

# **Brekeke PBX**

**Version 3**

**Web Service Developer's Guide**

**Brekeke Software, Inc.**

Version

Brekeke PBX Version 3 Web Service Developer's Guide  
Revised August 2013

Copyright

This document is copyrighted by Brekeke Software, Inc.

Copyright © 2013 Brekeke Software, Inc.

This document may not be copied, reproduced, reprinted, translated, rewritten or readdressed in whole or part without expressed, written consent from Brekeke Software, Inc.

Disclaimer

Brekeke Software, Inc. reserves the right to change any information found in this document without any written notice to the user.

---

- 1. Purpose ..... 6**
- 2. About Brekeke PBX Web Service..... 6**
- 3. Installation..... 6**
- 4. Configuring the Axis Soap Engine..... 6**
- 5. Web Service Deployment Descriptor (WSDD) ..... 7**
  - 5.1. Brekeke PBX WSDD ..... 7
  - 5.2. Scoped Services ..... 7
- 6. Starting Web Service..... 7**
- 7. Web Service Description Language..... 8**
  - 7.1. Obtaining WSDL from Deployed Service..... 8
  - 7.2. Creating Web Service Clients and Proxy-Generation ..... 8
  - 7.3. Generating the Client Stubs from the WSDL..... 9
  - 7.4. Issues with the Proxy Generation approach ..... 10
- 8. Brekeke PBX Web Service Methods ..... 11**
  - 8.1. applyArsRules (Deprecated since Brekeke PBX v3.1) ..... 11
  - 8.2. callControl ..... 11
  - 8.3. callControl\_mt (Multi-Tenant Function)..... 12
  - 8.4. createNote..... 12
  - 8.5. createNote\_mt (Multi-Tenant Function) ..... 13
  - 8.6. createTenant\_mt (Multi-Tenant Function) ..... 13
  - 8.7. createUser ..... 14
  - 8.8. createUser\_mt (Multi-Tenant Function)..... 14
  - 8.9. deleteMessage..... 15
  - 8.10. deleteMessage\_mt (Multi-Tenant Function) ..... 16
  - 8.11. deletePrompt..... 16
  - 8.12. deletePrompt\_mt (Multi-Tenant Function) ..... 17
  - 8.13. deleteRouteVariables..... 17
  - 8.14. deleteUser ..... 17
  - 8.15. deleteUser\_mt (Multi-Tenant Function)..... 18
  - 8.16. getArsVariables (Deprecated since Brekeke PBX v3.1) ..... 18
  - 8.17. getMessageCnt..... 19
  - 8.18. getMessageCnt\_mt (Multi-Tenant Function) ..... 19
  - 8.19. getMessageXml ..... 20
  - 8.20. getMessageXml\_mt (Multi-Tenant Function)..... 20
  - 8.21. getNotes ..... 21
  - 8.22. getNotes\_mt (Multi-Tenant Function)..... 21
  - 8.23. getNotesList..... 22

8.24.	getNotesList_mt.....	22
8.25.	getPromptXml .....	23
8.26.	getPromptXml_mt (Multi-Tenant Function) .....	23
8.27.	getRouteTemplateNamesOfTenant .....	24
8.28.	getRouteVariables .....	24
8.29.	getRouteVariablesOfTenant (Multi-Tenant Function).....	25
8.30.	getTenantAndUser (Multi-Tenant Function).....	25
8.31.	getTenantProperties_mt (Multi-Tenant Function).....	26
8.32.	getTenantProperty_mt (Multi-Tenant Function) .....	26
8.33.	getUserProperties.....	27
8.34.	getUserProperties_mt (Multi-Tenant Function) .....	27
8.35.	getUserProperty .....	28
8.36.	getUserProperty_mt (Multi-Tenant Function).....	29
8.37.	getUserPropertyNames .....	29
8.38.	getUserPropertyNames_mt (Multi-Tenant Function) .....	30
8.39.	getUserXml_mt (Multi-Tenant Function) .....	30
8.40.	insertRouteVariables .....	31
8.41.	listTenants_mt (Multi-Tenant Function) .....	32
8.42.	listUsers .....	32
8.43.	listUsers_mt (Multi-Tenant Function).....	32
8.44.	setArsVariables (Deprecated since Brekeke PBX v3.1) .....	33
8.45.	setNotes .....	34
8.46.	setNotes_mt (Multi-Tenant Function).....	34
8.47.	setTenantProperties_mt (Multi-Tenant Function) .....	35
8.48.	setTenantProperty_mt (Multi-Tenant Function) .....	35
8.49.	setUserProperties .....	36
8.50.	setUserProperties_mt (Multi-Tenant Function).....	36
8.51.	setUserProperty .....	37
8.52.	setUserProperty_mt (Multi-Tenant Function) .....	37
8.53.	updateRouteVariables.....	38
<b>9.</b>	<b>Extension Settings Properties.....</b>	<b>39</b>
<b>10.</b>	<b>Sample Programs .....</b>	<b>39</b>
10.1.	ArrayGetterClient .....	40
10.2.	ArraySetterClient.....	41
10.3.	GetterClient .....	42
10.4.	SetterClient .....	44
10.5.	CreatorUserClient .....	45
10.6.	ThirdPartyCC.....	46

---

**11. Web Service Security ..... 47**  
**12. Developing with PHP and other languages..... 48**  
    **Sample Message Xml ..... 49**  
**14. Sample Prompt Xml..... 50**  
**15. Sample Users List Xml..... 51**

## 1. Purpose

This document describes the Brekeke PBX Web Service. The Brekeke PBX Web Service allows third party applications to view and modify PBX options and user settings.

For the development of an application that integrates with Brekeke PBX version 3.2.x and later, it is recommended that you use Brekeke PAL WebSocket instead of Brekeke Web Service.

## 2. About Brekeke PBX Web Service

The Brekeke PBX Web Service was implemented using Axis Version 1.3. Axis is essentially a SOAP engine -- a framework for constructing SOAP processors such as clients, servers, gateways, etc. Axis also includes:

- ◆ A server which plugs into servlet engines such as Tomcat,
- ◆ Support for the Web Service Description Language (WSDL),
- ◆ An emitter tooling that generates Java classes from WSDL.

Please view the latest Axis documentation at: <http://ws.apache.org/axis/java/index.html>

## 3. Installation

The files for the Axis Engine and the Brekeke Web Service are automatically installed when the Brekeke PBX software is installed.

## 4. Configuring the Axis Soap Engine

The web.xml file in the /pbx/WEB-INF subdirectory contains information necessary to deploy the Brekeke PBX servlet and the AXIS servlet.

## 5. Web Service Deployment Descriptor (WSDD)

A deployment descriptor contains configuration information for Web Services available through the Axis engine.

### 5.1. Brekeke PBX WSDD

For the Brekeke PBX Web Service, the WSDD file is named "server-config.wsdd" which is located in the /pbx/WEB-INF subdirectory.

The Brekeke PBX Web Service is named "UserImpl" in the WSDD file.

### 5.2. Scoped Services

Axis supports scoping service objects three ways. "Request" scope will create a new object each time a SOAP request comes in for the web service. "Application" scope will create a singleton shared object to service all requests. "Session" scope will create a new object for each session-enabled client who accesses the web service. To specify the scope option, open the WSDD file for editing and change the scope parameter in the "UserImpl" service as follows:

```
<service name="UserImpl"...>
  <parameter name="scope" value="value"/>
  ...
</service>
```

Valid entries for "value" are "request", "session", or "application".

## 6. Starting Web Service

The web service automatically starts when Tomcat is started.

## 7. Web Service Description Language

WSDL describes Web Services in a structured way. A WSDL describes, in a machine-understandable way, the interface to the web service, the data types it uses, and where the web service is located.

### 7.1. Obtaining WSDL from Deployed Service

When you make a service available using Axis, there is typically a unique URL associated with that service. For the Brekeke PBX Web Service, this is usually:

**`http://<host>/pbx/services/UserImpl`**

where <host> is the host server name or host IP address:port.

Attach a "?wsdl" to the end of the URL, Axis will automatically generate a service description for the deployed service, and return it as XML in your browser. Type:

**`http://<host>/pbx/services/UserImpl?wsdl`**

where <host> is the host server name or host IP address:port.

The resulting xml description may be saved or used as input to proxy-generation (Section 7.3).

### 7.2. Creating Web Service Clients and Proxy-Generation

There are several ways to create clients for the web service. One way is to have Axis generate a wrapper class for the web service. This is done by taking the WSDL description of the service and generating Java classes that make the low level calls appropriate to building the SOAP requests for each operation, then post-processing the results into the declared return values. Axis also takes note of any URL of the service included in the WSDL and compiles this in to the classes. Thus the client will automatically bind to the URL that the WSDL talks about - which is often the URL of the (development) server that the WSDL was retrieved from.



---

The steps to create a web service client using proxy generation are:

1. Start the web service by running Tomcat.
2. Generate the WSDL file using the browser. See Section “Obtaining WSDL from Deployed Service”
3. Create the client stubs using the WSDL2Java tool. See Section “Generating the Client Stubs from the WSDL”
4. Compile the generated stub classes.
5. Write the client using the stub classes.

### 7.3. Generating the Client Stubs from the WSDL

You'll find the Axis WSDL-to-Java tool in "org.apache.axis.wsdl.WSDL2Java". The basic invocation form looks like this:

```
java org.apache.axis.wsdl.WSDL2Java userimpl.wsdl
```

Alternatively, you can use the wsdl directly from the web service itself.

```
java org.apache.axis.wsdl.WSDL2Java http://<host>/pbx/services/UserImpl?wsdl
```

where <host> is the host server name or host IP address:port.

This will generate only those bindings necessary for the client. Axis follows the JAX-RPC specification when generating Java client bindings from WSDL.

The WSDL2Java tool will create files in a directory structure that depends on the hostname. For instance, on a host with the default name of “localhost”, the stub or proxy classes were generated in “localhost:<port>\pbx\services\UserImpl” because that is the target namespace from the WSDL and namespaces map to Java packages. Compile the generated proxy classes and they will be ready for use.

## 7.4. Issues with the Proxy Generation approach

This automatic generation of proxy classes is convenient, as it makes calling a remote Web Service look almost like calling a local object. However, the developer should be aware of the following issues:

- ◆ These generated classes are only compatible with Axis. This is allowed by the JAX-RPC specification, which has a notion of compile time compatibility but not run-time compatibility. If you want stub classes that work with other organizations' SOAP implementation, you would need to generate stub classes from the WSDL using their platform's tools. The stub classes should all have the same names and methods, so the rest of the code should not change.
- ◆ Remote Web Services are not the same as local objects. Pretending that they are is going to lead you astray. In particular, a method call to a local object often takes a few microseconds, while a call to a remote service can take tens of seconds, and fail with an obscure network error in the process, leaving the caller unsure if the call was successful or not. Making blocking calls to a Web Service from a web service will lead to a very unhappy end user experience.
- ◆ You have a more complex build process, as you need the WSDL before compiling the client.

## 8. Brekeke PBX Web Service Methods

The Brekeke PBX Web Service makes functions and features of the Brekeke PBX Admintool available to third party applications. There is also a Third Party Call Control function that will allow you to create applications that make calls through the Brekeke PBX Server (Section 8.9). Functions for Multi-Tenant Brekeke PBX have “\_mt” as part of their names.

### 8.1. applyArsRules (Deprecated since Brekeke PBX v3.1)

**Signature:**

```
public void applyArsRules() throws RemoteException,UserImplException;
```

**Description:**

Update and apply all the ARS Rules including newly created or updated rules.

### 8.2. callControl

**Signature:**

```
public String callControl(String user, String from, String[] to, String type) throws  
RemoteException,UserImplException;
```

**Description:**

Allow the creation of third party applications that can create calls through the Brekeke PBX Server

**Parameters:**

user – String representing the call owner. Usually it is a Brekeke PBX user extension number

from – String representing the from-URL

to – String array representing the to-URL

Calls can be simultaneously made to different to-URLs

type – String “1” or string “2”

Type “1” will simultaneously call the from-URL and the to-URL then connect them

Type “2” will call the from-URL first. When the from-URL picks up, the to-URL is called, and the two calls are connected

**Returns:**

Success or Error message

### 8.3. **callControl\_mt (Multi-Tenant Function)**

**Signature:**

```
public String callControl_mt(String tenant, String user, String from, String[] to, String type)
throws RemoteException,UserImplException;
```

**Description:**

Allow third party applications to make calls through the Brekeke PBX Server

**Parameters:**

tenant – the name of a tenant company

user – string representing the call owner. Usually this is an extension number

from – string representing the from-URL

to – an array of strings representing the to-URL

Calls can be simultaneously made to different to-URLs

type – string “1” or string “2”

Type “1” will simultaneously call the from-URL and the to-URL then connect them.

Type “2” will call the from-URL first. When the from-URL picks up, the to-URL is called, and the two calls are connected.

**Returns:**

Success or Error message

### 8.4. **createNote**

**Signature:**

```
public void createNote(String noteName) throws RemoteException,UserImplException;
```

**Description:**

Create a new note in the Brekeke PBX

**Parameters:**

noteName – the name of the new note to be created

**Returns:**

If an error occurs, this method generates either a RemoteException or a UserImplException.

**8.5. createNote\_mt (Multi-Tenant Function)****Signature:**

```
public void createNote_mt (String tenant, String noteName) throws  
RemoteException,UserImplException;
```

**Description:**

Create a new tenant note in the Brekeke PBX

**Parameters:**

tenant – the name of a tenant company

noteName – the name of the new note to be created

**Returns:**

If an error occurs, this method generates either a RemoteException or a UserImplException.

**8.6. createTenant\_mt (Multi-Tenant Function)****Signature:**

```
public boolean createTenant_mt (String tenant) throws RemoteException,UserImplException;
```

**Description:**

Create a new tenant in the Brekeke PBX

**Parameters:**

tenant – the name of a tenant company

**Returns:**

True. If an error occurs, this method generates either a RemoteException or a UserImplException.

**Related methods:** setTenantProperties\_mt, setTenantProperty\_mt

## 8.7. createUser

**Signature:**

```
public boolean createUser(String userName, String password, String loginpw, String type)
throws RemoteException,UserImplException;
```

**Description:**

Create a Brekeke PBX extension

**Parameters:**

userName - a string representing an extension ID as defined in the Brekeke PBX Admintool

password - a string representing the corresponding password for this user voicemail box

loginpw – a string representing the corresponding password for this user account

type – a string representing the extension type, value can be be user, ringgroup, ivr, conditional, conference, callback

If set “type” other than user, do not need to set password and loginpw

**Returns:**

True. If an error occurs, this method generates either a RemoteException or a UserImplException

**Related methods:** setUserProperties, setUserProperty

## 8.8. createUser\_mt (Multi-Tenant Function)

**Signature:**

```
public boolean createUser_mt(String tenant, String userName, String password, String loginpw,
String type) throws RemoteException,UserImplException;
```

**Description:**

Create a Brekeke PBX tenant extension

**Parameters:**

tenant - the name of a tenant company

userName - a string representing an extension name as defined for this tenant

password - a string representing the corresponding password for user voicemail box

loginpw – a string representing the corresponding password for this user account

type – a string representing the user type, value can be user, ringgroup, ivr, conditional, conference, callback

If set “type” value other than user, do not need to set password and loginpw.

**Returns:**

True. If an error occurs, this method generates either a RemoteException or a UserImplException.

**Related methods:** setUserProperties\_mt, setUserProperty\_mt

## 8.9. deleteMessage

**Signature:**

public void deleteMessage(String user, String password, String id) throws

RemoteException,UserImplException;

**Description:**

Delete a voicemail message

**Parameters:**

user – the user extension

password – any string

id – identification string assigned to the voicemail

**Related methods:** getMessageXml

## 8.10. deleteMessage\_mt (Multi-Tenant Function)

**Signature:**

public void deleteMessage\_mt(String tenant, String user, String password, String id) throws  
RemoteException,UserImplException;

**Description:**

Delete a voicemail message

**Parameters:**

tenant – the name of a tenant company

user – the user extension

password – any string

id – identification string assigned to the voicemail

**Related methods:** getMessageXml\_mt

## 8.11. deletePrompt

**Signature:**

public void deletePrompt(String user, String password, String slang, String id) throws  
RemoteException,UserImplException;

**Description:**

Delete an uploaded user prompt file

**Parameters:**

user – the user extension

password – any string

slang – the language code Typically, en = English and ja =Japanese

id – identification string assigned to the voice prompt

**Related methods:** getPromptXml



---

## 8.12. deletePrompt\_mt (Multi-Tenant Function)

**Signature:**

```
public void deletePrompt_mt(String tenant, String user, String password, String slang, String id) throws RemoteException,UserImplException;
```

**Description:**

Delete an uploaded tenant extension prompt file

**Parameters:**

tenant – the name of a tenant company

user – the extension

password – any string

slang – the language code Typically, en = English and ja =Japanese

id – identification string assigned to the voice prompt

**Related methods:** getPromptXml\_mt

## 8.13. deleteRouteVariables

**Signature:**

```
public void deleteRouteVariables(String routeTemplateName, String routeName) throws RemoteException,UserImplException;
```

**Description:**

Delete a route under the defined ARS template and its values of Route Local Variables

**Parameters:**

routeTemplateName – the name of ARS route template name

routeName – the route name

## 8.14. deleteUser

**Signature:**

```
public boolean deleteUser(String userName) throws RemoteException,UserImplException;
```

**Description:**

Delete a Brekeke PBX extension

**Parameters:**

userName - a string representing an extension name

**Returns:**

True. If an error occurs, this method generates either a RemoteException or a UserImplException.

**8.15. deleteUser\_mt (Multi-Tenant Function)****Signature:**

```
public boolean deleteUser_mt(String tenant, String userName) throws  
RemoteException,UserImplException;
```

**Description:**

Delete a Brekek PBX extension from a tenant

**Parameters:**

tenant - the name of a tenant company

userName - a string representing an extension defined under the defined tenant

**Returns:**

True. If an error occurs, this method generates either a RemoteException or a UserImplException.

**8.16. getArsVariables (Deprecated since Brekeke PBX v3.1)****Signature:**

```
public String[][] getArsVariables(String routeName, int regIndex, String regex) throws  
RemoteException,UserImplException;
```

**Description:**

Return a list of ARS route variables matching the Regular Expression (regex) pattern in

column referred to by regIndex

**Parameters:**

routeName – the name of the ARS route containing the variables

regIndex – index of the column that need to perform pattern matching on

regex – the Regular Expression that must be matched

**Returns:**

Return an array of string arrays containing the ARS variables

**Related methods:** setArsVariables, applyArsRules

### 8.17. getMessageCnt

**Signature:**

```
public int getMessageCnt(String user, int type) throws RemoteException,UserImplException;
```

**Description:**

Get the number of voicemail messages

**Parameters:**

user – the user extension

type – integer indicating the type of voicemail where 1 = new, 2 = saved

**Returns:**

Return integer representing the number of voicemail messages in defined type

### 8.18. getMessageCnt\_mt (Multi-Tenant Function)

**Signature:**

```
public int getMessageCnt_mt(String tenant, String user, int type) throws  
RemoteException,UserImplException;
```

**Description:**

Get the number of voicemail messages

**Parameters:**

tenant – the name of a tenant company

user – the user extension

type – integer indicating the type of voicemail where 1= new, 2=saved

**Returns:**

Return integer representing the number of voicemail messages

**8.19. getMessageXml****Signature:**

```
public String getMessageXml(String user, int type, int limit) throws
```

```
RemoteException,UserImplException;
```

**Description:**

Get the users voice mail list

**Parameters:**

user – the user extension

type – integer indicating the type of voicemail; 1 = new messages and details, 2 = saved messages and details

limit – maximum number of messages to retrieve

**Returns:**

Return an xml document containing the voicemail details

**8.20. getMessageXml\_mt (Multi-Tenant Function)****Signature:**

```
public String getMessageXml_mt(String tenant, String user, int type, int limit) throws
```

```
RemoteException,UserImplException;
```

**Description:**

Get the users voice mail list

**Parameters:**

tenant – the name of a tenant company

user – the user extension

type – integer indicating the type of voicemail; 1 = new messages and details, 2 = saved message and details

limit – maximum number of messages to retrieve

**Returns:**

Return an xml document containing the voicemail details

**8.21. getNotes****Signature:**

```
public String[] getNotes(String notesName) throws RemoteException,UserImplException;
```

**Description:**

Retrieve information about a note

**Parameters:**

notesName – the name of the note

**Returns:**

Return the note's description, access level, and content in an array of string

For access level, 0 = no access, 1 = read only, and 2 = read/write

**8.22. getNotes\_mt (Multi-Tenant Function)****Signature:**

```
public String[] getNotes_mt(String tenant, String notesName) throws
```

```
RemoteException,UserImplException;
```

**Description:**

Retrieve information about a note

**Parameters:**

tenant – the name of a tenant company

notesName – the name of the note

**Returns:**

Return the note's description, access level, and content in an array of string

For access level, 0 = no access, 1 = read only, and 2 = read/write

**8.23. getNotesList****Signature:**

```
public String[] getNotesList(boolean isAdmin, int limit) throws
```

```
RemoteException,UserImplException;
```

**Description:**

Return a list of note names

**Parameters:**

isAdmin – Boolean indicating whether the search should be done as an administrator or a regular user

limit – Restrict the number of note names returned. For unlimited, set to -1.

**Returns:**

Return an array of string containing the list of note names

**8.24. getNotesList\_mt****Signature:**

```
public String[] getNotesList_mt(String tenant, boolean isAdmin, int limit) throws
```

```
RemoteException,UserImplException;
```

**Description:**

Return a list of note names

**Parameters:**

tenant – the name of a tenant company

isAdmin – Boolean indicating whether the search should be done as an administrator or regular user

limit – restricts the number of notes names that are returned. For unlimited, set to -1.

**Returns:**

Return an array of string containing the list of note names

**8.25. getPromptXml****Signature:**

```
public String getPromptXml(String user, String slang, int type, int limit) throws  
RemoteException,UserImplException;
```

**Description:**

Retrieve a list of uploaded prompt files

**Parameters:**

user – the extension

slang – the language code. Typically, en= English and ja=Japanese.

type – integer representing the group of prompts. 1=standard

limit – limits the number of records returned. Set to -1 for unlimited

**Returns:**

Return an xml string containing the list of user uploaded prompts

When set type as 2 but not set user, the method return system prompt xml list

**8.26. getPromptXml\_mt (Multi-Tenant Function)****Signature:**

```
public String getPromptXml_mt(String tenant, String user, String slang, int type, int limit) throws  
RemoteException,UserImplException;
```

**Description:**

Retrieve a list of tenant extension uploaded prompt files

**Parameters:**

tenant – the name of a tenant company

user – the user extension

slang – output file description language code. Typically, en= English and ja=Japanese.

type – integer representing the group of prompts. 1=standard

limit – limits the number of records returned. For unlimited, set to -1.

**Returns:**

Return an xml string containing the list of user uploaded prompts

**8.27. getRouteTemplateNameOfTenant****Signature:**

```
public String getRouteTemplateNameOfTenant(String tenant) throws  
RemoteException, UserImplException;
```

**Description:**

Retrieve a list of tenant route template names

**Parameters:**

tenant – the name of a tenant company

**Returns:**

An array of string containing the route template names which the tenant is assigned to and the route templates' variables' [Tenant (List)] fields are selected in [Route Local Setting] > [Field settings] page and

**8.28. getRouteVariables****Signature:**

```
public String[][] getRouteVariables( String routeTemplateName ) throws
```



RemoteException,UserImplException;

**Description:**

Retrieve the route name and its values of Route Local Variables created under the defined route template name

**Parameters:**

routeTemplateName - the name of ARS route template

**Returns:**

Two-dimension array of string containing the route names and values of Route Local Variables

### 8.29. **getRouteVariablesOfTenant (Multi-Tenant Function)**

**Signature:**

```
public String[][] getRouteVariablesOfTenant( String routeTemplateName, String tenant ) throws  
RemoteException,UserImplException;
```

**Description:**

Retrieve the route name and its values of Route Local Variables assigned to the defined tenant and created under the defined route template name

**Parameters:**

routeTemplateName - the name of ARS route template

tenant – the name of a tenant company

**Returns:**

Two-dimension array of string containing the route names and values of Route Local Variables

### 8.30. **getTenantAndUser (Multi-Tenant Function)**

**Signature:**

```
public String getTenantAndUser( String pnumber ) throws  
RemoteException,UserImplException;
```

**Description:**

Return the tenant name and user name associated with the assigned phone ID (pnumber)

**Parameters:**

pnumber – the assigned phone ID

**Returns:**

A colon string containing the tenant and user names in the format: <tenant>:<user>

**8.31. getTenantProperties\_mt (Multi-Tenant Function)****Signature:**

```
public String[] getTenantProperties_mt (String tenant, String[] propertyNames) throws  
RemoteException,UserImplException;
```

**Description:**

Return the list of tenant properties specified in the propertyNames array

**Parameters:**

tenant – the name of a tenant company

propertyNames – an array of string containing the property values to retrieve

**Returns:**

An array of string containing the result

**Notes:**

Valid property Names are: tenantid, desc, maxrecordingsessions, maxusers, and maxsessions

**8.32. getTenantProperty\_mt (Multi-Tenant Function)****Signature:**

```
public String getTenantProperty_mt (String tenant, String propertyName) throws  
RemoteException,UserImplException;
```

**Description:**

Return the tenant property value specified in the propertyName

**Parameters:**

tenant – the name of a tenant company

propertyName – string containing the property to retrieve

**Returns:**

A string containing the corresponding property value

**Notes:**

Valid property Names are: tenantid, desc, maxrecordingsessions, maxusers, and maxsessions

### 8.33. getUserProperties

**Signature:**

```
public String[] getUserProperties(String userName, String[] propertyNameArray) throws  
RemoteException,UserImplException;
```

**Description:**

Retrieve the value of the properties specified in the propertyNameArray parameter

**Parameter:**

userName - a string representing an extension

propertyNameArray - an array of string in property names whose values are to be retrieved

**Returns:**

An array of string containing properties' value

If an error occurs, this method generates either a RemoteException or a UserImplException

**Related methods:** getUserPropertyNames

### 8.34. getUserProperties\_mt (Multi-Tenant Function)

**Signature:**

```
public String[] getUserProperties_mt(String tenant, String userName, String[]  
propertyNameArray) throws RemoteException,UserImplException;
```

**Description:**

Retrieve the property values of the properties specified in the propertyNameArray parameter

**Parameter:**

tenant - the name of the tenant company

userName - a string representing an extension

propertyNameArray - an array of string defining property names whose values are to be retrieved

**Returns:**

An array of string containing property values

If an error occurs, this method generates either a RemoteException or a UserImplException.

**Related methods:** getUserPropertyNames\_mt

### 8.35. **getUserProperty**

**Signature:**

```
public String getUserProperty(String userName, String propertyName) throws  
RemoteException, UserImplException;
```

**Description:**

Retrieve a property value

**Parameters:**

userName - a string representing an extension

propertyName - a string containing a property name

**Returns:**

A string containing the property value

If an error occurs, this method generates either a RemoteException or a UserImplException.

**Related methods:** getUserPropertyNames

### 8.36. `getUserProperty_mt` (Multi-Tenant Function)

**Signature:**

```
public String getUserProperty_mt(String tenant, String userName, String propertyName)
```

throws `RemoteException`, `UserImplException`;

**Description:**

Retrieve a property value

**Parameters:**

tenant - the name of the tenant company

userName - a string representing an extension

propertyName - a string containing a property name

**Returns:**

A string containing the property value

If an error occurs, this method generates either a `RemoteException` or a `UserImplException`.

**Related methods:** `getUserPropertyNames_mt`

### 8.37. `getUserPropertyNames`

**Signature:**

```
public String[] getUserPropertyNames(String userName) throws
```

`RemoteException`, `UserImplException`;

**Description:**

Retrieve the names of properties that are currently saved in the user's property file

**Parameter:**

userName - an extension name

**Returns:**

An array of string containing the property names

If an error occurs, this method generates either a `RemoteException` or a `UserImplException`.

### 8.38. `getUserPropertyNames_mt` (Multi-Tenant Function)

**Signature:**

```
public String[] getUserPropertyNames_mt(String tenant, String userName) throws  
RemoteException,UserImplException;
```

**Description:**

Retrieve the names of properties that are currently saved in the user's property file

**Parameter:**

tenant - the name of the tenant company

userName - an extension

**Returns:**

An array of string containing the property names

If an error occurs, this method generates either a RemoteException or a UserImplException.

### 8.39. `getUserXml_mt` (Multi-Tenant Function)

**Signature:**

```
public String getUserXml_mt(String tenant, String filterString, String filterType, int nStartIndex,  
int limit) throws RemoteException,UserImplException;
```

**Description:**

Get the list of users' information

**Parameters:**

tenant – the name of a tenant company

filterString – a regular expression

filterType – 10 = contains, 20 = begins with, 30 = ends with, default is exact match

nStartIndex – the index within the result set from which to begin the output

limit – maximum number of users to retrieve

**Returns:**

Return an xml document containing the users' details

**8.40. insertRouteVariables****Signature:**

```
public boolean insertRouteVariables( String routeTemplateName, String[] vals ) throws  
RemoteException,UserImplException;
```

**Description:**

Add a new route and its values of Route Local Variables to specified route template

**Parameters:**

routeTemplateName – the name of an ARS route template

vals – an array of string containing route name and related Route Local Variable setting

val 0 – 1:Enabled, 0:Disabled

val 1 - The tenant name

val 2 - The route name

val 3 - password

val 4~5 - <reserved>

val 6~14 - v1~v9

val 15~23 - w1~w9

val 24~32 - x1~x9

val 0 to val 6 are required variables for this method; set empty string for the variable that is not needed

**Returns:**

True if the route is create successfully

False if an entry of the same route name already exists

### 8.41. listTenants\_mt (Multi-Tenant Function)

**Signature:**

```
public String[] listTenants_mt (int limit) throws RemoteException,UserImplException;
```

**Description:**

Return a list of Brekeke PBX Tenants

**Parameters:**

limit – restricts the number of tenants that are returned. For unlimited, set to -1.

**Returns:**

An array of string containing the tenant names

### 8.42. listUsers

**Signature:**

```
public String[] listUsers(String regExFilter, int limit) throws  
RemoteException,UserImplException;
```

**Description:**

Return a list of extensions matching the regular expression filter

**Parameters:**

regExFilter – a regular expression that the result must match

limit – restrict the number of users returned. For unlimited, set to -1.

**Returns:**

An array of string containing the result

### 8.43. listUsers\_mt (Multi-Tenant Function)

**Signature:**

```
public String[] listUsers_mt(String tenant, String regExFilter, int limit) throws  
RemoteException,UserImplException;
```



**Description:**

Return a list of extensions in a particular tenant that matches the regular expression filter

**Parameters:**

tenant – the name of a tenant company

regExFilter – a regular expression that the result must match

limit – restricts the number of tenants that are returned. For unlimited, set to -1.

**Returns:**

An array of string containing the result

**8.44. setArsVariables (Deprecated since Brekeke PBX v3.1)****Signature:**

```
public void setArsVariables(String routeName, int regIndex, String regex, String[][] varValues)
throws RemoteException,UserImplException;
```

**Description:**

Set the values for ARS route variables. This is performed in two steps:

Delete the rows of values matching the Regular Expression (regex) pattern in column referred to by regIndex.

Add the row of values specified in varValues.

**Parameters:**

routeName – the name of the ARS route containing the variables

regIndex – index of the column that we want to perform pattern matching on

regex – the Regular Expression that must be matched

varValues – two dimensional string array containing the ARS variable values

**Related methods:** getArsVariables, applyArsRules

---

## 8.45. setNotes

**Signature:**

public void setNotes(String[] notes) throws RemoteException,UserImplException;

**Description:**

Set information for a note, and create system note if the method is used in multi-tenant Brekeke PBX

**Parameters:**

notes – an array of string containing information about the notes

Element 0 contains the note name

Element 1 is note description

Element 2 is note access level

Element 3 is notes content

## 8.46. setNotes\_mt (Multi-Tenant Function)

**Signature:**

public void setNotes\_mt(String tenant, String[] notes) throws  
RemoteException,UserImplException;

**Description:**

Set information for a note in the defined tenant

**Parameters:**

Tenant – the name of a tenant company

notes – an array of string containing information about the notes

Element 0: note name

Element 1: description

Element 2: access level; No access: 0; Read only: 1; Read/Write: 2

Element 3: note content

---

### 8.47. setTenantProperties\_mt (Multi-Tenant Function)

**Signature:**

```
public void setTenantProperties_mt (String tenant, String[] propertyNames, String[]  
propertyValues) throws RemoteException,UserImplException;
```

**Description:**

Set the values for properties specified in the propertyNames array to the values specified in the propertyValues array

**Parameters:**

tenant – the name of a tenant company

propertyNames – the names of the tenant properties to be changed (See Notes below)

propertyValues – the values corresponding to the properties specified in propertyNames

**Notes:**

Valid property names are: tenantid, desc, maxrecordingsessions, maxusers, and maxsessions

### 8.48. setTenantProperty\_mt (Multi-Tenant Function)

**Signature:**

```
public void setTenantProperty_mt (String tenant, String propertyName, String propertyValue)  
throws RemoteException,UserImplException;
```

**Description:**

Set the value for property specified in the propertyName to the value specified in the propertyValue

**Parameters:**

tenant – the name of a tenant company

propertyName – the name of the tenant property to be changed (See Notes below)

propertyValue – the value corresponding to the property specified in propertyName

**Notes:**

Valid property names are: tenantid, desc, maxrecordingsessions, maxusers, and maxsessions

**8.49. setUserProperties****Signature:**

```
public void setUserProperties(String userName, String[] propertyNameArray, String[]  
propertyValueArray) throws RemoteException,UserImplException;
```

**Description:**

Set the properties listed in the propertyNameArray to the corresponding values listed in the  
propertyValuesArray

**Parameters:**

userName - a string representing a username as defined in the Brekeke PBX Admin

propertyNameArray - a string array of property names whose values are to be set

propertyValueArray- a string array containing values to be set

**Related methods:** getUserPropertyNames

**8.50. setUserProperties\_mt (Multi-Tenant Function)****Signature:**

```
public void setUserProperties_mt(String tenant, String userName, String[] propertyNameArray,  
String[] propertyValueArray) throws RemoteException,UserImplException;
```

**Description:**

Set the properties listed in the propertyNameArray to the corresponding values listed in the  
propertyValueArray

**Parameters:**

Tenant - the name of the tenant company

username - a string representing a username as defined in the Brekeke PBX Admin

propertyNameArray - a string array of property names whose values are to be set

propertyValueArray - a string array containing values to be set

**Related methods:** getUserPropertyNames\_mt

### 8.51. setUserProperty

**Signature:**

```
public void setUserProperty(String userName, String propertyName, String value) throws  
RemoteException,UserImplException;
```

**Description:**

Set the property listed in the propertyName parameter to the corresponding value in the propertyValue parameter

**Parameters:**

userName - a string representing a username as defined in the Brekeke PBX Admin

propertyName - a string containing a property name

value - a string containing a property value

**Related methods:** getUserPropertyNames

### 8.52. setUserProperty\_mt (Multi-Tenant Function)

**Signature:**

```
public void setUserProperty_mt(String tenant, String userName, String propertyName, String  
value) throws RemoteException,UserImplException;
```

**Description:**

Set the property listed in the propertyName parameter to the corresponding value in the value parameter

**Parameters:**

tenant - the name of the tenant company

userName - a string representing a username as defined in the Brekeke PBX Admin

propertyName - a string containing a property name

---

values - a string containing a property value

**Related methods:** getUserPropertyNames

### 8.53. updateRouteVariables

**Signature:**

```
public boolean updateRouteVariables( String routeTemplateName, boolean updatePassword,  
String[] vals ) throws RemoteException,UserImplException;
```

**Description:**

Set a route and its values of Route Local Variables to specified route template

**Parameters:**

routeTemplateName – the name of an ARS route template

vals – an array of string containing route name and related Route Local Variable setting

val 0 – 1:Enabled, 0:Disabled

val 1 - The tenant name

val 2 - The route name

val 3 - password

val 4~5 - <reserved>

val 6~14 - v1~v9

val 15~23 - w1~w9

val 24~32 - x1~x9

val 0 to val 6 are required variables for this method; set empty string for the variable that are not needed

**Returns:**

True if the route is modified successfully

False if an entry of the same route name already exists

## 9. Extension Settings Properties

The tables in the “Brekeke PAL WebSocket Developer’s Guide” contain property names that may be viewed or altered using the web service methods.

## 10. Sample Programs

Sample Client Applications can be downloaded from the Brekeke website. The example programs all use the Proxy-generation approach to create the client applications. There are many other methods to create client applications.

## 10.1. ArrayGetterClient

This is sample program demonstrating how to retrieve property values using a string array of property names.

```
import localhost.sipadmin.services.UserImpl.*;

public class ArrayGetterClient {
    public static void main(String[] args) {
        try {
            String[] propNameArray = { "name", "password", "email",
                "ringertime", "userplugin", "rtprelay", "admin", "demo",
                "ptn.index", "index" };
            //fill in an array of properties to retrieve
            String[] propValueArray;
            UserImplService afs = new
                UserImplServiceLocator("http://<Brekeke_PBX_IP>:<port>/pb
                x/services/UserImpl");          //webservice URL here
            UserImpl af = afs.getUserImpl();
            propValueArray = af.getUserProperties("test",
propNameArray);
            //retrieve the properties
            // results are returned as a String Array
            // loop through and display the contents of the String
Array
            for (int i = 0; i < propNameArray.length; i++) {
                System.out.println(propNameArray[i] + "=" +
                    propValueArray[i]);
            }
        } catch (Exception e) {
            System.err.println("Execution failed. Exception: " + e);
        }
    }
}
```



## 10.2. ArraySetterClient

This is a sample program demonstrating how to set property values using string arrays of property names and property values.

```
import localhost.sipadmin.services.UserImpl.*;

public class ArraySetterClient {
    public static void main(String[] args) {
        try {
            String[] propNameArray = { "name", "email" };
            //fill in the an array of property names
            String[] propValueArray = { "Bob Dole1", bob1@senator.org
            };
            //fill in an array of corresponding values
            UserImplService afs = new
            UserImplServiceLocator("http://<Brekeke_PBX_IP>:<port>/pb
            x/services/UserImpl");
            //webservice URL here

            UserImpl af = afs.getUserImpl();

            af.setUserProperties("test", propNameArray,
            propValueArray); //sample method call

            System.out.println("Ok");

        } catch (Exception e) {
            System.err.println("Execution failed. Exception: " + e);
        }
    }
}
```

### 10.3. GetterClient

This is a sample program demonstrating how to retrieve a single property value.

```
import localhost.sipadmin.services.UserImpl.*;
import org.apache.axis.utils.Options;

public class GetterClient {
    public static void main(String[] args) {
        try {
            Options options = new Options(args);
            String userName = "test";
            String propName = "";
            args = options.getRemainingArgs();

            if ((args == null) || (args.length < 2)) {
                System.out.println("GetterClient <User name> <property
                    name>");
                System.out.println("if <property name> is set to 'all',
                    this will show all property names and values.");
                return;
            } else {
                userName = args[0];
                propName = args[1];
            }

            UserImplService afs = new
            UserImplServiceLocator("http://<Brekeke_PBX_IP>:<port>/pbx/
            services/UserImpl");
            //webservice URL here

            UserImpl af = afs.getUserImpl();

            if (propName.equalsIgnoreCase("all")) {
                String[] propNameArray =
                af.getUserPropertyNames(userName); //example method call
                for (int i = 0; i < propNameArray.length; i++) {
```

```
        //example method call
        for (int i = 0; i < propNameArray.length; i++) {
            System.out.println(propNameArray[i] + "="
                + af.getUserProperty(userName,
propNameArray[i]));
            //example method call
        }
    } else {
        System.out.println(propName + "=" +
af.getUserProperty(userName, propName));
        //example method call
    }
} catch (Exception e) {
    System.err.println("Execution failed. Exception: " + e);
}
}
}
```

## 10.4. SetterClient

This is a sample program demonstrating how to set a single property value.

```
import localhost.sipadmin.services.UserImpl.*;
import org.apache.axis.utils.Options;

public class SetterClient {
    public static void main(String[] args) {
        try {
            Options options = new Options(args);
            String userName = "test";
            String propName = "";
            String propValue = "";
            args = options.getRemainingArgs();
            if ((args == null) || (args.length < 3)) {
                System.out.println("Usage: SetterClient <username>
                <property name> <property value>");
                return;
            } else {
                userName = args[0];
                propName = args[1];
                propValue = args[2];
            }
            UserImplService afs = new
            UserImplServiceLocator("http://<Brekeke_PBX_IP>:<port>/pbx/
            services/UserImpl");
            //webservice URL here
            UserImpl af = afs.getUserImpl();
            af.setUserProperty(userName, propName, propValue);
            System.out.println("Ok");
        } catch (Exception e) {
            System.err.println("Execution failed. Exception: " + e);
        }
    }
}
```

## 10.5. CreaterUserClient

This is a sample program demonstrating how to create a PBX user.

```
import localhost.sipadmin.services.UserImpl.*;
import org.apache.axis.utils.Options;

public class CreateUserClient {
    public static void main(String[] args) {
        try {
            Options options = new Options(args);
            String userName = "test";
            String password = "test";
            args = options.getRemainingArgs();
            if ((args == null) || (args.length < 2)) {
                System.out.println("Usage: CreateUserClient
                <username> <password>");
                return;
            } else {
                userName = args[0];
                password = args[1];
            }
            UserImplService afs = new
            UserImplServiceLocator("http://<Brekeke_PBX_IP>:<port>/pbx/
            services/UserImpl");
            //webservice URL here
            UserImpl af = afs.getUserImpl();
            if ( af.createUser(userName, password) ){
                System.out.println("Ok");
            }
        } catch (Exception e) {
            System.err.println("Execution failed. Exception: " + e);
        }
    }
}
```

## 10.6. ThirdPartyCC

This is a sample program demonstrating how to create a third party call control application.

```
import localhost.sipadmin.services.UserImpl.*;
import org.apache.axis.utils.Options;

public class ThirdPartyCC {
    public static void main(String[] args) {
        try {
            //fill in the an array of destination numbers
            String[] destination = { "556", "557" };

            //webservice URL here
            UserImplService afs = new
            UserImplServiceLocator("http://<Brekeke_PBX_IP>:<port>/pbx/
            services/UserImpl");

            UserImpl af = afs.getUserImpl();
            System.out.println(af.callControl("555","555",destination,"
            2" ));

        } catch (Exception e) {
            System.err.println("Execution failed. Exception: " + e);
        }
    }
}
```

## 11. Web Service Security

Access to the Web Service is controlled using the PBX Administration Tool. Only Clients with IP Addresses that match the regular expression defined in the PBX Administration Tool will be allowed to consume from the web service.

To define the valid client ip addresses:

1. Browse to the PBX Administration tool.
2. Login with Admin privileges.
3. Select Options from the menu.
4. Enter a regular expression for the text box labeled "Valid Client IP Pattern". For example, "192.168.\*" or "192.168.0.5".

## 12. Developing with PHP and other languages

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. WSDL is the standard language in which you describe Web services. It is an XML-based language that allows you to map a particular service in a language-neutral manner. It is therefore possible to access the Brekeke PBX web service using any programming language that has facilities to use the SOAP and WSDL standards.

In PHP 5, for instance, one can enable the soap extension and create the following application:

```
<?php
$client= new SoapClient( "http://<host>/pbx/services/UserImpl?wsdl" );
$array = array("4007");
print($client->callControl("555", "555", $array, "2"));
?>
```

<host> = the ip address and port address where pbx resides

Please consult with the documents for PHP on how to enable the soap extension.



## 13. Sample Message Xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<message_list>

<user>101</user>

<new_messages>10</new_messages>

<message_item>

<id>1235435612132</id>

<time>Wed, Dec.31, 04:00 PM</time>

<from>103</from>

<bytes>3040</bytes>

</message_item>

<message_item>

<id>1235436084027</id>

<time>Wed, Dec.31, 04:00 PM</time>

<from>103</from>

<bytes>2560</bytes>

</message_item>

<message_item>

<id>1235505000526</id>

<time>Wed, Dec.31, 04:00 PM</time>

<from>103</from>

<bytes>2880</bytes>

</message_item>

</message_list>
```

## 14. Sample Prompt Xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<prompt_list>
<prompt>
<type>user</type>
<index>1</index>
<filename>default</filename>
<description>Voicemail personal greeting</description>
<bytes>19648</bytes>
</prompt>
<prompt>
<type>user</type>
<index>2</index>
<filename>default</filename>
<description>Voicemail alternative greeting</description>
<bytes>0</bytes>
</prompt>
<prompt>
<type>user</type>
<index>3</index>
<filename>default</filename>
<description>Name</description>
<bytes>19648</bytes>
</prompt></prompt_list>
```

## 15. Sample Users List Xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<user_list>
<user>
<ext>101</ext>
<description>Billy Bob</description>
<type>User</type>
<assigned>abba101</assigned>
</user>
<user>
<ext>102</ext>
<description>Betty Sue</description>
<type>User</type>
<assigned>abba102</assigned>
</user>
<user>
<ext>103</ext>
<description>Andrew Jackson</description>
<type>User</type>
<assigned>abba103</assigned>
</user>
</user_list>
```