OpenScape Business

Web Services Interface Manual

WSI Reference

Version 2.1

Table of Contents

Table of History

Date	Version	Changes
2014-01-23	1.0	Document released for OSBiz V1R2
2014-01-28	1.1	Added description for gsSession, HookSubscribe, HookUnsubscribe
2014-05-20	1.2	Added description for AttendedTransfer, descriptions for SetForwarding and TransferCall corrected
2014-06-24	1.3	Update for OpenScape Business V1R3; Added MkCall
2015-06-15	2.0	Update for OpenScape Business V2
2015-09-15	2.01	Update of chapter 2.3
2015-09-15	2.02	Update of chapter 10
2019-10-29	2.1	Updated descriptions for EventStart, PhBookSearch, PhBookDetail, JournalRead Added descriptions for PhBookLookup, Journal Fetch, ConsultationCall, ReconnectCall, ConferenceCall, EndConference

1. About

This document addresses developers, who want to integrate Unified Communication functions of OpenScape Business into their applications.

The document describes mainly the functions of OpenScape Business, which are available for integration into 3rd party applications. Coding examples are included as well as descriptions of the required OpenScape Business IT-environment and prerequisites for interface usage.

Note: The content of this document refers to OpenScape Business V2R7

1.1. Technical Support

Technical Support in case of questions regarding the Web Services Interface can be obtained via Unify Technology Partner Program. More information is available within the following link:

http://www.unify.com/us/partners/technology-partner.aspx

1.2. Further information about OpenScape Business

General information and specific documentation about OpenScape Business is available within the Unify Partner Portal, which requires a login as registered partner of Unify:

http://www.unify.com/us/partners/partner-portal.aspx

An overview about OpenScape Business in general and description some specific topics is given within the Unify Experts Wiki within the Internet.

http://wiki.unify.com/wiki/OpenScape_Business

Please note:

The Expert Wiki may not contain latest and complete information in every case.

1.2.1. Installation-, Administration- and User Manuals

Technical documentation and user manuals of the OpenScape Business system can be downloaded either from e-docu server within the Unify Partner Portal or via the Administration Portal of the OpenScape Business system itself.

2. Web Service Interface Overview

The Web Services interface (WSI) provides functions for Unified Communication (UC) users of OpenScape Business. It enables external applications to monitor and control devices of the UC users as well as to get information about their presence status, journal and phone book entries.

2.1. Functions

The WSI supports clearly structured functions for:

- Call control
- Device control
- Monitoring / events
- Unified Communications
  - Phonebook access
  - User Journal access
  - User Presence Status access

2.2. Architecture

The Web Services Interface is a part of the UC application (UC Smart or UC Suite) of OpenScape Business.

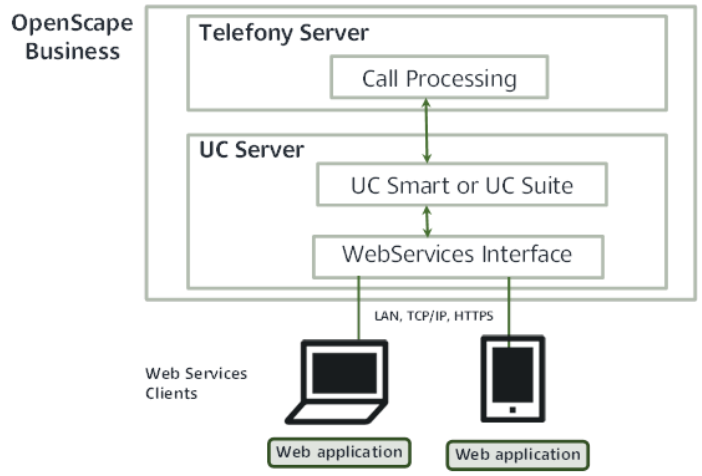

Figure 1 Schema of the interworking between all components of OpenScape Business

2.3. General access to the Web Services Interface

Interface access is done via Ethernet LAN with TCP / IP and HTTPS (HTTP optionally). Depending on the OpenScape Business hardware model, the connection has to be done either to the base system, UC Booster Card or UC Booster Server.

2.3.1. OpenScape Business with UC Smart application

The UC Smart application can be used within all models of OpenScape Business (incl. OpenScape Business X1, OpenScape Business X3, X5 or X8 with or without a UC Booster card / server and OpenScape Business S).

The connection to the Web Services Interface is done via

- IP address of the base system in OpenScape Business X systems without Booster card / server,
- IP address of the Booster card / server if connected, or
- server IP address in case of OSBiz server

The default port for WSI is TCP/8802 for HTTPS, TCP/8801 for HTTP can be optionally enabled via Admin Portal - UC Smart Assistant.

2.3.2. OpenScape Business with UC Suite application

The UC Suite application requires either a UC Booster Card or a UC Business Server in combination with OpenScape Business X3, X5 or X8 systems. In the case of OpenScape Business S no additional UC-Booster HW is required.

WSI is available via the IP address of the UC Suite server. The default port is TCP/8802 for HTTPS, TCP/8801 for HTTP can be optionally enabled via Admin Portal - Security Settings - Web Security.

2.3.3. Configuration examples

2.3.3.1. OpenScape Business X without UC Booster Card or Booster Server

The connection to the Web Services Interface is done via the IP address of the OpenScape Business main board. This option is available only with UC Smart.

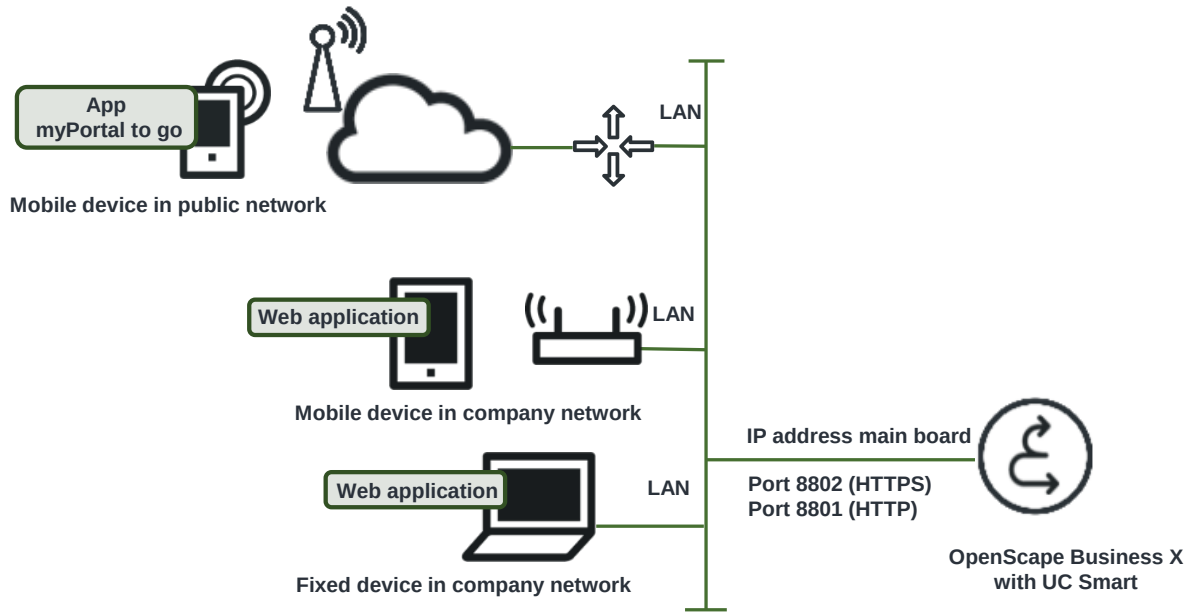

Figure 2 WSI connection to OpenScape Business X main board with UC Smart

2.3.3.2. OpenScape Business X with UC Booster Card

The connection to the Web Services Interface is done via the IP address of the UC-Booster Card.

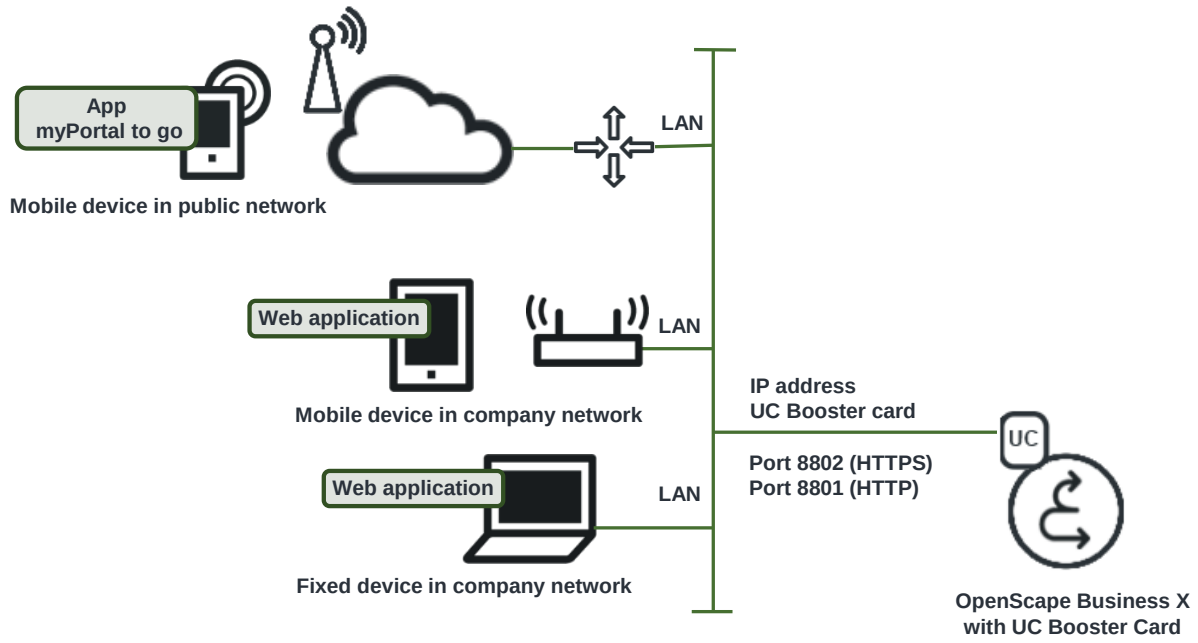

Figure 3 WSI connection to OpenScape Business X with UC Smart or UC Suite on UC Booster Card

2.3.3.3. OpenScape Business X with UC Booster Server

In large deployments the UC Booster Server is connected to the OpenScape Business system. In this case the connection to the Web Services Interface is done via the LAN Interface of the UC Booster Server.

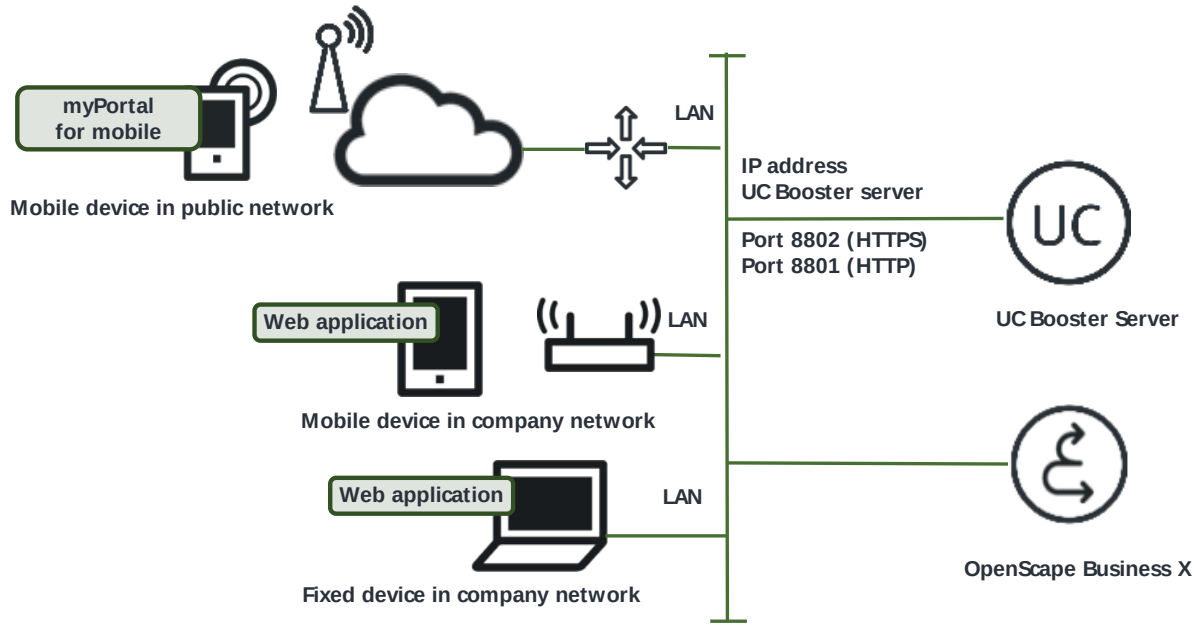

Figure 4 WSI connection to OpenScape Business X with UC Smart or UC Suite on UC Booster Server

2.3.3.4. OpenScape Business S

Since V2, OpenScape Business S supports both UC Smart and UC Suite applications. No additional UC Booster HW is required. The connection to the Web Services Interface is done via the IP address of the Server.

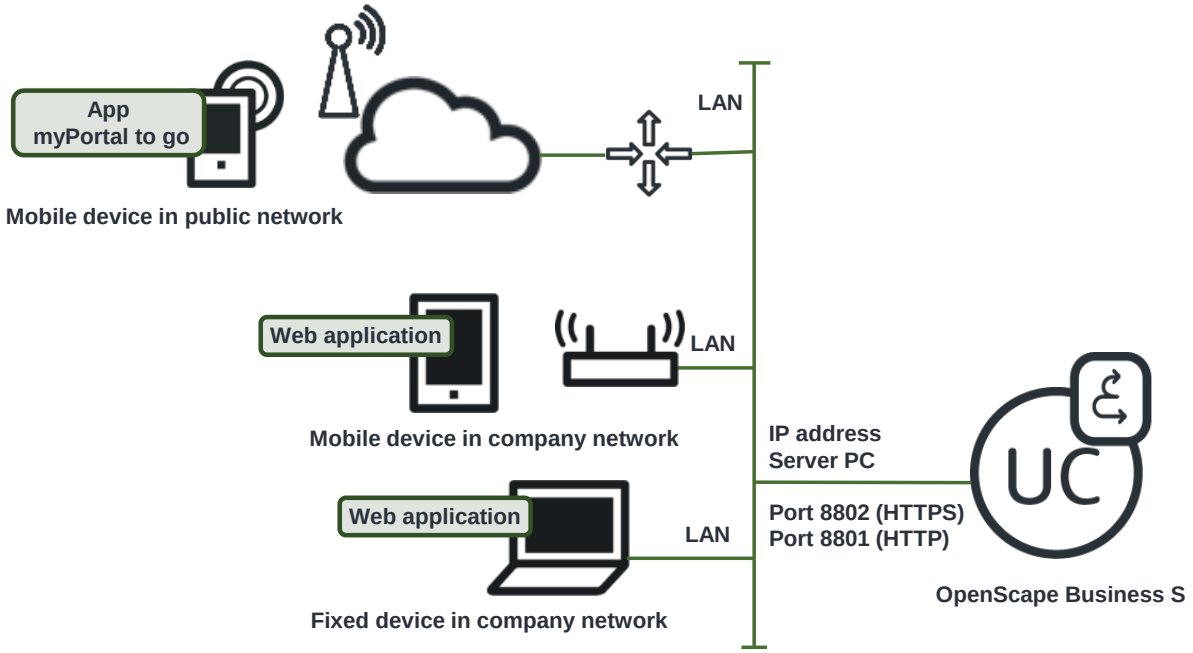

Figure 5 WSI connection to OpenScape Business S

2.4. Security Aspects

2.4.1. System Interface enabling

In order to prevent abuse, Web Services Interface is disabled (UC Smart) or restricted (UC Suite) within the factory default settings. It has to be enabled explicitly within the system administration.

Please note:

Some other applications of OpenScape Business like UC Smart clients or Mobility clients also use the Web Services Interface, therefore the interface may be already enabled within productive systems.

2.4.2. Use of HTTPS

Use of HTTPS instead of HTTP is mandatory in case that clients access the system via public Internet.

Use of HTTPS is also recommended for clients which are connected within a LAN, to keep privacy. Please note that OpenScape Business systems support uploading of trusted SSL certificates for use with the embedded web server, which results in improved HTTPS security.

Security hint:

HTTP should only be used for development purpose or in case that other means are activated within the network to ensure privacy and system consistency (VPN, firewalls etc.).

2.4.3. User Login

An application has to login into the WSI with the same login credentials as the UC myPortal user. Login credentials can be modified by the UC user via the appropriate clients or – in UC Smart mode – via UC Smart Assistant.

Whenever a login was successful, the server generates a session ID and sends it to the client in the Login response. This session ID has to be sent to the server with every command during the session.

2.5. Prerequisites

2.5.1. Licensing

The Web Services Interface itself is not licensed, but a UC Smart or UC Suite User license is required for every user who accesses the interface.

2.5.2. Activation of the Web Services Interface

In UC Smart mode, UC Smart has to be activated via Admin Portal – UC Smart Assistant. HTTP access is disabled by default.

In UC Suite mode, the access to the Web Services interface has to be configured via Admin Portal – Web Security. HTTP access is disabled by default.

2.6. Supported Devices

In general all devices, can be used, which are released for use with the UC Smart or UC Suite

Please note:

All commands, that require control of a phone, like the "MakeCall" command, are restricted to the following devices:

- DeskPhone CP HFA
- DeskPhone CP T
- DeskPhone IP HFA
- OpenStage HFA
- OpenStage T
- OptiPoint 500
- OptiPoint 410 / 420 HFA V5R7 or higher

2.7. Capacities

Depending on the OpenScape Business Hardware model different capacity values exist.

2.7.1. Number of users and client sessions

Web Services Interface based applications require a web session per logged in user and client application. The maximum number of available sessions of the web server depends on OpenScape Business system capacities in general.

System	Max. number of users	Max. number of client sessions[1]
OpenScape Business X3/X5/X8 V2 in UC Smart mode	50	100
OpenScape Business X3/X5/X8 V2 with Booster Card	150	300
OpenScape Business X3/X5/X8 V2 with Booster Server	250	500
OpenScape Business S V2	250	500

[1] Careful WSI client design according to the design / performance hints in this document is required to achieve these limits. Limits may be lower if other applications are used in parallel (e.g. CallCenter or other CSTA applications).

Please Note:

The available number of client sessions is shared by all applications, which use the Web Services Interface. As there currently are:

- myPortal to go
- myPortal Smart
- myPortal @work
- myPortal for OpenStage
- TAPI 120 in WSI / UC Smart mode
- Application Launcher
- myContacts
- OpenScape Business Attendant
- OpenScape Business BLF
- 3rd party application using the Web Services Interface

2.7.2. Device monitoring (“internal” monitor points)

The device monitoring mechanism of the Web Services Interface is independent of the standard CSTA monitor points. In case that multiple applications uses the Web Services Interface and control the same UC user only one “internal Monitor Point” is used by the UC Server. No dedicated resource management for monitor points is needed for applications utilizing the Web Services interface.

2.8. Functional Boundaries

Following boundaries have to be considered, when using the Web Services Interface:

1. Only users of the embedded Unified Communications solutions **UC Smart** or **UC Suite** can be addressed. Non UC users are not supported by the interface.
2. Web Services Interface is limited to the single systems. Within networks of OpenScape Business systems WSI has a single system view, even if application is connected to the Master Node.
3. Web Service Interface is “User centric”. Only one UC- User can be accessed via one Web Session.
4. Some functions are restricted to UC Smart only or UC Suite only. Such restrictions are documented per function in the following chapters.
5. The Web Services Interface is available also on embedded OpenScape Business platforms with limited hardware resources (CPU, RAM, …). A careful WSI client implementation is needed to maintain good overall system performance:
   - In many cases, the corresponding events already contain the changed information. In this case, the new data should be read from the event. Otherwise, only the entity changed should be read from the server instead of mass data queries.
   - Avoid data polling. The event mechanism should be used instead.
   - If information is often used in the client, a client-side caching mechanism should be used instead of reading the same information from the server again and again.
   - Phonebook queries can have high performance impact and should only be made on manual user request - not automatically / periodically.

3. General description of implemented functions

All provided functions are grouped in the following categories

- Call control
- Device control
- Monitoring and events
- UC functions (presence, phonebooks, etc.)

Every section consists of a description of the parameters, an https-request example, a short description how the example works and tables containing the response messages that are sent by the Web Services after a request was received and/or has been fulfilled.

In order to send any command, the client program or web page has to be successfully authenticated towards the web Services via login and password. It is necessary that the logged-in user has sufficient access rights within the class of service of the system to control certain device features.

3.1. CGI Syntax

The Web Services Interface (WSI) is accessible via https- or http-requests with CGI syntax. Requests send to the Web Services are composed of an address part and a command part.

Address part

https://<WebServices-IP-address>:8802/cgi-bin/ or http://<WebServices-IP-address>:8801/cgi-bin/

Note: For security reasons, it is **strongly recommended** to use https.

Command part

gadgetapi?cmd=[CommandName]&[ParameterName]=[ParameterValue]&[ParameterName]=[Param eterValue]&...

Simple example in pure html environment:

```
<html>
<body>
<button onclick="request('https://[Web Services IP address]:8802/cgi- bin/gadgetapi?
cmd=MakeCall&callingDevice=103&calledDirectoryNumber=101 ')"> Send Request
</button>
</body>
</html>
```

General Note: Dependent on the client environment, there may be restrictions regarding the “same origin policy”. For example, some older web browsers have such restrictions, so the given examples in this document will not work.

3.2. Parameter Types

Parameter	Type	Syntax	Description
<name>	**command**	ANSI text string	The command to call a certain function.
<name>	**deviceID**	internal number use of [0-9]	The device which will be controlled. Device ID refers always to the configured phone number of the appropriate device.
<name>	**Boolean**	“true” or “false”	In order to enable (true) or disable (false) a function.
<name>	**eventtype**	JSON String	Used for filter parameters

Table 1 The CGI parameter values can have a different type and syntax.

3.3. Call Number handling

All phone or extension numbers used in the Web Services depends on their field of usage. The numbers have to be entered with in following format:

- Internal number: <XX>or<XXX> or <XXXX> or <XXXXX>
- External number: +<country code><regional code><target number>

Example:

- Internal number: 101
- External number: +4989123456789

It is strongly recommended to use only the canonical format for any external phone numbers.

3.4. Using the examples

At every section you can find simple examples. If you want to execute the html request the following conditions have to be fulfilled in order to get the examples to work:

- Be sure using the right protocol https (Port 8802) or http (Port 8801). For security reasons, https usage is recommended.
- Be sure using correct network and device configuration (IP-address and device numbers).
- For these simple examples a valid login towards the Web Services is absolute necessary, before using them.
- For controlling a certain device, it is absolute necessary that the logged in user has the right to control it.

4. Login and Logout

4.1. Login

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

The login is mandatory to authenticate the user.

Format: cmd=Login&gsUser=100&gsPass=aBc12345

Parameter	Type	Description
gsUser	**deviceID**	the user ID
gsPass	**deviceID**	the user's password.

Return Values:

Positive:

```
<LOGIN>
  <ID>sessionID</ID>
  <CNT>count</CNT>
</LOGIN>
```

sessionID identifies the session for all further requests. Dependent on the client, the sessionID may also be stored on the client side within a cookie. With every following command of the session, the sessionID has to be sent – either implicitly with the cookie, or explicitly via command line parameter: "gsSession=sessionID". As the cookie support will be dropped in future product releases, it is **strongly recommended** to send the sessionID explicitly with every command via **gsSession** parameter.

The "count" shows the number of sessions for a user.

Negative (if the login fails):

```
<LOGIN>
  <ID>0</ID>
  <ERROR>reason</ERROR>
</LOGIN>
```

Important: Use the <ID> field to determine if the login was successful. If the ID=0 then the login failed!

Example https-request:

https://<Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=Login&gsUser=103&gsPass=aBc12345

Example https-request Description:

The user 103 logs in to the Web Services server.

When the login fails the ID is always zero and the optional <ERROR> tag is present. The reason can have the following values:

- LOGIN_FAILED Password wrong or user unknown, user not allowed to login. No automatic relogin attemps should be performed. Instead, the user should be asked to enter valid credentials.
- SERVICE_STARTING The server is starting up, login should be retried after 30sec.
- LOGIN_PWWEAK The user has to configure a new password before the Web Services Interfacecan be utilized. No automatic relogin attemps should be performed. Please note that additional reasons may be introduced at any time.

4.2. Logout

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

Ends the current session. The session is identified with the gsSession parameter.

Format: cmd=Logout

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<LOGOUT/>
```

Example https-request:

https://<Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=Logout&gsSession=xxxxxxxx

Example https-request Description:

The user 103 logs out from the Web Services server.

5. Call control

5.1. AnswerCall

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

The AnswerCall service answers an alerting call. The call will be answered in hands-free mode of the phone.

Parameter	Type	Description
deviceID	**deviceID**	The device which is ringing and should be answering the call.

Return Values:

Positive:

```
<AnswerCallResponse/>
```

Negative:

```
<CSTAErrorCode>
        CSTA error response
</CSTAErrorCode>
```

Example https-request:

https://<Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=AnswerCall&deviceID=103&gsSession=xxxxxxxx

Example https-request Description:

The device 103 is answering the call in hands-free mode.

5.2. ClearConnection

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

Clear connection terminates the current call for a specific phone.

Parameter	Type	Description
deviceID	**deviceID**	A device participating in a call that should be cleared.

Return Values:

positive

```
<ClearConnectionResponse/>
```

Negative:

```
<CSTAErrorCode>
        CSTA error response code
</CSTAErrorCode>
```

Example https-request:

https://<Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=ClearConnection&deviceID=103&gsSession=xxxxxxxx

Example https-request Description:

The connection of an active call of the phone 103 will be terminated.

5.3. DeflectCall

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

This function will deflect an incoming call from the original target to a new target. The first target will be not affected by this action.

Parameter	Type	Description
deviceID	**deviceID**	The device ID of the alerting phone.
newDestination	**deviceID**	The device ID for the target of the deflection.

Return Values:

Positive:

```
<DeflectCallResponse/>
```

Negative:

```
<CSTAErrorCode>
        CSTA error response code
</CSTAErrorCode>
```

Example https-request:

https://<Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=DeflectCall&deviceID=103&newDestination=109&gsSession=xxxxxxxx

Example https request Description

Deflects a call, without answering, from device 103 to device 109. Device 103 is ringing and the call should move to another phone without conversation. The DeflectCall command moves the call from the ringing 103 to 109. Phone 109 is ringing now and phone 103 is put back into normal mode with no activity.

5.4. MkCall

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

Initiates a basic call between the calling device and target device (called device), addressed by the calledDirectoryNumber. This is the asynchronous version of MakeCall command. The server responds immediately and the call is performed in the background. The result of the command is returned via the event channel as a MakeCall event. MkCall is possible when there is no active call. In order to establish a call the source phone will initiate the call towards the target.

Parameter	Type	Description
callingDevice	**deviceID**	The device which should be performing the call.
calledDirectoryNumber	**deviceID**	The target device that should be called.

Positive:

```
<DONE/>
```

Example https-request:

https://<Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=MkCall&callingDevice=101&calledDirectoryNumber=103&gsSession=xxxxxxxx

Example https-request Description:

The device 101 initiates a call to device 103. The device 101 is in hands-free-mode and the device 103 is ringing.

5.5. MakeCall

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

Initiates a basic call between the calling device and target device (called device), addressed by the calledDirectoryNumber. MakeCall is a blocking commend. Dependent on the target number / network it may take many seconds before the command returns, or even a timeout may occur. MakeCall is possible when there is no active call. In order to establish a call the source phone will initiate the call towards the target.

Parameter	Type	Description
callingDevice	**deviceID**	The device which should be performing the call.
calledDirectoryNumber	**deviceID**	The target device that should be called.

Return Values:

Positive:

```
<MakeCallResponse>
  <CallingDevice>
    <deviceID>device id</deviceID>
    <callID>call id</callID>
  </CallingDevice>
</MakeCallResponse>
```

Negative:

```
<CSTAErrorCode>
      CSTA error response code
</CSTAErrorCode>
```

Example https-request:

https://<Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=MakeCall&callingDevice=101&calledDirectoryNumber=103&gsSession=xxxxxxxx

Example https-request Description:

The device 101 initiates a call to device 103. The device 101 is in hands-free-mode and the device 103 is ringing.

Example, possible XML response from Web Services

```
<?xml version="1.0" encoding="UTF-8"?>
<MakeCallResponse>
  <CallingDevice>
    <deviceID>103</deviceID>
    <callID>00000010</callID>
  </CallingDevice>
</MakeCallResponse>
```

5.6. HoldCall

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

Puts an active call on consultation hold. Can only be used if there is an active and answered call.

Parameter	Type	Description
deviceID	**deviceID**	The device ID of the phone number which initiates the Hold

Return Values:

Positive:

```
<HoldCall/>
```

Negative:

```
<CSTAErrorCode>
        CSTA error response code
</CSTAErrorCode>
```

Example https-request:

https://<Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=HoldCall&deviceID=103&gsSession=xxxxxxxx

Example https-request Description:

If device 103 is in an active call, the command puts the active call on hold for other activities.

5.7. RetrieveCall

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

Retrieves a held call. This can be used to end the consultation call and get back to the original (held) call. It is only possible for calls that are in hold mode.

Parameter	Type	Description
deviceID	**deviceID**	Device that wants to get back a held call.

Return Values:

Positive:

```
<RetrieveCall/>
```

Negative:

```
<CSTAErrorCode>
        CSTA error response code
</CSTAErrorCode>
```

Example https-request:

https://<Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=RetrieveCall&deviceID=103&gsSession=xxxxxxxx

Example https-request Description:

The held call of device 103 is reactivated. The held call is active for conversation now.

5.8. ConsultationCall

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

Performs a consultation call on the device, where “newdst” defines the destination number for the consultation call.

Parameter	Type	Description
deviceID	**deviceID**	The controlled device.
callID	**callID**	The call ID of the active call
newdst	**deviceID**	The call number to consult

Please note that call IDs are hexadecimal. If the call ID is missing, it is guessed by the server.

Return Values:

Positive:

```
<ConsultationCallResponse>
  <callingDevice>
    <deviceID>device id</deviceID>
    <callID>call id</callID>
  </ callingDevice >
</ ConsultationCallResponse >
```

Negative:

```
<CSTAErrorCode>
        CSTA error response code
</CSTAErrorCode>
```

Example https-request:

https://<Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=ConsultationCall&deviceID=100&newdst=101&gsSession=xxxxxxxx

Example https-request Description:

Device 100 performs a consultation call to device 101.

5.9. AlternateCall

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

Alternates between calls in case of a consultation scenario. “callID_h” (call ID of the held call) and callID_a (call ID of the active call) are given in hexadecimal format.

Parameter	Type	Description
deviceID	**deviceID**	The controlled device.
callID_h	**callID**	The call ID of the held call
callID_a	**callID**	The call ID of the active call

Please note that call IDs are hexadecimal. If the call ID is missing, it is guessed by the server.

Return Values:

Positive:

```
<AlternateCallResponse/>
```

Negative:

```
<CSTAErrorCode>
        CSTA error response code
</CSTAErrorCode>
```

Example https-request:

https://<Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=AlternateCall&deviceID=100&callID_h=x&callID_a=y&gsSession=xxxxxxxx

Example https-request Description:

Device 100 alternates between calls with the given call IDs.

5.10. ReconnectCall

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

Performs a reconnect to the held party after a consultation call has been initiated. The party which in in consultation hold state is reconnected. The other call (if any) is terminated.

"callID_h" defines the callID of the held call in hexadecimal format.

Parameter	Type	Description
deviceID	**deviceID**	The controlled device.
callID_h	**callID**	The call ID of the held call.

Please note that call IDs are hexadecimal. If the call ID is missing, it is guessed by the server.

Return Values:

Positive:

```
<ReconnectCallResponse/>
```

Negative:

```
<CSTAErrorCode>
        CSTA error response code
</CSTAErrorCode>
```

Example https-request:

https://<Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=ReconnectCall&deviceID=100&callID_h=x&gsSession=xxxxxxxx

Example https-request Description:

Device 100 performs a reconnect to the help call with callID_h x.

5.11. ConferenceCall

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

This request is used to establish a three-party conference within the context of a consultation call, which means that both the help and the active consultation parties are connected to the user.

Parameter	Type	Description
deviceID	**deviceID**	The controlled device.
callID_h	**callID**	The call ID of the held call.
callID_a	**callID**	The call ID of the active call.

Please note that call IDs are hexadecimal. If the call ID is missing, it is guessed by the server.

Return Values:

Positive:

```
<ConferenceCall/>
```

Negative:

```
<CSTAErrorCode>
        CSTA error response code
</CSTAErrorCode>
```

Example https-request:

https://<Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=ConferenceCall&deviceID=100&callID_h=x&callID_a=y&gsSession=xxxxxxxx

Example https-request Description:

Device 100 establishes a three party conference while being in a consultation call.

Note: This method refers to system conferences only, but not to any kind of managed UC conference for UC Smart or UC Suite.

5.12. EndConference

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

This request is used to terminate a three-party conference. This method can only be used by the master of the conference (i.e. the party which has established the conference).

Parameter	Type	Description
deviceID	**deviceID**	The controlled device.
callID	**callID**	The call ID of the conference call.

Please note that call IDs are hexadecimal.

Return Values:

Positive:

```
<EndConference/>
```

Negative:

```
<CSTAErrorCode>
        CSTA error response code
</CSTAErrorCode>
```

Example https-request:

https://<Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=EndConference&deviceID=100&callID=x&gsSession=xxxxxxxx

Example https-request Description:

Device 100 terminates the three-party conference which it has established before via ConferenceCall command.

5.13. TransferCall

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

This request is used to make a blind transfer, without making a previous consultation to the transfer target device. This feature is also known as “Single Step Transfer”. Condition for this is that a call is active between your phone and the calling device and the target phone has no active call running.

Parameter	Type	Description
deviceID	**deviceID**	The controlled device.
transferredTo	**deviceID**	The transfer target device.
callID	**callID**	The call ID of the active call.

Please note that call IDs are hexadecimal. If the call ID is missing, it is guessed by the server.

Return Values:

Positive:

```
<SingleStepTransferCallResponse>
  <transferredCall>
    <deviceID>device id</deviceID>
    <callID>call id</callID>
  </transferredCall>
</SingleStepTransferCallResponse>
```

Negative:

```
<CSTAErrorCode>
        CSTA error response code
</CSTAErrorCode>
```

Example https-request:

https://<Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=TransferCall&deviceID=103&transferredTo=108&callID=abc&gsSession=xxxxxxxx

Example https-request Description:

For example there is a call established between the devices 100 and 103. The https-request transfers this call to device 108 (connecting devices 103 and 108).

Example, possible XML response from Web Services:

```
<?xml version="1.0" encoding="UTF-8"?>
<SingleStepTransferCallResponse>
  <transferredCall>
    <deviceID>103</deviceID>
    <callID>1ab</callID>
  </transferredCall>
</SingleStepTransferCallResponse>
```

5.14. AttendedTransfer

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

This request is used to make an attended transfer. Precondition is that the controlled device has an active call and a held call. The active and the held call will be connected.

Parameter	Type	Description
deviceID	**deviceID**	The controlled device.
callID_h	**callID**	The call ID of the held call.
callID_a	**callID**	The call ID if the active call.

Please note that call IDs are hexadecimal. If a call ID is missing, it is guessed by the server.

Return Values:

Positive:

```
<TransferCallResponse/>
```

Negative:

```
<CSTAErrorCode>
        CSTA error response code
</CSTAErrorCode>
```

Example https-request:

https://<Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=AttendedTransfer&deviceID=103&callID_h=x&callID_a=y&gsSession=xxxxxxxx

Example https-request Description:

For example user 103 has put user 100 on consultation hold (callID_h = x) and makes a consultation call to user 104 (callID_a =y). The https-request transfers 100 to the consultation device that currently is in an active call (connecting devices 100 and 104).

5.15. GetServerParam

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

Retrieves dial related configuration parameters from the server. These can be used e.g. to pre-process call numbers in the client.

Response structure:

```
<SERVERPARAM>
<DIALOUT>dialout prefix</DIALOUT>
<INTERNATIONAL>intl. prefix</INTERNATIONAL>
<NATIONAL>nat.dial prefix</NATIONAL>
<COUNTRY>country code</COUNTRY>
<AREA>area code</AREA>
<SYSNUM>system number for DID</SYSNUM>
<VOICEMAIL>voicemail access number</VOICEMAIL>
</SERVERPARAM>
```

Note: Dependent on the software version of the server and the system configuration, additional data may be found in the response structure. Such data should **not** be utilized because it may not be present with future software versions.

Example https-request:

https://<Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=GetServerParam&user=100&gsSession=xxxxxxxx

6. Device control

6.1. GetDoNotDisturb

Status: **deprecated** - use 'GetDeviceState' instead

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

Requests the DND ("do-not-disturb") status for a certain phone. For incoming call the phone is not available as valid target.

Parameter	Type	Description
device	**deviceID**	The phone which "DoNotDisturb" status should be read.

Return Values:

Positive:

```
<GetDoNotDisturbResponse>
   <doNotDisturbOn>true / false</doNotDisturbOn>
</GetDoNotDisturbResponse>
```

Negative:

Not defined.

Example https-request:

https://<Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=GetDoNotDisturb&device=103&gsSession=xxxxxxxx

Example https-request Description:

The "do-not-disturb" status for device 103 is not (false) activated. This information can be found in the XML answer of this request.

Example, possible XML response from Web Services

```
<?xml version="1.0" encoding="UTF-8"?>
<GetDoNotDisturbResponse>
   <doNotDisturbOn>false</doNotDisturbOn>
</GetDoNotDisturbResponse>
```

6.2. GetForwarding

Status: **deprecated** - use ‘GetDeviceState’ instead

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

Retrieves the forwarding configuration for target device.

Parameter	Type	Description
device	**deviceID**	The phone that’s configuration should be read.

Return Values:

Positive:

```
<GetForwardingResponse>
  <forwardingList>
    <forwardingListItem>
          <forwardingType>forwardImmediate</forwardingType>
          <forwardStatus>true / false</forwardStatus>
    </forwardingListItem>
  </forwardingList>
</GetForwardingResponse>
```

Negative:

Not defined

Example https-request:

https://<Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=GetForwarding&device=103&gsSession=xxxxxxxx

Example https-request Description:

Gets the forwarding status for the device 103.

Example, possible XML response from Web Services

```
<?xml version="1.0" encoding="UTF-8"?>
<GetForwardingResponse>
  <forwardingList>
    <forwardListItem>
      <forwardingType>forwardImmediate</forwardingType>
      <forwardStatus>false</forwardStatus>
    </forwardListItem>
  </forwardingList>
</GetForwardingResponse>
```

6.3. ClearDisplay

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

Clears any text that was previously set by any application via SetDisplay on a certain phone. The phones standard display will appear again. If nothing is displayed on the phone this command won't have any effect.

Parameter	Type	Description
device	**deviceID**	The target device

Return Values:

Positive:

```
<ClearDisplay/>
```

Negative:

Not defined

Example https-request:

https://<Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=ClearDisplay&device=103&gsSession=xxxxxxxx

Example https-request Description:

Clears (resets) the display of device 103.

6.4. SetDisplay

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

Sets the display text for a certain phone. In case of no other application is overwriting this space, the text will not be deleted until it is cleared via ClearDisplay by any application.
Important note: To avoid overload problems in the server, this function must not be used to create dynamic text effects (e.g. text scrolling). It may only be used to display static texts.

Parameter	Type	Description
device	**deviceID**	The target phone.
contentsOfDisplay	**Text**	The display text.

Return Values:

Positive:

```
<SetDisplay/>
```

Negative:

Not defined

Example https-request:

https://<Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=SetDisplay&device=103&contentsOfDisplay=testtext&gsSession=xxxxxxxx

Example https-request Description:

Writes the text “testtext” onto the display of device 103.

6.5. SetDoNotDisturb

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

Set “do-not-disturb status” for a certain phone. If this feature is activated then every call will be blocked. No forwarding is activated by this function.

Parameter	Type	Description
device	**deviceID**	The phone that should be set.
doNotDisturbOn	**Boolean**	Enable / Disable the setting with this parameter.

Return Values:

Positive:

```
<DONE/>
```

Negative:

```
<ERROR>
        <CODE>class:code</CODE>
        <DEV>device</DEV>
</ERROR>
```

Example https-requests:

https://<Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=SetDoNotDisturb&device=103&doNotDisturbOn=true&gsSession=xxxxxxxx

https://<Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=SetDoNotDisturb&device=103&doNotDisturbOn=false&gsSession=xxxxxxxx

Example https-request Description:

The first example activates the “do-not-disturb” status for device 103. The second one disables the “do-not-disturb” status for device 103.

6.6. SetForwarding

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

For a certain phone, all incoming calls are forwarded to another device (unconditional forwarding). This feature can be deactivated too. If a forwarding is active, a second forward with “activeForward=true” parameter will replace the first one.

Parameter	Type	Description
device	**deviceID**	The phone that should forward incoming calls.
forwardDN	**deviceID**	The phone that should receive the calls instead (do not use for deactivation).
activateForward	**Boolean**	Enable / Disable the setting with this param.

Return Values:

Positive:

```
<DONE/>
```

Negative:

```
<ERROR>
        <CODE>class:code</CODE>
        <DEV>device</DEV>
</ERROR>
```

Example https-requests

https://<Web Services IP address>:8802/cgi- bin/gadgetapi?cmd=SetForwarding&device=103&forwardDN=109&activateForward=true&gsSession=xxxxxxxx

https://<Web Services IP address>:8802/cgi- bin/gadgetapi?cmd=SetForwarding&device=103&activateForward=false&gsSession=xxxxxxxx

Example https-request Description:

In the first example the forward destination is programmed. Now every call for device 103 is sent to device 109. The forwarding target is shown on the display of the phone.

The second example is to delete the forward destination. All calls will arrive at device 103 again by sending this command.

6.7. GetDeviceState

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

Retrieves the device status of the user's device.

Format:

https://<Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=GetDeviceState&device=100&gsSession=xxxxxxxx

The server returns the device state in the format:

```
<?xml version="1.0" encoding="UTF-8"?>
<DevState user="100">
<CallList>
<Ops></Ops>
<Call>
<Calling name="100">100</Calling>
<Called name="MULAP102">102</Called>
<AnsDev name="MULAP102">102</AnsDev>
<CallID.ref>1</CallID.ref>
<CallID.dev>**102</CallID.dev>
<State>Established</State>
<Flags>
        <F>Outgoing</F>
</Flags>
<Ops>clearcall,blind_transfer,consultation</Ops>
</Call>
</CallList>
<PRESENCE>
<USER>100</USER>
<STATE>1</STATE>
<PTEXT></PTEXT>
<CONNECTED>true|false</CONNECTED>
<PHSTATE inout="" calltype="">Established</PHSTATE>
</PRESENCE>
<Forwarding></Forwarding>
<DND>Off</DND>
</DevState>
```

The *PRESENCE* block is identical to GetPresenceXML. NEW: the PHSTATE has an attribute calltype="type" where type can be empty, int or ext depending whether the ACTIVE call is internal or external. ONLINE tag holds the information if a user is logged in with at least one client (must be smartclient).

Field '*DND'* shows if the device has DND set or not, possible values: On/Off

Field '*Forwarding'* holds the forwarding destination (or empty if no forwarding is set). If it is forwarded to the voicemail number (to the one presented via GetServerParam) instead of the number $VOICEMAIL is set.

CallList block holds information about the actual calls/connections on the device and about the available call operations in the current state. All *Ops* fields are comma separated list. The order is not fixed.

Format:

<CallList>

<Ops></Ops> <!—operation for the device

<Call> <!—Multiple call blocks

</Call>

</CallList>

The <Ops> for the device can be:

- makecall when the makecall command is possible

The call block has the following format:

<Call>

<Calling name="">phone number</Calling>

<Called name="">phone number</Called>

<AnsDev name="">phone number</AnsDev>

<CallID.ref>callid</CallID.ref>

<CallID.dev>devid</CallID.dev>

<State>call state</State>

<ConfParties parties="x" master="100"> <!-- Conditional

</ConfParties>

<Flags>

<F></F> <!—Multiple <F>s are allowed

</Flags>

<Ops></Ops> <!—Comma separated list

</Call>

<State> can be:

- Established
- Hold <!—the call is on hold, either Active or Passive
- Ringing <!—The call is alerting
- Queued <!—Call waiting
- ServiceInitiated <!— The call has no remote party yet. It can indicate an offhook waiting for the user dial or it can appear for a short time during call setup.

The <F></F> can hold the following values:

- HoldA The call is on Hold, the holding party
- HoldP The call is on Hold, the held party
- Queued The call is waiting
- Busy The called party is busy
- Incoming It is an incoming call
- Outgoing It is an outgoing call
- Blocked Blocked
- DstNo Destination Not Obtainable, no such device for example
- Callback The call is a callback
- Conference The call is a conference, <ConfParties> is there

- ConferenceMaster The device is the master of the conference
- Internal The call is internal (both parties internal devices)
- External One of the party is not an internal device

calltype can be:

- internal
- external
- none

inout can be:

- incoming (the call is an incoming call)
- outging (the call is outging call)
- none (the call is neither of the above or cannot be determined)

Ops can have the following values:

- alternate
- hold
- attended_transfer
- rertrieve
- clearcall
- conference
- blind_transfer
- consultation
- answercall
- deflect
- remove_party
- reconnectcall
- end_conference
- leave_conference

If the call is a conference the <*ConfParties*> holds information about the conference:

<ConfParties parties=”x” master=”100”>

<Party> <!—can occur multiple times

<Device name=””>phone number</Device>

<CallID.ref></CallID.ref>

<CallID.dev></CallID.dev>

</Party>

</ConfParties>

The *parties* attribute shows the number of participant in the conference. The *master* attribute shows which device is the master. Depending whether the current device is the master or not the list of participants is sent. If the current device is not the master only the ‘self’ party is sent.

The <CallID.*> holds the connection information for a certain party.

7. Monitoring and events

Important: In general, the Web Service Interface events are encoded in JSON, but it is necessary to check the content type of any response because other content types like text/xml or text/plain may occur for example in error responses or in case of a timeout.

7.1. GetEvents

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

Fetches events[2] for the given device object. The function returns events only if the device has an active monitor and events occur. If no events can be obtained immediately, the Web Services holds the connection for aprox. 30 seconds and responds immediately when an event occurs in this period of time. Of course you can query the Web Services again if there was no event. If you want to receive events continuously, it is required to reactivate the GetEvents function after every event and after every timeout. To avoid overload on server side, any client must ensure that a minimum of 1 second delay is used between two concurrent GetEvents calls.

It is absolutely necessary to use EventStart before, otherwise the GetEvents function will not return any events.

Parameter	Type	Description
deviceObject	**deviceID**	Specifies the user to listen for events (this is usually the login id).

Return Values Example (for a HookEvent):

Positive:

{"error":0,"events":[{"jsonEvent":{"HookEvent":[{"deviceID":"111","intext":"int","hook":"Established",”inout”:”incoming”},{"deviceID":"110","intext":"int","hook":"Established",”inout”:”outgoing”}]},"EventForDevice":"100","type":"HookEvent"}}}

The positive result is of content type application/json, containing an array of events or an empty array. A positive response may contain one or more events. The client application has to evaluate this JSON string.

Timeout:

An empty response (content type: plain/text) is sent in case that no events occurred. The client application should call GetEvents again.

Example https-request (please note the specific, deviant syntax of this command: ...gadgetapi/GetEvents?...)

https://<Web Services IP address>:8802/cgi-bin/gadgetapi/GetEvents?deviceObject=103&gsSession=xxxxxxxx

Example https-request Description:

Retrieves the events from the server for device 103.

[2] Dependent on the evnt filters that have been set.

Event overview:

The events returned by the GetEvents service of the Web Services have a JSON structure. The returned value can contain zero or more events. Hence the single events are packed in an Array with syntax: [{...},{...},...,{...}] A returned value with no events would look like this: [].

All events have a type, EventForDevice and a receivedAt element in their content. The type of an element defines the structure of its content. Different event types have different additional content and a jsonEvent where all event types are shown that are responded by the PBX. In the following overview the event types which are designed for general use and their meaning are listed.

Event type	*Meaning*
HookEvent	The hook state of a device changed
DevStateChange	The status of a device has been changed
Journal	The user's call journal changed (only with UC Smart)
Phonebook	The content of a phonebook changed (only with UC Smart)
PresenceEvent	The presence status of a user changed

Please note that dependent on the OpenScape Business software version and UC application that is used other events may occur which are designed for use only with specific client applications. These should be ignored.

Detailed description of the events:

HookEvent:

{"error":0,"events":[{"jsonEvent":{"HookEvent":[{"deviceID":"111","intext":"int","hook":"Established","inout":"incoming"},{"deviceID":"110","intext":"int","hook":"Established","inout":"outgoing"}]},"EventForDevice":"100","type":"HookEvent"}}}

HookEvent tag is an array containing the devices whose hook state was changed since
the last monitor event occurred (any type). One block consists of:

- deviceID: the device whose hook state was changed
- intext: the overall internal/external call type for the device. It can be: "int", "ext", "none"
- hook: The hook state for the device. "Idle", "Established", "Ringing", "Hold", "Queued", "None", "ServiceInitiated". None means it can't be determined.
- inout: Shows whether the call incoming or outgoing. Possible values are: "incoming", "outgoing", "none"

The existence of a hook event is necessary but not sufficient condition for hook state change. It might happen that it is sent when the actual state is not changed or if the actual state changed back to the same state the client knows actually.

DevStateChange event:

{"jsonEvent":{"DevStateChange":{"delta":"call,forwarding,dnd,presence"}},"receivedAt":1291383106,"EventForDevice":"101","type":"DevStateChange"}

It is changed whenever a change occurs in the DeviceState. The "delta" holds the changed parts since the last read. Even if there is more than one change since the last read (with GetEvents) only one event is generated.

The state change is "cleared" with the GetEvents. Even if the GetDeviceState is called the event remains there until it is read.

Journal event: (only available with UC Smart)

{jsonEvent:{Journal:{evt:”new,missed,…,JID:0},receivedAt:time,EventForDevice:”100”,type:”Journal”}

The evt identifies the subevent:

- new: There is a new entry recorded in the journal
- deleted: A journal entry is removed: jid is non 0; multiple entry is removed: jid is missing
- missed: A new missed call has just been added to the journal
- missed_clr: one of the clients cleared the new journal indication flag
- note_add: Note added to journal entry
- note_update: Note message or date is updated
- note_del: Note removed
- note_alarm: Alarm is set and it is pending

The JID parameter is optional. For “new” and “missed” events it holds the ID for the journal entry. This ID can be used as a parameter to the JournalRead command to retrieve the record.

Phonebook event: (only available with UC Smart)

{jsonEvent:{Phonebook:{evt:”new,update,deleted, import, removall,…”,ID:”0”,
book:”pdir,edir,…},receivedAt:time,EventForDevice:”100”,type:”Phonebook”}

The server sends a Phonebook event if the phonebook is changed. Parameter ‘book’ describes which phonebook is modified, the ‘ID’ holds the ID of the modified entry. If ‘ID’ is missing then even the ID can’t be determined or the operation influenced the whole phonebook.

The evt can be:

- new: A new entry is added to the phonebook (use PhBookDetail to fetch the entry)
- update: The entry has been updated (use PhBookDetail to fetch the new values)
- deleted: The entry is removed
- import: An import operation was done and the phonebook should be reloaded
- removeall: All entries are removed from the phonebook
- reload: The phonebook was modified and the number of changes was too high or the changes cannot be identified individually

Not all phonebook types send events and not all events sent by a phonebook type.

PresenceEvent:

Presnum is a presence state number

ptext and ptime are optional.

{jsonEvent:{PresenceEvent:{deviceID:”102”,newstate:presnum,ptext:”anything”, ptime:””,online:”true/false”,connected:”true/false”}},
receivedAt:time,EventForDevice:”100”,type:”PresenceEvent”}

Time format: YYYY-MM-DD hh:mm:ssZ

The event is sent when a user changes his presence (state, text or return date) (and the visibility is set to internal or internal/external) or when the online flag changes. This event is not **sent** when the phstate or connected state changes.

7.2. EventStart

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

Starts event monitoring for a device, in order to get information about its events. These events will be collected and can be retrieved with the GetEvents request. Via the filter parameter a client can specify which event types he wants to receive.

The monitor will be active until it is stopped or the session ends.

Parameter	Type	Description
deviceObject	**deviceID**	The phone which has to be monitored
filter	**eventtype**	Optional. Monitor only events matching the filter.

Event types supported by the filter:

- PRESENCE: Presence events, sent when: presence is changed, device connected state is changed
- DEVSTATE: sent whenever something changes inside the DevState structure: Call, forwarding, DND, presence
- HOOKSTATE: for subscribed devices hook state change events will be sent depending on the presence visibility setting of the user
- JOURNAL: Sent when the call journal changed (UC Smart only)
- PHBOOK: informs about changes for phone book types (UC Smart only)

Return Value:

```
<DONE/>
```

Example https-request

https://<Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=EventStart&deviceObject=103&filter=DEVSTATE,PRESENCE&gsSession=xxxxxxxx

Example https-request Description:

Starts monitoring for device 103.

7.3. EventStop

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

Stops monitoring for a phone.

Parameter	Type	Description
deviceObject	**deviceID**	The phone where monitoring should stop.

Return Value:

```
<DONE/>
```

Example https-request

https://[Web Services IP address]:8802/cgi-bin/gadgetapi?
cmd=EventStop&deviceObject=103&gsSession=xxxxxxxx

Example https-request Description:

Stops monitoring for device 103.

7.4. MonitorStart

Status: **deprecated** - use 'EventStart' instead

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

Starts event monitoring for a phone, in order to get information about its events. These events will be collected and can be retrieved with the GetEvents request. Via the filter parameter a client can specify which event types he wants to receive.

The monitor will be active until it is stopped or the session ends.

Parameter	Type	Description
deviceObject	**deviceID**	The phone which has to be monitored
filter	**eventtype**	Optional. Monitor only events matching the filter.

Event types supported by the filter:

- PRESENCE: Presence events, sent when: Presence is changed, device connected state is changed
- DEVSTATE: sent whenever something changes inside the DevState structure: Call, forwarding, dnd, presence
- HOOKSTATE: for subscribed devices hook state change events will be sent depending on the presence visibility setting of the user
- JOURNAL: Sent when the call journal changed (UC Smart only)
- PHBOOK: informs about changes for phone book types (UC Smart only)

Return Values

Positive:

MonitorStartResponse

Negative:

Not defined

Example https-request

https://<Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=MonitorStart&deviceObject=103&filter=DEVSTATE,JOURNAL&gsSession=xxxxxxxx

Example https-request Description:

Starts monitoring for device 103.

Example possible, positive JSON response from Web Services

```
{"CrossRefID":"0001","error":0,"respType":"MonitorStartResponse","MonitorCount":3}
```

7.5. MonitorStop

Status: **deprecated** - use 'EventStop' instead

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

Stops monitoring for a phone.

Parameter	Type	Description
deviceObject	**deviceID**	The phone where monitoring should stop.

Return Values:

Positive:

```
MonitorStopResponse
```

Negative:

Not defined

Example https-request

https://[Web Services IP address]:8802/cgi-bin/gadgetapi?
_cmd=MonitorStop&deviceObject=103&gsSession=xxxxxxxx

Example https-request Description:

Stops monitoring for device 103.

Example, possible JSON response from Web Services

```
{"error":0,"respType":"MonitorStopResponse"}
```

7.6. GetFilter

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Format:

cmd=GetFilter&user=100

Description

Returns the Filter list associated with the user in a session.

Response (example):

```
<?xml version="1.0" encoding="UTF-8"?>
<GetFilter>
PRESENCE
DIALPARAM
</GetFilter>
```

The list is new line separated. CRE is Call Related Events (not CSTA based).

7.7. SetFilter

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Format:
cmd=SetFilter&user=100&filter={PRESENCE,DIALPARAM,DEVSTATE,HOOKSTATE,USERCONF,JOURNAL,PHBOOK}

Description

Sets the event filter for a user within an session. The values passed in ‘filter’ are comma separated. If none is specified all events are filtered (nothing is sent).

It controls the events for a session regardless which or how many devices are monitored.

Supported Filters:

PRESENCE: Presence event, sent when: presence is changed, device connected state is changed

DIALPARAM: Dial parameters + voicemail number changed

DEVSTATE: sent whenever something changes inside the DevState structure; Call, forwarding, dnd, presence

HOOKSTATE: for subscribed devices hook state change events will be sent
(in UC Smart: dependent on the presence visibility setting of the user)

USERCONF: Sent when user details (profile data) changed

JOURNAL: Sent when the journal changed (UC Smart only)

PHBOOK: informs about changes for phone book types.

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<SetFilter/>
```

7.8. HookSubscribe

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Format: cmd=HookSubscribe&user=100&devices=101,102,...

Description

The specified user subscribes for hook state changes of devices 101,102,...
The “devices” parameter is a comma separated list of device numbers. The monitor must be started and the filter HOOKSTATE has to be set to receive hook state change events.

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<HOOKSUBSCRIBE>
<DEV id=”101”>yes|no</DEV>
<DEV id=”102”>yes|no</DEV>
...
</HOOKSUBSCRIBE>
```

7.9. HookUnsubscribe

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Format: cmd=HookUnsubscribe&user=100&devices=101,102,...

Description

Stops the eventing for hook state changes of devices 101,102,...

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<HOOKUNSUBSCRIBE/>
```

7.10. Error Codes and Responses

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Commands can return either execution specific errors and results (see specific commands) or general ones. General ones can be:

```
<ERROR>
  <REASON></REASON>
</ERROR>
```

Where REASON can be one of:

- MISSING_PARAM, When at least 1 parameter is missing
- INVALID_PARAM, When the value of a parameter is incorrect
- UNKOWN_CMD, Command not known
- NOT_LOGGED_IN, client not logged in
- WOULD_BLOCK, When there is no LAC server address configured
- NOPERM, No Permission to execute this command for the given user
- NOCONN No connection to telephony system (LAC)

8. Phonebooks

There is a general interface to all types of phonebook. Please note that some phonebooks offer only a subset of the listed data set. The phonebook can contain several entries for a user. Currently the following entries are supported (dependent on the phonebook type):

- Surname
- Firstname
- Phone
- Phone2
- Phone3
- Email

For requests the ‘book’ parameter is mandatory. It indicates which phonebook type the access will be made to. See below for details.

8.1. PhBookSearch

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

Reads the phonebook given with the “book” argument for a user. A maximum of 100 search results (maxcnt parameter) can be requested. If the cnt parameter of the response indicates that more than maxcnt results were found, cnt will be bigger than maxcnt. In this case, any client should ask the user to refine the search query.

If the “name” parameter is present then the firstname and surname parameters are ignored and the specified string is searched in the lastname and in the firstname as well with a logic OR operator.

Format:

cmd=PhBookSearch&user=100&book={int|all|edir|pdir|ext}[&surname=Smith&firstname=John&name=Joe&start=1&maxcnt=100]

Response (Example):

```
<?xml version="1.0" encoding="UTF-8"?>
<PHBOOK cnt=”100”>
  <ITEM>
  <ID>98765</ID>
  <SURNAME>Dubois</SURNAME>
  <FIRSTNAME>Natalie</FIRSTNAME>
  <DISPLAYNAME>Dubois, Natalie</DISPLAYNAME>
  <PHONE>345</PHONE>
  <PURL>/pictures/unknown2.png</PURL>
  <PSURL>/pictures/unknown2_small.png</PSURL>
  <SRC>0</SRC>
  <TYPE>int</TYPE>
  </ITEM>
</PHBOOK>
```

Phonebook types:

- int: Internal users (if not excluded from the phonebooks via system configuration)
- edir: Speed dials
- pdir: Personal directory
- ext: External directory (UC Suite) or Global directory (UC Smart)
- all: All phonebooks

8.2. PhBookDetail

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

This command gives the details about a phonebook entry.

Format:

cmd=PhBookDetail&user=100&book={int|all|edir|pdir|ext}&id=xxx

where id is the ID of an existing phonebook entry.

It returns the following on success (example):

```
<?xml version="1.0" encoding="UTF-8"?>
<ITEM>
  <ID>%d</ID>                ID for the entry (for further operations)
  <USER> user </USER>
  <SURNAME> surname </SURNAME>
  <FIRSTNAME> firstname </FIRSTNAME>
  <PHONE> 01234567 </PHONE>
  <PHONE2> 01234568 </PHONE2>
  <PHONE3> 01234569 </PHONE3>
  <EMAIL> email </EMAIL>
</ITEM>
```

8.3. PhBookLookup

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

This command tries to resolve a phone book entry by the given call number. It returns zero or one result. The mechanism used a “similar to” algorithm to find a best match, independent of the given number format in case of an external/national/international call number.

Format:

cmd=PhBookLookup&user=100&number=01234567

where number is a given phone number.

It returns the following on success (example):

```
<?xml version="1.0" encoding="UTF-8"?>
<ITEM>
  <ID>%d</ID>                ID for the entry (for further operations)
<SURNAME>Mertens</SURNAME>
<FIRSTNAME>Per</FIRSTNAME>
<DISPLAYNAME>Mertens, P.</DISPLAYNAME>
<PHONE>+43 5 678-45678</PHONE>
<PHONE3>+43 676  345678</PHONE3>
<ORGANIZATION>Atos</ORGANIZATION>
<CITY>Vienna</CITY>
<EMAIL>per.mertens@atos.net</EMAIL>
<TYPE>ext</TYPE>
</ITEM>
```

9. Presence

9.1. SetPresence

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Format: https:// <Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=SetPresence&user=100&presence=state_id&ptext=txt&ptime=time

The state_id (presence state) can be:

Numerical STATE	**Meaning**
1	Office
2	Meeting
3	Sick
4	Break
5	GoneOut
6	Holiday
7	Lunch
8	Home
9	DND (*Do Not Disturb)

The ptext parameter is optional.

The time format for the ptime parameter is: yyyy-mm-dd hh:mmZ, e.g. 2011-10-15 08:30Z. ptime is optional in the context of UC Smart, but mandatory in UC suite (except for status “Office”).

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<DONE/>
```

9.2. SetCallMe

Status: current

Supported platforms: ( ) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Format: https:// <Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=SetCallMe&user=100[&callme=phonenumber]

Description

Sets the CallMe state for the user if the CallMe number is specified. Otherwise it will deactivate the CallMe state (presence changes to Office). Please note that the CallMe feature is only available with UC Suite.

9.3. GetPresenceXML

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Format:

https:// <Web Services IP address>:8802/cgi-bin/gadgetapi?cmd=GetPresenceXML&user=100,101,...

The server returns the presence state for the listed users in the format:

```
<?xml version="1.0" encoding="UTF-8"?>
<PRESENCE_LIST>
  <PRESENCE>
    <USER></USER>
    <STATE></STATE>     <!—presence state
    <PTEXT><PTEXT>
    <PTIME> </PTIME>
    <CONNECTED> </CONNECTED>
    <PHSTATE>callstate</PHSTATE>
  </PRESENCE>
</PRESENCE_LIST>
```

A maximum of 50 users can be listed in a single GetPresenceXML request. Otherwise the presence query has to be split into several requests.

Note: The function ‘GetPresence’ offered in previous products with Web Services is not supported anymore and should be replaced by ‘GetPresenceXML’.

9.4. PresenceDBGet

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

( ) OpenScape Business with UC Suite

Description

This method gets the forwarding target for a given presence state. Please note that this option is only available with UC Smart.

cmd=PresenceDBGet&user=100&state=presencestate

Returns:

```
<?xml version="1.0" encoding="UTF-8"?>
<PRESENCE_DB>
  <PRESENCE>
    <FORWARD></FORWARD>
    <STATE>number</STATE>
  </PRESENCE>
</PRESENCE_DB>
```

9.5. PresenceDBSet

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

( ) OpenScape Business with UC Suite

Description

This method sets the forwarding target for a given presence state. When the presence state is set, the server will automatically set the forwarding. Please note that this option is only available with UC Smart.

Format:
cmd=PresenceDBSet&user=100&state=presencenum&forwarding=number

Returns:

```
<?xml version="1.0" encoding="UTF-8"?>
<PRESENCEDBSET/>
```

10. User Journal

10.1. JournalRead

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

This method reads the call journal of the user.

Format: cmd=JournalRead&user=100&[maxcnt=v|start=u||dir={in|out}|est={yes|no}][&startdtime=YYYY-MM-DD HH:MM:SS][&enddtime=YYYY-MM-DD]

start and maxcnt will work the same way as in the PhBookSearch command.

Response, list:

```
<?xml version="1.0" encoding="UTF-8"?>
<JOURNAL count="x">
  <ITEM>
    <JID></JID> <!-- Journal ID>
    <DIR>in|out</DIR>
    <CALLER>1234567</CALLER>
    <CALLED>12</CALLED>
    <BEGIN>YYYY-MM-DD HH:MM:SS</BEGIN>
    <END>YYYY-MM-DD HH:MM:SS</END>
    <DURATION>seconds</DURATION>
    <EST>yes|no</EST>
    <INTEXT> </INTEXT>
    </ITEM>
</JOURNAL>
```

Name resolution:
Dependent on the software version in the server, two different ways to resolve names are supported. Both are optional, thus it can happen that no name is available at all.
Option 1: JournalRead will try to resolve the called or caller number based on the direction using the Device list, Speed Dials, and the user's Personal Directory. If there is a matching entry – **and only if there is one!** – the <PHENTRY> tag will be included containing the name, the Phonebook ID and the phonebook type where it was found.

Option 2: The name may be given within the <CALLER> and/or <CALLED> tag of an <ITEM>. Example:
<CALLER surname="Smith" firstname="John">103</CALLER>

It is recommended to check first if the <CALLER> / <CALLED> tag contains a name and utilize it when present. Otherwise check if the <PHENTRY> information is available.

In both cases, the algorithm tries to find the best matching number.

It is possible to specify start and/or end date –time. In this case the list will contain only calls that falls into the specified date range. The list is returned in ascending order by date.

Please note that JournalEvent is not supported with UC Suite. In case of UC Suite, please use HookEvent or DevStateChange event as an update trigger for the journal if needed.

10.2. JournalGroupByDate

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

This method reads a list of call journal entries which were created in the specified time frame.

Format: cmd=JournalGroupByDate&user=100&[maxcnt=v|start=u||dir={in|out}|est={yes|no}][&startdtime=YYYY-MM-DD HH:MM:SS&enddtime=YYYY-MM-DD]

start and maxcnt will work the same way as in the PhBookSearch command.

Response, list:

```
<?xml version="1.0" encoding="UTF-8"?>
<JOURNALGROUP count="x">
  <ITEM>
    <DATE>YYYY-MM-DD 00:00:00</DATE>
    <CNT></CNT>          <!—how many calls received on the given day
  </ITEM>
</JOURNALGROUP>
```

It will return a list of dates with the number of calls on those days matching the given constraints. The interface works similar as the phonebook, fragmentation is possible.

Specifying the start and end date is also possible but here both must be specified. Works as in the JournalRead case.

List is returned in ascending order by date.

10.3. JournalFetch

Status: current

Supported platforms: (X) OpenScape Business with UC Smart

(X) OpenScape Business with UC Suite

Description

This method reads one specific call journal entry of the user which is identified by the journal ID (JID).

Format: cmd=JournalFetch&user=100&jid=1234

jid is a valid journal id for the call journal of user 100.

Response, list:

```
<?xml version="1.0" encoding="UTF-8"?>
<JOURNAL count="x">
  <ITEM>
    <JID></JID> <!-- Journal ID>
    <DIR>in|out</DIR>
    <CALLER>1234567</CALLER>
    <CALLED>12</CALLED>
    <BEGIN>YYYY-MM-DD HH:MM:SS</BEGIN>
    <END>YYYY-MM-DD HH:MM:SS</END>
    <DURATION>seconds</DURATION>
    <EST>yes|no</EST>
    <INTEXT> </INTEXT>
    </ITEM>
</JOURNAL>
```

11. Appendix

11.1. Abbreviations

API	Application Programming Interface
CGI	Common Gateway Interface
CSTA	Computer Supported Telecommunications Application
CSP	CSTA Service Provider
DB	Database
DND	Do Not Disturb
WS	Web Services
WSI	Web Services Interface
HFA	HiPath Feature Access
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
IP-address	Internet Protocol address
JRE	Java™ Runtime Environment
LAS	Lightweight Application Server
MB	Megabyte
Mbit/s	Megabit per second
ME	Medium Enterprise
PBX	Private Branch Exchange
OSBiz	OpenScape Business
PWD	Password
RAM	Random Access Memory
SDK	Software Development Kit
SIP	Session Initiation Protocol
SP	Service Pack
V	Version
WBM	Web Based Management
WSI	Web Services Interface

unify
atos collaboration solutions