Brekeke PBX

Version 2

Web Service Developer’s Guide

Brekeke Software, Inc.

Version

Brekeke PBX Version 2 Web Service Developer's Guide

Revised February 2010

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 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 (with default password) 14
8.9. createUser_mt (Multi-Tenant Function) 15
8.10. createUser_mt (with default password) (Multi-Tenant Function) 15
8.11. deleteMessage 16
8.12. deleteMessage_mt (Multi-Tenant Function) 16
8.13. deletePrompt 17
8.14. deletePrompt_mt (Multi-Tenant Function) 17
8.15. deleteUser 18
8.16. deleteUser_mt (Multi-Tenant Function) 18
8.17. getArsVariables 19
8.18. geMessageCnt 19
8.19. getMessageCnt_mt (Multi-Tenant Function) 20
8.20. getMessageXml 20
8.21. getMessageXml_mt (Multi-Tenant Function) 21
8.22. getNotes 21

8.23. getNotes_mt (Multi-Tenant Function) 22
8.24. getNotesList 22
8.25. getNotesList_mt 23
8.26. getPromptXml 23
8.27. getPromptXml_mt (Multi-Tenant Function) 24
8.28. getTenantAndUser (Multi-Tenant Function) 24
8.29. getTenantProperties_mt (Multi-Tenant Function) 25
8.30. getTenantProperty_mt (Multi-Tenant Function) 25
8.31. getUserProperties 26
8.32. getUserProperties_mt (Multi-Tenant Function) 26
8.33. getUserProperty 27
8.34. getUserProperty_mt (Multi-Tenant Function) 27
8.35. getUserPropertyNames 28
8.36. getUserPropertyNames_mt (Multi-Tenant Function) 28
8.37. getUserXml_mt (Multi-Tenant Function) 29
8.38. listTenants_mt (Multi-Tenant Function) 29
8.39. listUsers 30
8.40. listUsers_mt (Multi-Tenant Function) 30
8.41. setArsVariables 31
8.42. setNotes 31
8.43. setNotes_mt (Multi-Tenant Function) 32
8.44. setTenantProperties_mt (Multi-Tenant Function) 32
8.45. setTenantProperty_mt (Multi-Tenant Function) 33
8.46. setUserProperties 33
8.47. setUserProperties mt (Multi-Tenant Function) 34
8.48. setUserProperty 35
8.49. setUserProperty_mt (Multi-Tenant Function) 35
9. User Settings Properties 36
10. Sample Programs 38
10.1. ArrayGetterClient 38
10.2. ArraySetterClient 39
10.3. GetterClient 40
10.4. SetterClient 42
10.5. CreaterUserClient 43
10.6. ThirdPartyCC 44
11. Web Service Security 45

12. Developing with PHP and other languages 46
13. Sample Message Xml.. 47
14. Sample Prompt Xml.. 48
15. Sample Users List Xml 49

1. Purpose

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

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.

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.

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 7.1.
3. Create the client stubs using the WSDL2Java tool. See Section 7.3
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.

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\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 Admin tool 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 PBX Server (See Section 8.9).

Multi-Tenant versions of functions have “_mt” as part of their names.

8.1. applyArsRules

Signature:

public void applyArsRules() throws RemoteException,UserImplException;

Description:

Re-applies 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:

Allows the creation of third party applications that can create calls through the PBX Server.

Parameters:

users – String representing the call owner. Usually this is an 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, then 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:

Allows the creation of third party applications that can create calls through the PBX Server.

Parameters:

tenant is the name of the tenant company

users – String representing the call owner. Usually this is an 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, then the two calls are connected.

Returns: Success or Error message.

8.4. createNote

Signature:

public boolean createNote(String notesName) throws RemoteException,UserImplException;

Description:

Creates a new note in the PBX.

Parameters:

tenant – the name of a tenant company.

notesName – the name of the new notes to be created

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

8.5. createNote_mt (Multi-Tenant Function)

Signature:

public boolean createNote_mt (String tenant, String notesName) throws

RemoteException,UserImplException;

Description:

Creates a new tenant notes in the PBX.

Parameters:

tenant – the name of a tenant company.

notesName – the name of the new notes to be created

Returns: True. 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:

Creates a new tenant in the PBX.

Parameters:

tenant – the name of a tenant company.

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

Notes:

Use setTenantProperties or setTenantProperty after creating the tenant.

8.7. createUser

Signature:

public boolean createUser(String userName, String password) throws

RemoteException,UserImplException;

Description:

Creates a PBX User (extension).

Parameters:

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

password - a string representing the corresponding password.

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

8.8. createUser (with default password)

Signature:

public boolean createUser(String userName) throws RemoteException,UserImplException;

Description:

Creates a PBX User (extension).

Parameters:

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

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

8.9. createUser_mt (Multi-Tenant Function)

Signature:

public boolean createUser_mt(String tenant, String userName, String password) throws

RemoteException,UserImplException;

Description:

Creates a PBX User (extension).

Parameters:

tenant is the name of the tenant company

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

password - a string representing the corresponding password.

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

8.10. createUser_mt (with default password) (Multi-Tenant Function)

Signature:

public boolean createUser_mt(String tenant, String userName) throws

RemoteException,UserImplException;

Description:

Creates a PBX User (extension).

Parameters:

tenant is the name of the tenant company

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

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

8.11. 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 – the users password

id – identification string assigned to the voicemail. This value is returned in the xml from the getMessageXml_mt() function.

Returns: None.

8.12. 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 – the users password

id – identification string assigned to the voicemail. This value is returned in the xml from the getMessageXml_mt() function.

Returns: None.

8.13. deletePrompt

Signature:

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

Description: Delete an uploaded prompt file.

Parameters:

user – the user extension

password – the users password

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

id – identification string assigned to the voicemail. This value is returned in the xml from the getPromptXml() function.

Returns: None.

8.14. 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 prompt file.

Parameters:

tenant – the name of a tenant company

user – the user extension

password – the users password

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

id – identification string assigned to the voicemail. This value is returned in the xml from the getPromptXml_mt() function.

Returns: None.

8.15. deleteUser

Signature:

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

Description:

Deletes a PBX User (extension).

Parameters:

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

Returns:

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

8.16. deleteUser_mt (Multi-Tenant Function)

Signature:

public boolean deleteUser_mt(String tenant, String userName) throws

RemoteException,UserImplException;

Description:

Deletes a PBX User (extension).

Parameters:

tenant is the name of the tenant company

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

Returns:

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

8.17. getArsVariables

Signature:

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

Description:

Returns 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 we want to perform pattern matching on

regex – the Regular Expression that must be matched

Returns:

Returns an array of string arrays containing the ARS variables.

Related functions: setArsVariables, applyArsRules

8.18. geMessageCnt

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, 3=new and saved

Returns:

Return integer representing the number of voicemail messages.

8.19. 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, 3=new and saved

Returns:

Return integer representing the number of voicemail messages.

8.20. 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 – 0= new messages, 1= saved messages

limit – maximum number of messages to retrieve

Returns:

Return an xml document containing the voicemail details.

8.21. 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 – 0= new messages, 1= saved messages

limit – maximum number of messages to retrieve

Returns:

Return an xml document containing the voicemail details.

8.22. getNotes

Signature:

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

Description:

Retrieve information about a note.

Parameters:

noteName – the name of the note

Returns:

Returns the notes description, access level, and notes content in a String array. Element 0 contains the description, 1 the access level, and 2 the contents.

8.23. 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

noteName – the name of the note

Returns:

Returns the notes description, access level, and notes content in a String array. Element 0 contains the description, 1 the access level, and 2 the contents.

8.24. getNotesList

Signature:

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

RemoteException,UserImplException;

Description:

Returns a list of note names.

Parameters:

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:

Returns a string array containing the list of note names.

8.25. getNotesList_mt

Signature:

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

RemoteException,UserImplException;

Description:

Returns a list of note names.

Parameters:

tenant – then 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:

Returns a string array containing the list of note names.

8.26. getPromptXml

Signature:

public String getPromptXml(String user, String lang, int type, int limit) throws

RemoteException,UserImplException;

Description:

Retrieve a list of uploaded prompt files

Parameters:

user – the user extension.

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

type – integer representing the group of prompts. 1=standard, 2 = custom

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

Returns: Returns an xml string containing the list of user uploaded prompts.

8.27. 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 uploaded prompt files

Parameters:

tenant – the name of a tenant company

user – the user extension.

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

type – integer representing the group of prompts. 1=standard, 2 = custom

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

Returns:

Returns an xml string containing the list of user uploaded prompts.

8.28. 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 then tenant and user names in the format: <tenant>:<user>.

8.29. 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 – string array containing the property values to retrieve.

Returns: A string array containing the result.

Notes:

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

8.30. 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.

8.31. getUserProperties

Signature:

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

Description:

Retrieves the Property values of properties specified in the propertyNameArray parameter.

Parameter:

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 retrieved.

Returns:

String array containing property values. If an error occurs, this method generates either a RemoteException or a UserImplException.

8.32. getUserProperties_mt (Multi-Tenant Function)

Signature:

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

Description:

Retrieves the Property values of properties specified in the propertyNameArray parameter.

Parameter:

tenant is 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 retrieved.

Returns:

String array containing property values. If an error occurs, this method generates either a RemoteException or a UserImplException.

8.33. getUserProperty

Signature:

public String getUserProperty(String userName, String propertyName) throws

RemoteException,UserImplException;

Description:

Retrieves a property value.

Parameters:

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

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.

8.34. getUserProperty_mt (Multi-Tenant Function)

Signature:

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

throws RemoteException,UserImplException;

Description:

Retrieves a property value.

Parameters:

tenant is 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.

Returns:

A string containing the property value. If an error occurs, this method generates either a RemoteException or a UserImplException.

8.35. getUserPropertyNames

Signature:

public String[] getUserPropertyNames(String userName) throws

RemoteException,UserImplException;

Description:

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

Parameter:

username is a username as defined in the Brekeke PBX Admin.

Returns:

String Array containing the property names. If an error occurs, this method generates either a RemoteException or a UserImplException.

8.36. getUserPropertyNames_mt (Multi-Tenant Function)

Signature:

public String[] getUserPropertyNames_mt(String tenant, String userName) throws

RemoteException,UserImplException;

Description:

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

Parameter:

tenant is the name of the tenant company

username is a username as defined in the Brekeke PBX Admin.

Returns:

String Array containing the property names. If an error occurs, this method generates either a RemoteException or a UserImplException.

8.37. 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.

8.38. listTenants_mt (Multi-Tenant Function)

Signature:

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

Description:

Return the list of all PBX Tenants

Parameters:

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

Returns:

A string array containing the result.

8.39. listUsers

Signature:

public String[] listUsers(String regExFilter, int limit) throws

RemoteException,UserImplException;

Description:

Return the list of all PBX Users (extension) matching the regular expression filter.

Parameters:

regExFilter – a regular expression that the result must match.

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

Returns:

A string array containing the result.

8.40. listUsers_mt (Multi-Tenant Function)

Signature:

public String[] listUsers_mt(String tenant, String regExFilter, int limit) throws

RemoteException,UserImplException;

Description:

Return the list of all PBX Users (extension) for 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:

A string array containing the result.

8.41. setArsVariables

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

Returns: None.

Related functions: getArsVariables, applyArsRules

8.42. setNotes

Signature:

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

Description:

Set information for a note.

Parameters:

notes – a string array containing information about the notes. Element 0 contains the note name, 1 the description, 2 the access level, and 3 the notes content.

Returns: None.

8.43. setNotes_mt (Multi-Tenant Function)

Signature:

public void setNotes_mt(String tenant, String[] notes) throws

RemoteException,UserImplException;

Description: Set information for a note.

Parameters:

Tenant – the name of a tenant company

notes – a string array containing information about the notes. Element 0 contains the note name, 1 the description, 2 the access level, and 3 the notes content.

Returns: None.

8.44. setTenantProperties_mt (Multi-Tenant Function)

Signature:

public void setTenantProperties_mt (String tenant, String[] propertyNames, String[]

propertyValues) throws RemoteException,UserImplException;

Description:

Sets the values for properties specified in the propertyNames array to the values specifiedin 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.

Returns: None. If an error occurs, this method generates either a RemoteException or a UserImplException.

Notes:

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

8.45. setTenantProperty_mt (Multi-Tenant Function)

Signature:

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

Description:

Sets the value for property specified in the propertyName to the value specifiedin 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.

Returns:

None. If an error occurs, this method generates either a RemoteException or a UserImplException.

Notes:

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

8.46. setUserProperties

Signature:

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

Description:

Sets 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.

propertyValuesArray- a string array containing values to be set.

Returns:

None. If an error occurs, this method generates either a RemoteException or a UserImplException.

8.47. setUserProperties mt (Multi-Tenant Function)

Signature:

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

Description:

Sets the properties listed in the propertyNameArray to the corresponding values listed in the propertyValuesArray.

Parameters:

tenant is 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.

propertyValuesArray- a string array containing values to be set.

Returns:

None. If an error occurs, this method generates either a RemoteException or a UserImplException.

8.48. setUserProperty

Signature:

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

Description:

Sets 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.

propertyValuesArray- a string containing a property value.

Returns: None. If an error occurs, this method generates either a RemoteException or a UserImplException.

8.49. setUserProperty_mt (Multi-Tenant Function)

Signature:

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

Description:

Sets the property listed in the propertyName parameter to the corresponding value in the propertyValue parameter.

Parameters:

tenant is 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.

propertyValuesArray- a string containing a property value.

Returns:

None. If an error occurs, this method generates either a RemoteException or a UserImplException.

9. User Settings Properties

The table below contains property names that may be viewed or altered using the web service methods. This list is incomplete.

Property Name	Description	Value
admin	Is the user an admin user type?	true or false
allowjoin	Allow others to join conversation?	true or false
canjoin	Allowed to join other's conversation?	true or false
defaultpickup	Enable one touch Call Pickup for the predetermined group extensions by assigning the group number	group number
dtmfcommand	Allow this user to use call transfer/hold features?	true or false
email	Email for forwarding voice mail info	email address
emailattachment	Attach WAV file to email?	true or false
emailnotification	Enable email notification?	true or false
greetingtype	Greeting message	1 = Personal 2 = Alternative 3 = System
index	Pattern Setting	number
language	Language	en or ja
maxsessioncount	Maximum number of sessions	-1 means unlimited or 0 thru 6

Property Name	Description	Value
messageforward	The extension number(s) to which received voicemail messages will be forwarded	list of extensions
name	The description of this user	text
noanswerforward	number or SIP-URI to which the call will be forwarded when Ringer timeout has occurred	number or uri
password	password	text
personalivr	IVR Type	aa,setup,fm,ptn
phoneforward	phone number(s) or SIP-URI to directly forward all calls	comma separated list
recording	Activate Call Recording?	True or false
ringertime	Ringing timeout to wait for the called user to answer (in seconds)	number
rtprelay	RTP Relay	true,false,true_codec, false_g711u,default
userplugin	Type of Call Forwarding	user,fw,schedule,conf

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>:28080/pbx/servi
          ces/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://172.16.11.85:28080/pbx/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://172.16.11.85:28080/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++) {
              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://172.16.11.85:28080/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://172.16.11.85:28080/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://172.16.11.85:28080/pbx/servi
              ces/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>
```