellular Service

Contents

1 INTRODUCTION 4
1.1 Purpose and Scope 4
1.2 Typographic Conventions 4
2 GENERAL 6
3 UCIP in Use 7
3.1 Introduction 7
3.2 Message Description 7
4 Syntax Notation 10
4.1 XML-RPC Syntax 10
4.1.1 Example Request 11
4.1.2 Example Response 11
4.1.3 Example Fault Response 12
5 Messages 13
5.1 How to Read the Tables 13
5.2 Fault Response 14
5.8 Get Balance And Date 15
5.8.1 Request 15
5.8.2 Response 15
5.8.2.1 Response Codes 17
5.15 UpdateBalanceAndDate 17
5.15.1 Request 17
.... 19
5.15.2 Response 19
Table 30 UpdateBalanceAndDate Response 19
.... 19
.... 20
5.15.2.1 Response codes 20
5.22 Common Structures and Arrays 20
5.22.1 accountAfterRefill and accountBeforeRefill 20
.... 21
5.22.2 accountFlags 21
5.22.3 accountFlagsAfter and accountFlagsBefore 22
5.22.6 chargingRequestInformation 22
5.22.7 chargingResultInformation 23
5.22.12 dedicatedAccountInformation 23

5.22.13 dedicatedAccountChangeInformation 25
.... 26
5.22.16 dedicatedAccountSelection 27
5.22.17 dedicatedAccountUpdateInformation 27
.... 29
5.22.20 messageCapabilityFlag 29
5.22.22 offerInformationList 30
6 Elements 32
6.5 accountValue1 and accountValue2 32
6.14 adjustmentAmountRelative 32
6.19 aggregatedBalance1 and aggregatedBalance2 (PC:05225) 32
6.20 allowCropOfCompositeDedicatedAccounts 32
6.23 cellIdentifier (PC:08875) 33
6.44 creditClearancePeriod 33
6.45 currency1 and currency2 33
6.61 externalData1, externalData2, externalData3 and externalData4 33
6.73 languageIDCurrent 34
6.78 mainAccountValueNew (PC:05163) 34
6.83 negativeBalanceBarringDate 34
6.93 originHostName 35
6.94 originNodeType 35
6.95 originOperatorID 36
6.96 originTimeStamp 36
6.97 originTransactionID 36
6.123 requestFirstAccessibleAndExpiredBalanceAndDateFlag 37
6.130 requestSubDedicatedAccountDetailsFlag 37
6.139 serviceClassCurrent 38
6.156 serviceFeeExpiryDate 38
6.157 serviceFeeExpiryDateCurrent 38
6.158 serviceFeeExpiryDateRelative 39
6.167 serviceRemovalPeriod 39
6.171 subscriberNumber 39
6.172 subscriberNumberNAI 39
6.175 supervisionExpiryDate 40
6.176 supervisionExpiryDateCurrent 40
6.177 supervisionExpiryDateRelative 40

6.181 temporaryBlockedFlag 41
6.185 transactionCode 41
6.186 transactionCurrency 42
6.187 transactionType 42
7 Data Type Definitions 43
7.1 Data Types 43
(1) This protocol expects the ASCII strings to be coded in 7-bit USASCII (ISO 646 45
7.2 Element formats 45
7.3 Logical Values 46
8 State Behavior 47
8.1 Introduction 47
8.2 Traffic Cases 47
10 Compatibility Mechanisms 48
11 Transfer Mechanism 49
11.1 Introduction 49
11.2 Request Mechanism 49
11.3 Response Mechanism 50
11.4 Error Handling 50
11.5 Load Balancing and Fail Over 50
11.6 Authentication, Authorization and Security 51
11.7 TCP Level Compliance 51
12 Protocol Format Changes 52
12.1 Version 4.1 52
12.2 Version 4.0 53
12.3 Version 3.5 58
13 Commands Sample for Vendor 58

1 INTRODUCTION

This section covers the introduction.

1.1 Purpose and Scope

The purpose of this Programmers Guide - UCIP is to describe how to handle User Communication Integration Protocol (UCIP) in the AIR system, using the IP-based protocol version 4.1 the Account Information and Refill server (AIR).

The following are included:

- UCIP in Use
- Syntax Notation
- Messages
- Elements
- Data Type Definitions
- State Behavior
- Examples
- Compatibility Mechanisms
- Transfer Mechanisms

The target groups for this document are as follows:

- Application Designer
- Application Administrator
- System Administrator

For more information regarding the target groups, see AIR Customer Product Information - Overview, Reference [1].

1.2 Typographic Conventions

This document follows a set of typographic rules that make the document consistent and easy to read. The typographic conventions is found in the AIR Customer Product Information - Overview, Reference [1].

Elements and data types that are specific for product customization are marked with PC in the section describing the element or data type. Functions and characteristics of a product customization is not part of the standard delivery.Contact your local Ericsson office for more information regarding ordering and Compatibility.

All elements of a PC message are also product customizations when inside such a message. If an element or data type is marked PC within an otherwise standard record, is noted as such in the relevant chapters.

The following applies to functions marked with (PC). The PC-tag is placed in the beginning and end of the description, or included in the header, where the complete section is dedicated to this type of function.

Elements and data types that are specific for product customization are marked with the product customizations unique id in the tables for the messages where the elements or data types are included. An example of a product customization unique id is: PC: 01234. Functions and characteristics of a Product customization is not part of the standard delivery, contact your local Ericsson office for more information regarding ordering and compatibility.

2 GENERAL

The AIR system is used by connected systems to make enquiries on account balance information of prepaid accounts and to refill prepaid accounts in a mobile telephony network.
For a general description of AIR, see AIR Network Element Description, Reference [2]..

3 UCIP in Use

This section covers UCIP in use.

3.1 Introduction

UCIP is intended for user self-services such as Adjustments, Account Refill, and Account Enquiries and to extract account details. UCIP is an IP-based protocol used for integration towards the AIR server from the external application. UCIP is an XML over HTTP based protocol, which makes it easy to integrate with a central integration point within a network. The protocol supports both session as well as event based clients. A UCIP request is sent to one of the AIR servers within the network and for redundancy purposes it is required to have N+1 AIR system in the network. The elements part of this protocol will be transferred using XML-RPC messages. An XML-RPC message is a body of an HTTP-POST request, see Section 3.2 on page 5.

3.2 Message Description

The messages that are included in UCIP are listed and described in Table 1

Message	Description	DR Generated
GetAccountDetails	This message is used in order to validate and tailor the user communication. The GetAccountDetails message is used to obtain the account and subscriber information.	-
GetAccountManagementCounters (ID:01871)	This message is used to retrieve the account management counters.	-
GetAccountServiceFeeData (PC:06214)	This message is used to fetch service fee data assigned to a subscriber.	-
GetAccumulators	This message is used when the accumulator values should be obtained. It is also used to obtain the start and end dates related to accumulators.	-
GetAllowedServiceClasses (ID:01871)	This message is used to list service classes that the subscriber is allowed to change to.	-

Message	Description	DR Generated
GetBalanceAndDate	This message makes it possible to enquire the balances and expiry dates of an account associated with a specific subscriber identity.	-
GetFaFList	This message is used to fetch and list the Family and Friends (FaF) numbers with attached FaF indicators.	-
GetOffers	This message is used to fetch offers assigned to an account.	-
GetRefillOptions	This message is used to fetch the refill options.	-
GetUsageThresholdsAndCounters	This message is used to fetch usage counters.	-
Refill	This message is used to apply a refill from an administrative or external system to a prepaid account associated with a specific subscriber identity. It can be a voucherless refill from for example a bank gateway, or a voucher refill made through IVR or WEB gateway.	AIR,Account Database
UpdateAccountDetails	This message is used to change the account information when a user initiated change is performed. For example, to set a flag after the first IVR call has been made or a change of the preferred language.	Account Database
UpdateBalanceAndDate	This message is used to make adjustments to the account balances and expiry dates. It is possible to do both relative adjustments (negative or positive) or set the dates to an absolute date for both main and dedicated accounts.	Account Database, AIR
UpdateCommunityList	This message set or updates the list of communities which the account belong to.	Account Database
UpdateFaFList	This message is used to update the Family and Friends list for either the account or subscriber.	Account Database
UpdateOffer	This message is used to assign a new offer or update an already assigned offer to an account.	Account Database
UpdateServiceClass	This message is used to update the service class for the subscriber.	Account Database

Message	Description	DR Generated
UpdateSubscriberSegmentatio n	This message is used in order set or update the parameters which are used for subscriber segmentation.	Account Database
UpdateUsageThresholdsAndC ounters	This message updates the value of usage counters and usage thresholds.	-

4 Syntax Notation

This section covers syntax notation.

4.1 XML-RPC Syntax

XML-RPC is a Remote Procedure Calling protocol and the body of an XML-RPC request is formatted using XML. A procedure executes on the AIR server and the value it returns is also formatted in XML. The interface to the AIR server interface uses XML-RPC over HTTP.

For information about XML-RPC, see World Wide Web Consortium. Extensible Markup Language (XML) 1.0, Reference [7] and XML-RPC Home Page, Reference [8].

The HTTP connections are persistent. The users of the connections are expected not to open more connections than they need and to reuse them. When a connection is no longer needed it must be closed to secure that other can use them. The number of connections needed by each client depends on
the load generated from the client and the latency in the responses from AIR.

The protocol uses the `struct` construct to achieve named values which is necessary as some parameters are optional. The body of the response is a single XML structure, a `methodResponse`, this can contain a single `params`. The `params` contains a single `param` which in turn contains a single `value`.

The `methodResponse` could also contain a `fault` containing a `faultCode` and a `faultString`. See Section 4.1.3 on page 11 for an example of a fault response.

Note:
`methodResponse` can *not* contain both a `fault` and a `params`.
For an example of a request, see Section 4.1.1 on page 9.
For an example of a response, see Section 4.1.2 on page 10.

4.1.1 Example Request

Example of a successful request:

Example Request
Example of a successful request:

```
<?xml version="1.0"?>
<methodCall>
<methodName>Request.getName</methodName>
<params>
<param>
<value>
<struct>

<member>
<name>RequestNumber</name>
<value><string>00471627612</string></value>
</member>
</struct>
</value>
</param>
</params>
</methodCall>
```

4.1.2 Example Response

This section holds examples.

Example of a successful response 1:

```
<?xml version="1.0"?>
<methodResponse>
<params>
<param>
<value>
<struct>
<member>
<name>agent</name>
<value><string>PrePaidServices</string></value>
</member>
</struct>
</value>
</param>
</params>
</methodResponse>
```

Example of a successful response 2:

```
<?xml version="1.0"?>
<methodResponse>
<params>
```

```
<param>
<value>
<struct>
<member>
<name>responseCode</name>
<value><i4>0</i4></value>
</member>
<member>
<name>languageIDCurrent</name>
<value><i4>3</i4></value>
</member>
<member>
<name>AccumulatorInformation</name>
<value>

<array>
<data>
<value>
</struct>
<member>
<name>
accumulatorID
</name>
<value>
<i4>2</i4>
</value>
</member>
<member>
<name>
accumulatorValue
</name>
<value><i4>3</i4></value>
</member>
</struct>
</value>
</data>
</array>
</value>
</member>
</struct>
</value>
</param>
</params>
</methodResponse>
```

4.1.3 Example Fault Response

This section holds examples of fault response.

Example of a fault response:

```
<?xml version="1.0"?>
<methodResponse>
<fault>
<value>
<struct>
<member>
<name>faultCode</name>
<value><int>1001</int></value>
</member>
<member>
<name>faultString</name>
<value><string>Mandatory field missing.</string></value>
```

```
</member>
</struct>

</value>
</fault>
</methodResponse>
```

5 Messages

This section holds all request messages that are sent towards the AIR server and all response messages that are replied to the interworking network entity from where the request originated. The messages are described in separate sections.

5.1 How to Read the Tables

Both in requests and responses each parameter is listed with condition stating
Whether the parameter is mandatory or optional.

In requests a column named Type state these condition. An optional parameter can in the request be omitted if not used, but lacking a mandatory parameter will result in an error response.
For responses there are two columns in the tables, indicating error condition marked as ‘‘Error’’ and successful request marked as ‘‘Normal’’. An optional parameter that has no value or no new value might be omitted, this is decided by the application using the protocol. Both in requests and responses there is an optional column called “Subtype” where additional information is given concerning the relation between parameters. Example: Two parameters might both be optional, but if one of them is given it is mandatory to give the other since they together form a function. Likewise it can be that parameters are mutually exclusive. For a description of the notation for each data element see Table 2.
For a description of the notation for the Subtype, see Table 3.

TABLE 2   Data element notation

Notation	**Description**
M	Mandatory data element.
O	Optional data element.
N/A	Not Applicable data element.

TABLE 3 Subtype notation

Notation	**Description**
XOR	The parameters are mutually exclusive, only one of them can be given

Notation	**Description**
AND	If one of the parameters is given, it is mandatory to give the other
OR	At least one of the parameters must be given

5.2 Fault Response

A fault response message is returned when the server cannot process a request. This is valid when the request is not possible to parse, the operation is unknown by the server, mandatory fields are missing or fields containing illegal data types or value ranges. It is also valid when the AIR server is overloaded. In the latter scenario the client can resend the request to the same or another AIR server. The fault response follows the XML-RPC fault construct and is a valid return message for all requests that are supported by UCIP, see Table4.

Table 4 Fault response message

Element	**Reference**	**Type**
faultCode	Section 6.68	M
faultString	Section 6.69	M

5.8 Get Balance And Date

The message *GetBalanceAndDate* is used to perform a balance enquiry on the account associated with a specific subscriber identity. Also lifecycle dates are presented. Information is given on both main and dedicated accounts.

5.8.1 Request

Table 15 Get balance and date

Element	**Reference**	**Type**
originNodeType	Section 6.94	M
originHostName	Section 6.93	M
originTransactionID	Section 6.97	M
originTimeStamp	Section 6.96	M
subscriberNumberNAI	Section 6.172	O
subscriberNumber	Section 6.171	M
messageCapabilityFlag	Section 5.22.20	O
dedicatedAccountSelection	Section 5.22.16	O

Element	**Reference**	**Type**
chargingRequestInformation (ID:01871)	Section 5.22.6	O
requestSubDedicatedAccountDetailsFlag (1)	Section 6.130	O
requestFirstAccessibleAndExpiredBalanceAndDateFlag	Section 6.123	O

(1) *will also include requestFirstAccessibleAndExpiredBalanceAndDate if set*

5.8.2 Response

TABLE 16 GetBalanceAndDate Response

Element	**Reference**	**Normal**	**Subtype**	**Error**
responseCode	Section 5.8.2.1	M		M
originTransactionID	Section 6.97	M		O
serviceClassCurrent	Section 6.139	M		N/A
currency1	Section 6.45	O	AND currency1 is also mandatory in case dedicatedAccountValue1 is returned	N/A
accountValue1	Section 6.5	O		N/A
aggregatedBalance1 (PC:05225)	Section 6.19	O		N/A

Element	**Reference**	**Normal**	**Subtype**	**Error**
currency2	Section 6.45	O	AND currency2 is also mandatory in case dedicated AccountValue2 is returned	N/A
accountValue2	Section 6.5	O		N/A
aggregatedBalance2 (PC:05225)	Section 6.19	O		N/A
dedicatedAccountInformation	Section 5.22.12	O		N/A
supervisionExpiryDate	Section 6.175	O		N/A
serviceFeeExpiryDate	Section 6.156	O		N/A
creditClearanceDate	Section 6.44	O		N/A
serviceRemovalDate	Section 6.167	O		N/A
languageIDCurrent	Section 6.73	M		O
temporaryBlockedFlag	Section 6.181	O		N/A
chargingResultInformation (ID:01871)	Section 5.22.7	O		O
accountFlagsAfter	Section 5.22.3	O		N/A
account FlagsBefore	Section 5.22.3	O		N/A
offerInformationList	Section 5.22.22	O		N/A

5.8.2.1 Response Codes

The following response codes could be included in a GetBalanceAndDate response:

0, 100, 102, 123, 124, 126, 137, 139, 197, 999.

5.15 UpdateBalanceAndDate

The message *UpdateBalanceAndDate* is used by external system to adjust balances, start dates and expiry dates on the main account and the dedicated accounts. On the main account it is possible to adjust the balance and expiry dates both negative and positive (relative) direction and it is also possible to adjust the expiry dates with absolute dates. The dedicated accounts balances, start dates and expiry dates could be adjusted in negative and positive direction or with absolute values.
Note:
It is not possible to do both a relative and an absolute balance or date set for the same data type (example: it is possible to either set an absolute OR a relative adjustment to the service fee expiry date).
It is also possible to set the Service removal and Credit clearance periods on account.

5.15.1 Request

Table 29 UpdateBalanceAndDate

Element	Reference	Type	Subtype
originNodeType	Section 6.94 page 106	M	

Element	Reference	Type	Subtype
originHostName	Section 6.93	M	
originTransactionID	Section 6.97	M	
originTimeStamp	Section 6.96	M	
subscriberNumberNAI	Section 6.172	O	
messageCapabilityFlag	Section 5.22.20	O	
subscriberNumber	Section 6.171	M	
originOperatorID	Section 6.95	O	
transactionCurrency	Section 6.186	O	Mandatory in case the adjust change the value on any account
adjustmentAmountRelative	Section 6.14	O	XOR
mainAccountValueNew (PC:05163)	Section 6.78	O	
dedicatedAccountUpdateInformation	Section 5.22.17	O	
allowCropOfCompositeDedicatedAccounts	Section 6.20	O	
supervisionExpiryDateRelative	Section 6.177	O	XOR
supervisionExpiryDate	Section 6.175	O	
serviceFeeExpiryDateRelative	Section 6.158	O	XOR
serviceFeeExpiryDate	Section 6.156	O	
creditClearancePeriod	Section 6.44	O	

serviceRemovalPeriod	Section 6.167	O	
transactionType	Section 6.187	O	
transactionCode	Section 6.185	O	
externalData1	Section 6.61	O	
externalData2	Section 6.61	O	
serviceFeeExpiryDateCurrent	Section 6.157	O	
supervisionExpiryDateCurrent	Section 6.176	O	
serviceClassCurrent	Section 6.139	O	
cellIdentifier (PC:08875)	Section 6.23	O	

5.15.2 Response

Table 30 UpdateBalanceAndDate Response

Element	**Reference**	**Normal**	**Error**
responseCode	Section 5.15.2.1	M	M
originTransactionID	Section 6.95	M	O
originOperatorID	Section 6.97	O	O
currency1	Section 6.45	O Mandatory in case acco untValue1 present.	N/A
currency2	Section 6.45	O Mandatory in case acco untValue2 present.	N/A

Element	Reference	Normal	Error
negativeBalanceBarringDate	Section 6.83	O	N/A
accountFlagsAfter	Section 5.22.3	O	N/A
accountFlagsBefore	Section 5.22.3	O	N/A
dedicatedAccountChangeInformation	Section 5.22.13	O	N/A
accountValue1	Section 6.5	O	N/A
accountValue2	Section 6.5	O	N/A

5.15.2.1 Response codes

The following response codes could be included in an UpdateBalanceAndDate

response:

0, 100, 102, 104, 105, 106, 121, 122, 123, 124, 126, 136, 139, 153, 163, 164,

167, 204, 212, 999.

5.22 Common Structures and Arrays

This section covers common structures and arrays.

5.22.1 accountAfterRefill and accountBeforeRefill

The *accountBeforeRefill and accountAfterRefill* are enclosed in a <struct> of their own. accountBeforeRefill and accountAfterRefill contains financial and lifecycle data that might be affected during a refill. The accountBeforeRefill struct carries data on account BEFORE the refill is applied. This information can be requested in order to be able to display the start values before refill, in order to give the user the ability to verify the start conditions of the account when doing the refill. The accountAfterRefill struct carries data on account AFTER the refill is applied.
This information in the response and gives the result on the account after refill and promotions are added.

- In accountBeforeRefill the serviceClassTemporaryExpiryDate and serviceClassOriginal are returned only if the account has a temporary service class.
- In accountAfterRefill the serviceClassTemporaryExpiryDate and serviceClassOriginal are always returned when the account is assigned a temporary service class.
- The promotionPlanID is returned in accountBeforeRefill if a promotion plan ID exists for the account. In accountAfterRefill it is only returned if a promotion plan progression has been made.
- The serviceFeeExpiryDate is only returned in accountAfterRefill when service fee date is changed. In accountBeforeRefill it is always returned if it exists in the account database.
- The supervisionExpiryDate is only returned in accountAfterRefill when supervision date is changed. In accountBeforeRefill it is always returned if it exists in the account database.

- The creditClearanceDate is only returned in accountAfterRefill if supervision period or credit clearance removal period is changed. In accountBeforeRefill it is always returned if it exists in the account database.
- The serviceRemovalDate is only returned in accountAfterRefill if service fee date or service removal period is changed. In accountBeforeRefill it is always returned if it exists in the account database.

TABLE43 accountBeforeRefill and accountAfterRefill

Element	**Norma l**	**Error**
serviceClassTemporaryExpiryDate	O	N/A
serviceClassOriginal	O	N/A
serviceClassCurrent	M	N/A
accountFlags	O	N/A
promotionPlanID	O	N/A
serviceFeeExpiryDate	O	N/A
supervisionExpiryDate	O	N/A
creditClearanceDate	O	N/A
serviceRemovalDate	O	N/A
accountValue1	M	N/A
accountValue2	O	N/A
dedicatedAccountInformation	O	N/A
usageAccumulatorInformation	O	N/A
serviceOfferings	O	N/A
communityIdList	O	N/A
offerInformationList	O	N/A

5.22.2 accountFlags

The *accountFlags* parameters contains life cycle state flags of the account, indicating the actual status of the account. It is enclosed in a <struct> of its own.

Table44 accountflags

Element	Type	Subtype
activationStatusFlag	O	
negativeBarringStatusFlag	O	
supervisionPeriodWarningActiveFlag	O	
serviceFeePeriodWarningActiveFlag	O	
supervisionPeriodExpiryFlag	O	
serviceFeePeriodExpiryFlag	O	
twoStepActivationFlag (PC:03327)	O	

5.22.3 accountFlagsAfter and accountFlagsBefore

The *accountFlagsAfter and accountFlagsbefore* parameters contains life cycle state flags of the account, indicating the actual status of the account after and before. It is enclosed in a <struct> of its own.

Table45 accountFlagsAfter and accountFlagsBefore

Element	Type	Subtype
activationStatusFlag	O	
negativeBarringStatusFlag	O	
supervisionPeriodWarningActiveFlag	O	
serviceFeePeriodWarningActiveFlag	O	
supervisionPeriodExpiryFlag	O	
serviceFeePeriodExpiryFlag	O	

5.22.6 chargingRequestInformation

The *chargingRequestInformation* parameter contains request information for a charged end user communication event.
It is enclosed in a <struct> of its own.

TABLE 48

Element	Type	Subtype
chargingType	O	
chargingIndicator	O	
reservationCorrel ationID	O	

5.22.7 chargingResultInformation

The *chargingResultInformation* parameter contains result information for a charged end user communication event. It is enclosed in a <struct> of its own.
Note:
For GetBalanceAndDate Response the currency1 and currency2 are sent separately and are not included in chargingResultInformation.

TABLE 49

Element	Norma l	Error
cost1	O	O
currency1	O	O
cost2	O	O
currency2	O	O
chargingResultCode	O	N/A
reservationCorrelationID	O	N/A
chargingResultInformationS ervice	O	O

5.22.12 dedicatedAccountInformation

The struct *dedicatedAccountInformation* contains balances and dates for dedicated accounts. For get requests the `subDedicatedAccountInformation` parameter will only be included if specifically indicated with the `requestSubDedicatedAccountDetailsFlag`. A `dedicatedAccountInformation` struct with no value or date parameters constitutes a composite dedicated account with no assigned sub dedicated account. It is enclosed in a <struct> of its own. Structs are placed in an <array>.
Table 54 dedicatedAccountInformation
Element
Reference
Type
Subtype
dedicatedAccountID

M
dedicatedAccountValue1

Section 6.53
O
OR
dedicatedAccountValue2
Section 6.53
O
expiryDate
Section 6.57
O
startDate
Section 6.168

O
offerID
Section 6.88
O
Only included if the dedicatedAccountInformation array itself is not wrapped
by an offer structure.
dedicatedAccountRealMoneyFlag (PC:05225)
Section 6.51
O
closestExpiryDate
Section 6.36
O
AND
closestExpiryValue1
Section 6.37
(1)
OR
closestExpiryValue2
Section 6.37
(1)
closestAccessibleDate
Section 6.34
O
AND

closestAccessibleValue1
Section 6.35
(2)
OR
closestAccessibleValue2
Section 6.35
(2)
subDedicatedAccountInformation
Section 5.22.36
O
dedicatedAccountActiveValue1
Section 6.47
O
dedicatedAccountActiveValue2
Section 6.47
O
dedicatedAccountUnitType
Section 6.52
O
compositeDedicatedAccountFlag
Section 6.39
O
(1) May only be included if closestExpiryDate is included.
(2) May only be included if closestAccessibleDate is included.

5.22.13 dedicatedAccountChangeInformation

The struct *dedicatedAccountChangeInformation* contains information of changes done to balances and dates for dedicated accounts. A `dedicatedAccountChangeInformation` struct with no value or date parameters constitutes a composite dedicated account with no assigned sub dedicated account.
It is enclosed in a <struct> of its own. Structs are placed in an <array>.

Element	Type		Subtype	
Element	DA	Compo site-DA	Subtype	
dedicatedAccountID	M	M		
dedicatedAccountV alue1	O	O	OR	
dedicatedAccountV alue2	O	O	OR	
expiryDate	O	O		
startDate	O	O		
offerID	O	O		
dedicatedAccoun tRealMoneyFlag (PC:05225)	O	O		
closestExpiryDate	N/A	O		
closestExpiryValue1	N/A	O(1)	OR	
closestExpiryValue2	N/A	O(2)	OR	
closestAccessibleD ate	N/A	O		
closestAccessibleVa lue1	N/A	O(3)	OR	
closestAccessibleVa lue2	N/A	O(4)	OR	

Element	Type		Subtype
Element	DA	Composite-DA	Subtype
dedicatedAccountActiveValue1	N/A	O	OR
dedicatedAccountActiveValue2	N/A	O	OR
subDedicatedAccountChangeInformation	N/A	O	
dedicatedAccountUnitType	O	O	

5.22.16 dedicatedAccountSelection

The *dedicatedAccountSelection* parameter is used to select which dedicated accounts that will be returned. If no dedicated account IDs are specified in the request all installed dedicated accounts are returned. The request contains first and last identities for a sequence of dedicated accounts. If a single dedicated account shall be returned, dedicatedAccountIDFirst could be used alone, or the same identity could be used for both dedicatedAccountIDFirst and dedicatedAccountIDLast. Overlapping sequences is allowed, the response will only contain one instance per dedicated account. Structs are placed in an <array> with maximum 255 entries.

Table 58 dedicatedAccountSelection

Element	**Type**	**Subtype**
dedicatedAccountIDFirst	M	
dedicatedAccountIDLast	O	

5.22.17 dedicatedAccountUpdateInformation

The struct *dedicatedAccountUpdateInformation* contains information for updating balances and expiry date for dedicated accounts. When adding value to a composite dedicated account the `dedicatedAccountUpdateInformation` can be repeated several times with the same `dedicatedAccountID` to create several sub dedicated accounts. if `AllowCropOfCompositeDedicatedAccounts` is set to true, it is allowed to adjust the dates of the composite dedicated to be inside the dates of the sub dedicated accounts. The sub dedicated accounts will in this case be adjusted to match the dates of the composite dedicated account. Sub dedicated account outside the new dates will be deleted. Sub dedicated accounts only partly outside the new dates of the composite dedicated account will have its dates updated but the value will not be changed. The different types in the table represents the data to send to perform different services. ‘‘Update value and balance on a Prime-DA’’ describes the data needed for updating a non composite dedicated account, a dedicated account without sub dedicated accounts. ‘‘Add value to Composite-DA’’ describes the data needed to add value to a composite dedicated account. The value will be added to an existing sub dedicated account if the dates match an existing one, otherwise a new sub dedicated account will be created. ‘‘Change information on a Composite-DA’’ describes the data needed to update the information on composite dedicated account level and not on the sub dedicated account level It is enclosed in a <struct> of its own. Structs are placed in an <array>

Table 59 dedicatedAccountUpdateInformation

Element	**Type**			**Subty**
Element	**Update value and balance on a Prime-DA.**	**Add value to Composite-DA.**	**Change information on Composite-DA**	**Subtype**
dedicatedAccountID	M	M	M	
adjustmentAmountRelative	O	M(1)	N/A	XOR
dedicatedAccountValueNew	O	N/A	N/A	XOR
adjustmentDateRelative	O	N/A	O	XOR
expiryDate	O	O Default: DateInfinite	O	XOR
startDate	O	O Default: DateBeginningOfTime	O	XOR
adjustmentStartDateRelative	O	N/A	O	XOR
dedicatedAccountUnitType	O	O	N/A	

Element	Type			Subty
Element	**Update value and balance on a Prime-D A.**	**Add value to Composite-D A.**	**Change information on Composite-D A**	**Subtype**
expiryDateCurrent (2)	O	O	O	
startDateCurrent (3)	O	O	O	

(1) The adjustment amount can only be positive.

(2) Must be the same value as stored on the dedicated account for the operation to be successful.

(3) Must be the same value as stored on the dedicated account for the operation to be successful.

5.22.20 messageCapabilityFlag

The messageCapabilityFlag parameter indicates the possible actions

that may be performed on the account due to an operation initiated over this

protocol. It is enclosed in a <struct> of its own.

Table 61

Element	Type	Subtype
promotionNotificationFl ag	O	
firstIVRCallSetFlag	O	
accountActivationFlag	O	

5.22.22 offerInformationList

The struct `offerInformationList` contains dates for offers. In case of removal of offers it contains the dates before the offers were removed. It is enclosed in a <struct> of its own. The structs are placed in an <array>

TABLE 63

Element	Normal	Subtype
offerID	M	
startDate	O	
expiryDate	O	
offerType	O	

6 Elements

6.5 accountValue1 and accountValue2

The `accountvalue1 and accountValue2` parameters contains the account value for the subscriber's master account. This is not taking in consideration any ongoing chargeable events. 1 indicates an account value in the first currency to be announced and 2 an account value in the second one.
Data Type: <string>
Element Format: Price
Element Value Range: -999 999 999 999 to 999 999 999 999

6.14 adjustmentAmountRelative

The `adjustmentAmountRelative` parameter defines the amount of the adjustment (positive or negative) to be applied to the account.
Data Type: <string>
Element Format: Unit
Element Value Range: -9 223 372 036 854 775 807 to 9 223 372 036 854 775 807

6.19 aggregatedBalance1 and aggregatedBalance2 (PC:05225)

The `aggregatedBalance1 and aggregatedBalance2` parameters contains the aggregated balance for the subscriber. This is not taking in consideration any ongoing chargeable events.
`aggregatedBalance1` indicates an aggregated balance in the first currency to be announced and `aggregatedBalance2` the aggregated balance in the second currency. Aggregated balance is used to display the total balance of real money on the subscribers account. Real money can be seen as money added to the account by the subscriber and does not include various bonuses or promotions. Subscribers aggregated balance is the sum of main account value and the dedicated accounts marked with the `dedicatedAccountRealMoneyFlag` flag.
Data Type: <string>
Element Format: Price
Element Value Range: -999 999 999 999 to 999 999 999 999

6.20 allowCropOfCompositeDedicatedAccounts

The `allowCropOfCompositeDedicatedAccounts` parameter is used to indicate if it shall be allowed to crop a composite dedicated account
Data Type: <boolean>
Element Value Range: See Table 85

Table 85 Element Value Range – allowCropOfCompositeDedicatedAccounts

Value Range	Description
0 (false) (default value)	Do not allow crop
1 (true)	Allow crop

6.23 cellIdentifier (PC:08875)

The `cellIdentifier` parameter output the CellGlobal Identity (CGI) or Service Area Identity (SAI). The cellIdentifier can be sent as an input parameter through UCIP. If not sent as parameter, cellIdentifier is retrieved from the HLR using ATI request.
Data Type: <string>
Element Format: Alphanumeric
Element Value Range: 10 to 19

6.44 creditClearancePeriod

The `creditClearancePeriod` parameter contains the period until credit clearance.
Data Type: <int> or <i4>
Element Value Range: 0 to 1023

6.45 currency1 and currency2

The `currency1 and currency2` parameters contains the currencies to be presented to the end user. `currency1` indicates the first currency to be announced and `currency2` the second one.
Data Type: <string>
Element Format: Currency

6.61 externalData1, externalData2, externalData3 and externalData4

These parameters are used as a spare parameter for customizations to include in data records.
Data Type: <string>
Element Size: 1 to 128

6.73 languageIDCurrent

The `languageIDCurrent` parameter defines the subscriber's preferred language.
Data Type: <int> or <i4>
Element Value Range: See Table 103

Table 103 Element Value Range – languageIDCurrent

Value Range	Elements
1	Operator specific language 1
2	Operator specific language 2
3	Operator specific language 3
4	Operator specific language 4

6.78 mainAccountValueNew (PC:05163)

The `mainAccountValueNew` parameter contains a value to assign to a main account. This is not taking in consideration any ongoing chargeable events.

Data Type: <string>

Element Format: Price

Element Value Range: 0 to 999 999 999 999

6.83 negativeBalanceBarringDate

The `negativeBalalanceBarringDate` parameter contains the date when a subscriber is scheduled to be barred, or was barred, due to negative balance.
Data Type: <dateTime.iso8601>
Element Value Range: DateMin to DateMax

6.93 originHostName

The `originHostName` parameter contains an identifier string for the host where the operation originated from. The host name shall be unique within the network for a given network element type.
Data Type: <string>
Element Format: Alphanumeric
Element Size: 1 to 255

6.94 originNodeType

The `originNodeType` parameter defines the origin node type that is set by AIR or provided by an external network element. It contains the type of the logical node from where the operation originated. External network elements are not allowed to use the reserved names on network element types, (except EXT).
Data Type: <string>
Element Size: 1 to 8
Element Format: Alphanumeric
Element Value Range: See Table 110

Table 110 Element Value Range – originNodeType

Network Element type	**Description**
EXT	External system
AIR	Account information and refill
ADM	Administrative system
UGW	USSD gateway
IVR	Interactive voice response system
OGW	On-line gateway
SDP	Service data point

Note: These names on network element types, (except EXT) are not allowed for settings from an external network element

6.95 originOperatorID

The `originOperatorID` parameter is the identity of the system user or the session from where the operation was initiated. It might be used for security management or logging purposes for an example.
Data Type: <string>
Element Size: 1 to 255
Element Format: Alphanumeric

6.96 originTimeStamp

The `originTimeStamp` parameter contains the date and time of sending the request by the entity that initiated the operation.
Data Type: <dateTime.iso8601>
Element Value Range: See Table 145

6.97 originTransactionID

The `originTransactionID` parameter reference to a single operation, generated by the system it was initiated from.
Note:

Each operation must have a **unique** value
The value in the `originTransactionID` parameter must be unique per operation and can be a sequence number. An operation in this case is e.g. one refill.
Data Type: <string>
Element Format: Numeric
Element Value Range: 0 to 9999 9999 9999 9999 9999

6.123 requestFirstAccessibleAndExpiredBalanceAndDateFlag

The `requestFirstAccessibleAndExpiredBalanceAndDateFlag` parameter is used to indicate the requested detail level of the content in the `dedicatedAccountInformation`.
Data Type: <boolean>

Table 119 Element Value Range - requestFirstAccessibleAndExpiredBalanceAndDateFlag

Value	**Description**
0 (false) (default value)	The response will not include the associated closestExpiryDate, closestExpiryValue1, closestExpiryValue2, closestAccessibleDate, closestAccessibleValue1,closestAccessibleValue2.
1 (true)	The response will include the associated closestExpiryDate, closestExpiryValue1, closestExpiryValue2, closestAccessibleDate, closestAccessibleValue1,closestAccessibleValue2.

6.130 requestSubDedicatedAccountDetailsFlag

The `requestSubDedicatedAccountDetailsFlag` parameter is used to indicate the requested detail level of the content in the `dedicatedAccountInformation`.
Data Type: <boolean>
Table 126
Element Value Range - requestSubDedicatedAccountDetailsFlag

Value	Description
0 (false) (default value)	The response will not include the associated subDedicatedAccountInformation structs.
1 (true)	The response will include the associa ted subDedicatedAccountInformation structs.

6.139 serviceClassCurrent

The `serviceClassCurrent` parameter contains the service class currently used by the subscriber. This might be a temporary Service Class, which is controlled by a temporary Service Class expiry date (separate parameter).
Data Type: <int> or <i4>
Element Value Range: 0 to 9999

6.156 serviceFeeExpiryDate

The `serviceFeeExpiryDate` parameter contains the expiry date of the service fee period.
Data Type: <dateTime.iso8601>
Element Value Range: See Table 145.

6.157 serviceFeeExpiryDateCurrent

The `serviceFeeExpiryDateCurrent` parameter contains the current expiry date of the service fee period. Used for validation. No validation is performed if omitted.
Data Type: <dateTime.iso8601>
Element Value Range: DateMin to DateMax or DateInfinite

6.158 serviceFeeExpiryDateRelative

The `serviceFeeExpiryDateRelative` parameter is used to make a relative adjustment to the current service fee expiry date. The adjustment can be positive or negative. It is expressed in number of days.
Data Type: <int> or <i4>
Element Value Range: -999 to -1 and 1 to 999
Element Value Range: -32627 to -1 and 1 to 32627 (PC:02808)

6.167 serviceRemovalPeriod

The `serviceRemovalPeriod` parameter contains the period until service removal.
Data Type: <int> or <i4>
Element Value Range: 0 to 1023

6.171 subscriberNumber

The `subscriberNumber` parameter contains the subscriber identity of the subscriber related to the operation. The default format of the parameter is the same numbering format as used by the account database, this also includes support of leading zeroes. If another format is used then it must be indicated by `subscriberNumberNAI` parameter.
Data Type: <string>
Element Format: Numeric
Element Size: 1 to 28

6.172 subscriberNumberNAI

The `subscriberNumberNAI` parameter contains the Nature of Address Indicator identifies the format of the `subscriberNumber` parameter.
Data Type: <int> or <i4>
Element Value Range: See Table 135

Table 135 Element Value Range - subscriberNumberNAI

Value Range	Element
0	Unknown
1	International number
2	National significant number
3	Network specific number
4	Subscriber number
5	Reserved
6	Abbreviated Number
7	Reserved for extension

6.175 supervisionExpiryDate

The `supervisionExpiryDate` parameter contains the expiry date of the supervision period.
Data Type: <dateTime.iso8601>
Element Value Range: See Table 145

6.176 supervisionExpiryDateCurrent

The `supervisionExpiryDateCurrent` parameter contains the current expiry date of the supervision period. Used for validation. No validation is performed if omitted.
Data Type: <dateTime.iso8601>
Element Value Range: DateMin to DateMax or DateInfinite

6.177 supervisionExpiryDateRelative

The `supervisionExpiryDateRelative` parameter is used to make a relative adjustment to the current supervision expiry date. The adjustment can be positive or negative. It is expressed in number of days.
Data Type: <int> or <i4>
Element Value Range: -999 to -1 and 1 to 999
Element Value Range: -32627 to -1 and 1 to 32627 (PC:02808)
Note:
A value of zero (0) will be rejected with a fault code 100.

6.181 temporaryBlockedFlag

The `temporaryBlockedFlag` parameter is a flag indicating whether the subscriber and operator has access to subscriber and account data. A temporary blocked subscriber does not have access to the account, and any messages requesting changes in the account or subscriber data are rejected. Therefore in case temporary blocked is set, a temporary blocked indication is set in the response code of the response message and the action is rejected. It is always possible to read data, but the only way to change data again is to unblock the subscriber. In case temporary blocked flag is not returned in the response messages, then the subscriber is not blocked.
Data Type:<boolean>
Element Value Range: See Table 138

Table 138 Element Value Range – temporaryBlockedFlag

Value Range	Description
0 (false) (default value)	Unblocked
1 (true)	Temporary blocked

6.185 transactionCode

The `transactionCode` parameter is used to specify the operation in more detail. It will associate a description of a certain operation along with `transactionType`. Reserved values must not be used when MINSAT is used as Administrative System.

Table 139 Reserved values to not be used when using MINSAT as Administrative System.

Value
SCC
FAF
CBE
ADJ
TC
TV
REBATE
DEB
DEDUCTIONGSM
DEDUCTIONPERIODIC

6.186 transactionCurrency

The `transactionCurrency` parameter contains an ID to point out what currency is used for the transaction. A transaction parameter includes data regarding a requested change.
Data Type: <string>
Element Format: Currency

6.187 transactionType

The `transactionType` parameter is used to specify the operation in more detail. It will associate a description of a certain operation along with `transactionCode`. Reserved values must not be used when MINSAT is used as Administrative System.

Table 140 Reserved values to not be used when using MINSAT as Administrative System.

VALUE
EUC
PIN
TT
GSM

XML-RPC Data Type: <string>
Element Format: Alphanumeric
Element Size: 1 to 30

7 Data Type Definitions

This section covers data type definitions.

7.1 Data Types

XML-RPC data types are used to transfer UCIP messages. If a parameter on element level does only support a specific size, range or format of a given data type, then this is further described in Section 6 on page 79.
The data types that are used in UCIP are listed and described in the table Table 145.

Table 145 Data type definition

Data Type	Definition
<array>	A collection of elements. In this context used as a collection of structures (<struct>).
<boolean>	A boolean value. Value range: · 0 (false) · 1 (true)

Data Type	**Definition**
<dateTime.iso8601>	A date and time. An element of this type only contains date, unless it is specified to also include time. The used format does not strictly follow the XML-RPC specification on date format, as the UCIP protocol allows timezone information to be included in the data type. It does however follow the ISO 8601 specification. Parsers for this protocol must be able to parse dates containing timezone. Format: yyyyMMddThh:mm:ssTZ Value Range: · yyyy = 1970 to 9999 · MM = 01 to 12 · dd = 01 to 31 · hh: = 00 to 24 · mm: = 00 to 59 · ss: = 00 to 59 · TZ = [+/-]0000 to [+/-]2359 TZ is the timezone deviation in hours and minutes from the UTC. TZ is always preceded by either a " + " or " - " sign. In the case when only a date is transferred, the time must be set at 12:00:00, and the and TZ must be set at +0000 (UTC). Example 1: 19980817T14:08:59+ 0130 (timestamp) Example 2: 19980817T12:00:00+ 0000 (date only)
<int>or<i4>	A signed, 32-bit integer.

).

Data Type	Definition
<string>	An ASCII string, which may contain zero bytes. If no type is indicated, the default type is string. (1)
<struct>	A collection of key-value pairs. The keys are strings; the values may be of any type.

(1) This protocol expects the ASCII strings to be coded in 7-bit USASCII (ISO 646

7.2 Element formats

Element formats are specific value ranges used for the elements. The logical data types that are used in UCIP are listed and described in the table Table 146

Table 146 Element formats

Data Type	Definition	Data Type
Alphanumeric	Alphanumeric characters are those comprised by the combined set of the 26 alphabetic characters, A to Z, and the Arabic numerals, 0 to 9. The alphanumeric characters set may include both upper and lower case letters as well as space (“ ”). The alphanumeric data type is a string consisting of alphanumeric characters.	<string>
Currency	Characters according to the ISO 4217 standard, see Codes for the representation of currencies and funds, Reference [4] Example: `EUR` for Euro and `SEK` for Swedish Krona.	<string>
Numeric	The numeric format consists of the Arabic numerals 0 to 9.	<string>
SignedNumeri c	The SignedNumeric format is a Numeric value plus the minus sign “-” in case of a negative value.	<string>

Data Type	Definition	Data Type
Price	The Price is a numeric value plus the minus sign “ - ” in case of a negative value. The value is provided in the lowest denomination of the currency.	<string>
Unit	The data type contains the Unit of the dedicated account values and is a numeric value plus the minus sign “ - ” in case of a negative value. Unit can represent : Time, money, service specific units and volume. The value is provided in the lowest denomination of the currency.	<string>

7.3 Logical Values

Logical Values are used in the Elements value ranges to define specific values for the elements.

Table 147 Logical Values

Logical value	Description	Data Type	Value
DateBeginningOf Time	For start dates this means always available.	<dateTime.iso8601>	00000101T12:00:00+0000
DateInfinite	The infinite date. For expiry dates this means no expiry.	<dateTime.iso8601>	99991231T12:00:00+0000
DateMax	The maximum valid date (except DateInfinite)	<dateTime.iso8601>	20380119T12:00:00+0000
DateMin	The oldest possible date.	<dateTime.iso8601>	19700101T12:00:00+0000
DateToday	The date of the day when the request is sent.	<dateTime.iso8601>	-
DateTomorrow	The date of the day after the request is sent.	<dateTime.iso8601>	-

8 State Behavior

This section covers state behavior.

8.1 Introduction

The messages on the UCIP does not use any complex state machines, as only single request-response dialogues are handled. XML-RPC syntax is used in the following section.

8.2 Traffic Cases

Examples of UCIP messages used to model potential AIR usage scenarios are to be found in Section 9 on page 151. Additional HTTP information is not presented in the examples. For additional information on the transport layer, see Section 10 on page 159.

10 Compatibility Mechanisms

A client must use the `user-agent` HTTP field to indicate client identity, protocol version and client version in such a way that a server can provide different answers depending on the client and client version, if the server deems it necessary.
According to the HTTP specification, the `User-Agent` field may contain multiple product tokens and comments identifying the client. By convention, the product tokens are listed in the order of their significance for identifying the application. The token must be organized as follows ‘‘User-Agent: client-name/protocol version/client version’’
The following data is to be used:

Table 148 User agent data

Parameter	**Description and value**
client name	The client name are decided by client. Not used by server logic but useful for fault tracing.
protocol version	The protocol version must be 4.1
client version	The client version are decided by client. Not used by server logic but useful for fault tracing.

An example is: User-Agent: IVR/4.0/1.0
If the server cannot handle the protocol version, it will return HTTP status code 403 (Forbidden).

11 Transfer Mechanism

This section cover transfer mechanism.

11.1 Introduction

The transfer mechanism used is HTTP, both HTTP requests and HTTP responses use headers to send information about the HTTP message. A HTTP header is a series of lines, with each line contains a name followed by a colon and a space, and then a value. The fields can be arranged in any order. Some header fields are used in both request and response headers, while others are appropriate only for either a request or a response. Many request header fields will allow the client to specify several acceptable options in the value part and, in some cases, even rank each option's preference. Multiple items are separated using a comma. Some fields can occur more than once in a single header. For example, a header can have multiple ‘‘Warning’’ fields.
Example:
A client could send a request header that includes ‘‘Content-Encoding: gzip, compress’’ indicating it would accept either type of compression. If the server uses gzip encoding for the response body, its response header would include ‘‘Content-Encoding: gzip’’.
The syntax used in UCIP is described in Section 4 on page 9.
For more information about the transfer mechanism used by HTTP, see Hypertext Transfer Protocol - - HTTP/1.1, Reference [6].

Example:

```
[ POST BEGIN ]
POST /Air HTTP/1.1
Content-Length: 286
Content-Type: text/xml
Date: Mon, 30 Aug 2004 13:17:39 MEST
Host: ws2258:10010
User-Agent: IVR/4.1/1.0
Authorization: Basic dXNlcjpwYXNzd29yZA==
[ XMLRPC REQUEST GOES HERE ]
[ POSTEND ]
```

11.2 Request Mechanism

All XML-RPC requests should be submitted using the HTTP `POST` command.
The post includes a Uniform Resource Identifier (URI) indicating which responder that should handle the request. The content of this URI is not defined, it is up to the implementing server to decide which URIs that are to be handled.
Use correct content-length and content-type. Provide the content-length in the request and set the content-type to ‘‘text/xml’’. Include the calling host in the HTTP header and use the User-agent to identify the calling service.

Table 149 URI responder

Parameter	Description and value
URI	For all messages in this document the URI must be set to `Air`.

11.3 Response Mechanism

Provide correct content-length in the response and set content-type to ‘‘text/xml’’.
Use the ‘‘server’’ field to identify the responding server.

11.4 Error Handling

A total application failure such as a server crash or an unhandled server exception returns a HTTP response code in the 500 series (Internal Server Error). If the client uses an unsupported version of the protocol, HTTP response code 403 is returned. If the XML-RPC request is incomplete, uses illegal data types or has other structural problems, the XML-RPC fault construct must be used to indicate the Failure to the client. HTTP response code 200 (successful) must be used for this kind of situation. Complete XML-RPC requests uses the response code defined in the response messages to indicate success or failure. Failures like ‘‘Temporary Blocked’’, should not be considered as faults and therefore must not use the XML-RPC fault construct. HTTP response code 200 must be used. Requests sent to the AIR server that are rejected due to overload are responded with XML-RPC fault response with faultCode = 1007 In case of a mismatch of user:password the response will be “401 Unauthorized WWW-Authenticate: Basic realm=”/Air”.

In case of incorrect content length, the response will be ‘‘400 Incomplete request’’.

If an user is not allowed to do the specific action the response will be “403 Forbidden: Insufficient user privileges" (header version 1.1).

11.5 Load Balancing and Fail Over

The client may at its own discretion balance the load over several AIR servers within the same AIR system. In case an AIR server is unavailable or overloaded, it is the responsibility of the client to fail-over to another AIR server within the AIR system.
The following are indications of that an AIR server is unavailable or overloaded:
·
·
·
Not possible to acquire a new connection to the AIR server Requests sent to the AIR server time out
Request sent to the AIR server are rejected due to overload (XML-RPC fault response with faultCode = 1007)
The client may later, at its own discretion, attempt to contact the AIR server again.

11.6 Authentication, Authorization and Security

Authentication (Authorization) is mandatory and it is done at HTTP level. All security is intended to be handled at transport level and downwards. For higher security an encryption scheme like IPSec is recommended.

Using information from the HTTP header the server will handle the authentication in a two step process.
1. The HTTP header must contain an Authentication field containing “Basic user:password”. The user:password is b64 encoded, in other words it is not sent in readable text. The user:password is compared to a user:password configured in the AIR. In case of a mismatch of user:password the response will be “401
Unauthorized WWW-Authenticate: Basic realm=”/Air”.
2. The User (as in bullet 1 above) can be configured towards a list of allowed actions, configured in the AIR application. This means that one user can be blocked for installing subscribers but is allowed to change account data. Another user might only have access to read the accounts.
In case of that the user is not allowed to perform an action the response will be “403 Forbidden: Insufficient user privileges (header version 1.1)."

11.7 TCP Level Compliance

To secure the server performance and robustness, an RST will always be sent when the server closes the connection. This procedure differs from the standard TCP FIN/ACK teardown

12 Protocol Format Changes

This section describes changes in different versions.

12.1 Version 4.1

This version contains the modifications described in Table 150

Table 150 Protocol Format Changes in Version 4.1

Message	**Protocol Message Parameter**	**Changes compared version 4.0**	**Comment**
dedicatedAccountInformation	dedicatedAccountUnitType	Removed PC tag and changed name of parameter	
dedicatedAccountRefillInformation	dedicatedAccountUnitType	Removed PC tag and changed name of parameter	
dedicatedAccountUpdateInformation	dedicatedAccountUnitType	Removed PC tag and changed name of parameter	
subDedicatedAccountUpdateInformation	dedicatedAccountUnitType	New parameter	
GetOffers	offerSelection	Changed error code returned for explicit request from 214 to 165,to indicate correct system behaviour.	

12.2 Version 4.0

This version contains the modifications described in Table 151.

Message	Protocol Message Parameter	Changes compared version 3.5	Comment
Refill(response)	startDate	Removal of pc tags and moved to standard.	
GetBalanceAndDate(Response)	startDate	Removal of pc tags and moved to standard.	
GetBalanceAndDate (Response) GetOffers (Response)	closestAccessibleDate, closestAccessibleValue1, closestAccessibleValue2, closestExpiryDate, closestExpiryValue1 and closest ExpiryValue2	Added	
GetBalanceAndDate (Response) GetOffers (Response)	subDedicatedAccountInformation.dedicatedAccountID	Removed	
UpdateBalanceAndDate(Request)	startDate and adjustmentStartDateRelative	Removal of pc tags and moved to standard.	
UpdateBalanceAndDate(Response)	startDate and Response codes 163,164	Removal of pc tags and moved to standard.	
UpdateBalanceAndDate (Response)	dedicatedAccountInformation	Replaced by dedicatedAccount ChangeInformation	
UpdateBalanceAndDate (Response)	dedicatedAccountChangeInformation	Replaces dedicated AccountInformation	
UpdateBalanceAndDate (Response)	compositeDedicatedAccountInformation	Removed.	
UpdateBalanceAndDate (request)	allowCropOfCompositeDedicatedAccounts	New parameter	

Message	Protocol Message Parameter	Changes compared version 3.5	Comment
GetAccountManagementCounters (ID:01871)	-	New message	
GetAccumulators (response)	accountFlagsAfter and accountFlagsBefore	New parameter	
UpdateBalanceAndDate (request)	serviceFeeExpiryDateCurrent	New parameter	
UpdateBalanceAndDate (request)	supervisionExpiryDateCurrent	New parameter	
UpdateBalanceAndDate (request)	serviceClassCurrent	New parameter	
UpdateBalanceAndDate (request)	expiryDateCurrent	New parameter	
UpdateBalanceAndDate (request)	expiryDateCurrent	New parameter	
UpdateBalanceAndDate (response)	negativeBalanceBarringDate	New parameter	
UpdateBalanceAndDate (response)	accountFlagsAfter	New parameter	
UpdateBalanceAndDate (response)	responseCode	204 added	
UpdateBalanceAndDate (response)	dedicatedAccountInformation	New parameter	
UpdateBalanceAndDate (response)	accountValue1	New parameter	
UpdateBalanceAndDate (response)	accountValue2	New parameter	

Message	Protocol Message Parameter	Changes compared version 3.5	Comment
GetAccountDetails (response)	maxServiceFeePeriod	New parameter	
GetAccountDetails (response)	maxSupervisionPeriod	New parameter	
GetAccountDetails (response)	negativeBalanceBarringDate	New parameter	
GetAccountDetails (response)	accountFlagsBefore	New parameter	
GetRefillOptions (response)	accountFlagsAfter	New parameter	
GetRefillOptions (response)	accountFlagsBefore	New parameter	
UpdateAccountDetails (request)	languageIDCurrent	New parameter	
UpdateAccountDetails (response)	accountFlagsAfter	New parameter	
UpdateAccountDetails (response)	accountFlagsBefore	New parameter	
UpdateAccountDetails (response)	responseCode	204 added	
UpdateServiceClass (response)	accountFlagsAfter	New parameter	
UpdateServiceClass (response)	accountFlagsBefore	New parameter	
UpdatePromotionPlan	responseCode	204 added for Action Set and Action Add	
UpdateOffer	offerType	New parameter	

Message	Protocol Message Parameter	Changes compared version 3.5	Comment
UpdateOffer	responseCode	Added response code 214	
UpdateOffer	responseCode	Added response code 215	
GetOffers	offerSelection.offerType	New parameter	
GetOffers	responseCode	Added response code 214	
Refill(request)	requestSubDedicatedAccountDetailsFlag	Added	
Refill(response)	subDedicatedAccountRefillInformation	Added in dedicatedAccountRefillInformation in refillValueTotal and refillValuePromotion in refillInformation.l	
Refill(response)	closestExpiryDate	Added in dedicated AccountInformation in accountAfterRefill and accountBeforeRefill.	
Refill(response)	closestExpiryValue1	Added in dedicated AccountInformation in accountAfterRefill and accountBeforeRefill.	
Refill(response)	closestExpiryValue2	Added in dedicated AccountInformation in accountAfterRefill and accountBeforeRefill.	
Refill(response)	closestAccessibleDate	Added in dedicated AccountInformation in accountAfterRefill and accountBeforeRefill.	
Refill(response)	closestAccessibleValue1	Added in dedicated AccountInformation in accountAfterRefill and accountBeforeRefill.	

Message	**Protocol Message Parameter**	**Changes compared version 3.5**	**Comment**
Refill(response)	closestAccessibleValue2	Added in dedicated AccountInformation in accountAfterRefill and accountBeforeRefill.	
Refill(response)	subDedicatedAccountInformation	Added in dedicated AccountInformation in accountAfterRefill and accountBeforeRefill.	
Refill(response)	dedicatedAccountActiveValue1	Added in dedicated AccountInformation in accountAfterRefill and accountBeforeRefill.	
Refill(response)	dedicatedAccountActiveValue2	Added in dedicated AccountInformation in accountAfterRefill and accountBeforeRefill.	
DeleteOffer	-	PC tag removed.	
GetAccountDetails	offerInformationList	New parameter added.	
GetBalanceAndDate	offerInformationList	New parameter added.	
GetOffers	-	PC tag removed.	
UpdateOffer	-	PC tag removed	

12.3 Version 3.5

This version contains the modifications described in Table 152 .

Table 152 Protocol Format Changes in Version 3.5

Message	**Protocol Message Parameter**	**Changes compared version 3.4**	**Comment**
GetAccountDetails (request)	requestPamInformationFlag	New parameter	
GetAccountDetails (response)	pamInformationList	New parameter	

13 Commands Sample for Vendor

Sample Header

```
POST /Air HTTP/1.1
Host: 10.13.32.214
Content-Type: text/xml
User-Agent: UGw Server/4.1/1.0
Accept: *
Authorization: Basic ZmRzdXNlcjpFcmljc3NvbiE=
Content-Length: 1374                    ⍰ Set accordingly
```

Sample Update Balance And Date Command

```
<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
<methodName>UpdateBalanceAndDate</methodName>
<params>
<param>
<value>
<struct>
<member>
<name>originNodeType</name>
<value><string>EXT</string></value>
</member>
<member>
<name>originHostName</name>
<value><string>M2M</string></value>
</member>
<member>
<name>originTransactionID</name>
<value><string>00110713122128000</string></value>
</member>
<member>
<name>originTimeStamp</name>
<value><dateTime.iso8601>20110713T12:21:32+0100</dateTime.iso8601></value>
</member>
<member>
<name>subscriberNumberNAI</name>
<value><int>2</int></value>
</member>
<member>
<name>subscriberNumber</name>
<value><string>3085259265</string></value>
</member>
<member>
<name>transactionCurrency</name>
<value><string>PKR</string></value>
</member>
<member>
<name>adjustmentAmountRelative</name>
<value><string>10</string></value>
</member>
<member>
<name>externalData2</name>
<value><string>M2M_Transfer</string></value>
</member>
<member>
```

```
<name>externalData1</name>
<value><string>USSD</string></value>
</member>
<member>
<name>transactionType</name>
<value><string>M2M</string></value>
</member>
<member>
<name>transactionCode</name>
<value><string>M2MTransferFromA</string></value>
</member>
</struct>
</value>
</param>
</params>
</methodCall>
```

Sample Get Balance And Date Command

```
POST /Air HTTP/1.1
User-Agent: UGw Server/3.0/1.0
Content-type: text/xml
charset=ISO-8859-1
Connection: keep-alive
Host: 10.13.32.204:10010
Content-Length: 978
Authorization:Basic ZmRzdXNlcjpmZHN1c2Vy

<?xml version="1.0"?>
<methodCall>
<methodName>GetBalanceAndDate</methodName>
<params>
<param>
<value>
<struct>
<member>
<name>originTransactionID</name>
<value>
<string>240006</string>
</value>
</member>
<member>
<name>originNodeType</name>
<value>
<string>EXT</string>
</value>
```

```
</member>
<member>
<name>originHostName</name><value>
<string>1</string>
</value>
</member>
<member>
<name>originTimeStamp</name>
<value>
<dateTime.iso8601>20060113T22:28:54+0000</dateTime.iso8601>
</value>
</member>
<member>
<name>subscriberNumber</name>
<value>
<string>923085259223</string>
</value>
</member>
</struct>
</value>
</param>
</params>
</methodCall>
```

Note Please set all the sample commands accordingly.

←--→