Brekeke PAL WebSocket
Version 3

Developer's Guide

Brekeke Software, Inc.
Version
Brekeke PAL WebSocket v3 Developer's Guide

Copyright
This document is copyrighted by Brekeke Software, Inc.
Copyright © 2018 Brekeke Software, Inc.
This document may not be copied, reproduced, reprinted, translated, rewritten, or readdressed in whole or in 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 notice to the user.
**Table of Contents**

1. Purpose .......................................................................................................................... 5
2. Brekeke PAL WebSocket Usage ..................................................................................... 5
   2.1. Configuration at Brekeke PBX .................................................................................. 5
   2.2. URL ......................................................................................................................... 5
   2.3. Protocol .................................................................................................................... 5
3. Connecting with Brekeke PAL WebSocket ...................................................................... 6
   3.1. login ......................................................................................................................... 6
   3.2. URL format ............................................................................................................... 6
   3.3. Response .................................................................................................................. 7
4. Notifications ................................................................................................................... 8
   4.1. notify_callrecording .............................................................................................. 8
   4.2. notify_line .............................................................................................................. 8
   4.3. notify_park ............................................................................................................. 9
   4.4. notify_registered ................................................................................................... 9
   4.5. notify_status ......................................................................................................... 9
   4.6. notify_voicemail .................................................................................................. 10
   4.7. notify_queue .......................................................................................................... 11
5. Methods ........................................................................................................................ 11
   5.1. barge ..................................................................................................................... 11
   5.2. callforward ......................................................................................................... 12
   5.3. cancelTransfer .................................................................................................... 12
   5.4. conference ........................................................................................................... 13
   5.5. createExtension ................................................................................................. 13
   5.6. createTenant ...................................................................................................... 14
   5.7. deleteExtension ................................................................................................. 14
   5.8. deleteNote .......................................................................................................... 15
   5.9. deleteRouteVariables ....................................................................................... 15
   5.10. disconnect ........................................................................................................ 16
   5.11. getAllDids ....................................................................................................... 16
   5.12. getContact ....................................................................................................... 17
5.13. getContactList........................................................................................................ 17
5.14. getExtensionProperties ......................................................................................... 18
5.15. getExtensions .......................................................................................................... 18
5.16. getNote ................................................................................................................... 19
5.17. getNoteNames......................................................................................................... 20
5.18. getOptions .............................................................................................................. 20
5.19. getPnApplicationInfo.............................................................................................. 20
5.20. getPhonebooks..................................................................................................... 21
5.21. getPlanSwitchTimer................................................................................................. 21
5.22. getRouteTemplateNames ....................................................................................... 22
5.23. getRouteVariables.................................................................................................. 22
5.24. getTalkerInfo......................................................................................................... 23
5.25. getTenantProperties............................................................................................... 24
5.26. insertRouteVariables ............................................................................................ 24
5.27. makeCall................................................................................................................ 25
5.28. park....................................................................................................................... 25
5.29. pnmanage.............................................................................................................. 26
5.30. pnsend................................................................................................................ 27
5.31. remoteControl...................................................................................................... 27
5.32. setContact ........................................................................................................... 28
5.33. setDid.................................................................................................................. 28
5.34. setExtensionProperties......................................................................................... 30
5.35. setNote............................................................................................................... 30
5.36. setPlanSwitchTimer............................................................................................. 31
5.37. setTenantProperties............................................................................................. 31
5.38. startRecording ....................................................................................................... 32
5.39. stopRecording ....................................................................................................... 32
5.40. transfer ................................................................................................................ 33
5.41. unpark.................................................................................................................. 33
5.42. updateRouteVariables.......................................................................................... 34
6.  Phonebook utility for javascript.................................................................................. 35
6.1. Retrieving standard field information ........................................................... 35
6.2. Creating a display name .............................................................................. 36
7. Extension Settings Properties ......................................................................... 37
  7.1. Callback Extension .................................................................................. 37
  7.2. Conference Extension ............................................................................. 37
  7.3. Groups Extension .................................................................................... 37
  7.4. IVR Extension ......................................................................................... 38
  7.5. Schedule Extension ................................................................................ 40
  7.6. User extension ........................................................................................ 41
8. Sample Programs ............................................................................................. 44
  8.1. Example 1 ................................................................................................ 44
  8.2. Example 2 ................................................................................................ 45
1. Purpose

This document describes Brekeke PAL WebSocket, which is an interface that retrieves call details and also obtains and updates settings values from Brekeke PBX.

Software Requirements
Brekeke PAL WebSocket requires Brekeke PBX v3.2 or later with Brekeke PAL option enabled. Brekeke PBX must be installed with Apache Tomcat 7.0.42 or later.

2. Brekeke PAL WebSocket Usage

Use a WebSocket client that conforms to RFC6455.

2.1. Configuration at Brekeke PBX

Defining the valid WebSocket client IP addresses is required to connect with Brekeke PBX.

1. Log in to Brekeke PBX Admintool with admin privileges.
2. In the [Options] > [Settings] > [Valid WebSocket Client IP Pattern] field, enter a regular expression to define the client’s IP address pattern.

For example, "^192\.168\..+$" will include any client whose IP address starts with "192.168."

2.2. URL

Access Brekeke PAL WebSocket API using the following URL:

ws://<Brekeke PBX Host>:< Brekeke PBX Port>/pbx/ws?<login information>

For details about URL format, refer to section 3, “Connecting with Brekeke PAL WebSocket.”

2.3. Protocol

JSON-RPC 2.0 is used as a remote procedure call protocol. For more information, please refer to http://www.jsonrpc.org/.
3. Connecting with Brekeke PAL WebSocket

Get connected with Brekeke PBX by logging in as system administrator “sa”, administrator, or user extension and retrieve necessary information required in the following requests by defining parameters: status, voicemail, registered, park, line, and queue, and gain access to allow the use of methods. Once login succeeds with login_user other than “sa”, you will no longer need to set the tenant parameter in the request methods. If the tenant parameter is set, it will be ignored.

3.1. login

Description:
Send encrypted login information and encrypted login user password that are encrypted with MD5 hasher algorithm.

Parameters:

login_password – A string encrypted with MD5 hasher algorithm in the following format:

```
<login_user>:<nonce>:<login_password>
```

where <login_user> is login user extension number, <login_password> is login user password encrypted with MD5 hasher algorithm, and <nonce> is the value of parameter “nonce” got from the connection response.

Version:
3.3 or later

3.2. URL format

```
ws://<Brekeke-PBX-IP>:<Brekeke-PBX-Port>/pbx/ws?tenant=<tenant-name>&login_user=<username>&login_password=<password>&user=<user-extension>&registered=<true|false>&status=<true|false>&voicemail=<true>&callrecording=<true|false>&park=<park-value>&line=<line-value>&queue=<queue-value>
```

- `<Brekeke-PBX-IP>`: IP address of the server where Brekeke PBX is installed
- `<Brekeke-PBX-Port>`: The same port number as the one used to access Brekeke PBX Admintool
- `<tenant-name>`: Tenant name. Do not specify this value if logging in as “sa” or in the case of using the single tenant version.
<user-name> Login user name

<password> The password for the login_user extension. If login password is not set from URL, Brekeke PAL WebSocket login method can be used to send encrypted login information as explained above.

<user-extension[n]> The user extension(s) whose information will be retrieved from Brekeke PBX. This setting can be omitted when you don’t need to receive any user’s information.

The values can be: * (user=*), or multiple values connected by “&” can be set (ex. user=200&user=201&user=202). Setting * as value means all users of the system or under the specified tenant.

When the login_user does not have an admin privilege, all users specified in these parameters must be allowed by the [User] > [Settings] > [PAL WebSocket settings] > [Access users].

<park-value> Park ID(s) information will be retrieved from Brekeke PBX. The values can be: *, or multiple values connected by “&” can be set. (Refer to <user-extension>.)

<line-value> Line ID(s) information will be retrieved from Brekeke PBX. The values can be: *, or multiple values connected by “&” can be set. (Refer to <user-extension>.)

<queue-value> Group extension(s) information will be retrieved from Brekeke PBX. The values can be: *, or multiple values connected by “&” can be set. (Refer to <user-extension>.)

3.3. Response

When login password is not specified in the URL, “login_password_required” event with one parameter “nonce” will be returned.

If login succeeds, WebSocket notifications will be returned with request information.

If login fails, an error message will be returned.
4. Notifications

Once a Brekeke PAL WebSocket connection has been created, the notify methods listed below will be sent back to clients. These notify methods contain the information required by request methods.

4.1. notify_callrecording

Description:
Describes the user’s call recording status.

Parameters:
user – user extension
talker_id – talker ID (v3.6 or later)
status – on or off

Version:
3.3 or later

4.2. notify_line

Description:
Shared call status.

Parameters:
line – shared line ID with index
status – on or off
line_talker_id – talker ID of the shared line
line_talker – talker of the shared line
assigned_talker_id – talker ID of the user in the shared call
4.3. **notify_park**

**Description:**
Parked call status.

**Parameters:**
- park – park number
- status – on or off
- room_id – room ID of the call when park status is on, required by some request methods below as rid
- talker_id – talker ID of the user when park status is on, required by some request methods below as tid

4.4. **notify_registered**

**Description:**
Describes the user’s registered status.

**Parameters:**
- user – user extension
- registered – true or false

4.5. **notify_status**

**Description:**
Information about call status.

**Parameters:**
- user – user extension
- status – call status code
- other_number – the extension of the other user in the call
- room_id – room ID of the call, required by some request methods below as rid
- talker_id – talker ID of the user, required by some request methods below as tid
- user_display_name – the display name of the user
- other_user_display_name – the display name of the other user in the call
- time – the time stamp of the call
- logid – the ID number, which is the same as the number in the request method
rescode – response code, used only in the disconnected call response with status -1

disconnected_by – 1 or 0, used only in the disconnected call response with status -1. “1” means the call is disconnected by the callee and “0” means the call is disconnected by the caller.

q850code – q850 code, used only in the disconnected call response with status -1

**Call status codes:**

CALLING = 0

INCOMING = 1

CALL_SUCCESS = 2

ENDTALKING = 12

ANSWER_SUCCESS = 14

PARK_CANCEL = 21

PARK_START = 30

STARTRINGING = 65

HOLD = 35

UNHOLD = 36

DISCONNECT = -1

4.6. **notify_voicemail**

**Description:**

Information about user voicemail.

**Parameters:**

user – user extension

ew – the number of new voicemail messages

unread – the number of unread voicemail messages

read – the number of read voicemail messages

saved – the number of saved voicemail messages
4.7. **notify_queue**

**Description:**
Information about call queue.

**Parameters:**
- extension – group extension of the call queue
- status – on or off
- room_id – room ID of the call

**Version:**
3.6 or later

5. **Methods**

5.1. **barge**

**Permissions:**
admin

**Description:**
Allows a user to barge into a conversation. The barging user is the extension specified in the user parameter. The barging user will be able to choose whether to listen or speak to the participants in the conversation.

**Parameters:**
- tenant – tenant name
- user – the extension of the user who is barging into the conversation
- tid – an integer; the talker ID of one of the users whose conversation is barged into
- listen – true or false
- speak – true, false, or tutor

**Returns:**
A success message or an error message.
5.2. callforward

Permissions:
admin

Description:
Forwards a conversation to a list of users.

Parameters:
tenant – tenant name
rid – room ID assigned to the conversation
user – an array of user extensions to whom the conversation will be forwarded

Returns:
A success message or an error message.

5.3. cancelTransfer

Permissions:
admin or user

Description:
Cancel an attended transfer call.

Parameters:
tenant – tenant name
tid – talker ID that is in the process of being transferred

Returns:
A success message or an error message.

Related methods:
transfer
5.4.  conference

Permissions:
admin or user

Description:
Start three-way conference during an attended transfer call.

Parameters:
tenant – tenant name
talker_id – talker ID that is in the process of being transferred

Returns:
A success message (starts with succeeded:) or an error message.

Related methods:
transfer

Version:
3.7 or later

5.5.  createExtension

Permissions:
admin

Description:
Creates a Brekeke PBX extension.

Parameters:
tenant – tenant name
extension – a string representing an extension ID as defined in Brekeke PBX Admintool
password – a string representing the corresponding password for a user’s voicemail box
login_password – a string representing the corresponding password for a user account
type – a string representing the extension type; value can be user, ringgroup, ivr, conditional, conference, or callback

If the “type” is set to non-user, the password and login_password do not need to be provided.

Returns:
If the method succeeds, it returns “true” in the result field.

If the method fails, it returns an associative array in the error field.
Related methods:
deleteExtension, getExtensionProperties, getExtensions, setExtensionProperties

5.6. createTenant

Permissions:
sa

Description:
Creates a new tenant in Brekeke PBX.

Parameters:
tenant – tenant name

Returns:
If the method succeeds, it returns “true” in the result field.
If the method fails, it returns an associative array in the error field.

Related methods:
getTenantProperties, setTenantProperties

5.7. deleteExtension

Permissions:
admin

Description:
Deletes a Brekeke PBX extension from a tenant.

Parameters:
tenant – tenant name
extension – a string representing an extension defined under the tenant

Returns:
If the method succeeds, it returns “true” in the result field.
If the method fails, it returns “false” in the result field.

Related methods:
createExtension, getExtensionProperties, getExtensions, setExtensionProperties
5.8. deleteNote

Permissions:
admin or user

Description:
Deletes a note.

Parameters:
tenant – tenant name
name – the name of the note to be deleted

Returns:
If the method succeeds, it returns “true” in the result field.
If the method fails, it returns an associative array in the error field.

Related methods:
getNote, getNoteNames, setNote

5.9. deleteRouteVariables

Permissions:
sa

Description:
Deletes a route and its Route Local Variables values under the defined ARS template.

Parameters:
template – the ARS route template name
name – the route name

Returns:
If the method succeeds, it returns “OK” in the result field.

Related methods:
gRoutingModuleTemplateNames, getRouteVariables, insertRouteVariables, updateRouteVariables
5.10. disconnect

Permissions:
admin or user

Description:
Disconnects a call with either tid or rid.

Parameters:
tenant – tenant name
tid – talker ID of the user’s session. Admin users can disconnect a call with either tid or rid.
rid – room ID of a call. Available only when log in with admin privilege. Admin users can disconnect a call with either tid or rid.
delay – the amount of time to delay before disconnecting a call when the call is disconnected by rid

Returns:
A success message or an error message.

5.11. getAllDids

Permissions:
admin

Description:
Gets the tenant’s Route Local Variables within the Route Templates that are configured as DID on the Field settings page.

✓ Fields that do not have the Tenant Access (List) checkbox selected in the Field settings cannot be returned.

Parameters:
tenant – tenant name

Returns:
If the method succeeds, it returns an array of each route’s Route Local Variables and value as a result.

Related methods:
setDid
5.12. getContact

Permissions:
admin or user

Description:
Retrieves a contact.

Parameters:
aid – the ID of the contact

Returns:
If the method succeeds, it returns contact information in the format shown below:

- aid – the ID of the contact
- display_name – a display name of the contact
- phonebook – a name of the phonebook that the contact belongs to
- shared – true if the phonebook is shared
- info – all field data of the contact (Refer to the [Phone book] > [Import/Export] page of the product admintool for the details of the standard fields.)

Related methods:
setContact

Version:
3.8 or later

5.13. getContactList

Permissions:
admin or user

Description:
Retrieves lists of the contact summary.

Parameters:
phonebook – a name of the phonebook
shared – true if the phonebook is shared (admin only)
search_text – keywords to search contacts (separated by a white space)
offset – an offset position to retrieve the contact (default: 0)
limit – maximum number of contact to be retrieved (default: 100, maximum: 1000)
Returns:
If the method succeeds, it returns an array of a contact summary in the format shown below:

- aid – the ID of the contact
- display_name – a display name of the contact

Version:
3.8 or later

5.14. getExtensionProperties

Permissions:
admin or user

Description:
Retrieves the property values of the properties specified in the property_names parameter.

Parameters:
tenant – the name of the tenant company
extension – a string representing an extension
property_names – a string array defining property names of the values to be retrieved

Returns:
If the method succeeds, it returns an array that contains the requested property_names values to the result field. If the method fails, it returns an associative array to the error field.

Related methods:
createExtension, deleteExtension, getExtensions, setExtensionProperties

5.15. getExtensions

Permissions:
admin

Description:
Returns a list of extensions that match the regular expressions filter for a particular tenant.

Parameters:
tenant – tenant name
pattern – a regular expression that the result must match
limit – restricts the number of tenants that are returned. For unlimited, set to -1.
type – a string representing the user type. Value can be user, ringgroup, ivr, conditional, conference, or callback.

property_names – a string array defining property names of the values to be retrieved

**Returns:**

If the method succeeds, it returns an array to the result field that contains the extensions for the type requested. If the property_names is specified, it returns an array of arrays that contain the extension as the first item and requested property_names values as the following items to the result field (v3.7 or later). If the method fails, it returns an associative array in the error field.

**Related methods:**
createExtension, deleteExtension, getExtensionProperties, setExtensionProperties

### 5.16. getNote

**Permissions:**

admin or user

**Description:**

Retrieves information about a note.

**Parameters:**

tenant – tenant name

name – the name of the note

**Returns:**

If the method succeeds, it returns the following list in the result field:

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

description – description

name – note name

note – note content

If the method fails, it returns an associative array in the error field.

**Related methods:**
deleteNote, getNoteNames, setNote
5.17. getNoteNames

Permissions:
admin or user

Description:
Returns a list of note names.

Parameters:
tenant – tenant name

Returns:
If the method succeeds, it returns an array that contains the note names in the result field.

Related methods:
deleteNote, getNote, setNote

5.18. getOptions

Permissions:
admin or user

Description:
Retrieves the optional feature information.

Returns:
If the method succeeds, it returns the optional feature information such as pn (Push Notification),
sdn (SDN).

Version:
3.7.5.6 or later

5.19. getPnApplicationInfo

Permissions:
admin or user

Description:
Retrieves application information that registered for Push Notification.

Parameters:
appname – Name of the application that uses push notification
service_id - 1: APNS, 2: GCM/FCM, 3: WebPush
Returns:
If the method succeeds, it returns application information in the format shown below:

appid – the Application ID

desc – description of the application

Version:
3.8 or later

5.20. getPhonebooks

Permissions:
admin or user

Description:
Retrieves all the phonebook’s names that the user has access to.

Returns:
If the method succeeds, it returns an array of a phonebook summary in the format shown below:

phonebook – a phonebook name

shared – true if the phonebook is shared (can be omitted)

Version:
3.8 or later

5.21. getPlanSwitchTimer

Permissions:
admin or user

Description:
Retrieves the Timer settings.

Parameters:
extension – a string representing an extension

number – 1 or 2 represents Timer 1 or Timer 2 in the extension setting. Default setting is 1.

Returns:
If the method succeeds, it returns a string in the result field formatted as “P<start-end-date>A<days-of-week>W<weeks>D<date-pattern>T<times>.” For details about the returned
string format, refer to the “Schedule Extension” table in section 7, “Extension Settings Properties.”

Related methods:
setPlanSwitchTimer

5.22. **getRouteTemplateNames**

Permissions:
admin

Description:
Retrieves a list of tenant route template names.

Parameters:
tenant – tenant name

Returns:
If the method succeeds, it returns an array that contains the Route Names in the result field.

Related methods:
deleteRouteVariables, getRouteVariables, insertRouteVariables, updateRouteVariables

5.23. **getRouteVariables**

Permissions:
sa

Description:
Retrieves the route name and its Route Local Variables values assigned to the defined tenant and created under the defined route template name.

Parameters:
tenant – tenant name
template – name of the ARS route template

Returns:
If the method succeeds, it returns an array that contains each route’s Route Local Variables in the result field. See below for more details:

val 0 – 1: Enabled, 0: Disabled
val 1 – tenant name
val 2 – route name
val 3 – password
val 4~5 – <reserved>
val 6~14 – v1~v9
val 15~23 – w1~w9
val 24~32 – x1~x9

Related methods:
deleteRouteVariables, getRouteTemplateNames, insertRouteVariables, updateRouteVariables

5.24. getTalkerInfo

Permissions:
admin or user

Description:
Returns talker’s information from the call.

Parameters:
tenant – tenant name
room_id – room ID of the call
talker_id – talker ID

Returns:
If talker_id is specified, it returns the talker’s information. If not specified, it returns all of the talker’s information.

Version:
3.6 or later
5.25. \textbf{getTenantProperties}

\textbf{Permissions:}

sa

\textbf{Description:}

Returns the list of tenant properties specified in the property\_names array.

\textbf{Parameters:}

\begin{itemize}
  \item tenant – tenant name
  \item property\_names – a string array containing the property values to retrieve
\end{itemize}

\begin{itemize}
  \item ✓ Valid property names are: tenantid, desc, maxrecordingsessions, maxusers, and maxsessions.
\end{itemize}

\textbf{Returns:}

If the method succeeds, it returns an array that contains the requested property\_names values in the result field.

If the method fails, it returns an associative array in the error field.

\textbf{Related methods:}

createTenant, setTenantProperties

5.26. \textbf{insertRouteVariables}

\textbf{Permissions:}

sa

\textbf{Description:}

Adds a new route and its Route Local Variables values to a specified route template.

\textbf{Parameters:}

\begin{itemize}
  \item template – the name of an ARS route template
  \item values – a string array containing the route name and its related Route Local Variable setting
\end{itemize}

\begin{itemize}
  \item val 0 – 1: Enabled, 0: Disabled
  \item val 1 – tenant name
  \item val 2 – route name
  \item val 3 – password
  \item val 4~5 – <reserved>
  \item val 6~14 – v1~v9
  \item val 15~23 – w1~w9
\end{itemize}
val 24~32 – x1~x9
val 0 to val 6 are required variables for this method. Set an empty string for any variable that is not needed.

**Returns:**
If the method succeeds, it returns “true” in the result field.
If the method fails, it returns an associative array in the error field.

**Related methods:**
deleteRouteVariables, getRouteTemplateNames, getRouteVariables, updateRouteVariables

### 5.27. makeCall

**Permissions:**
admin or user

**Description:**
Places a call between a caller and callee.

**Parameters:**
tenant – tenant name
user – a PBX user extension representing the call owner
from – the caller’s user extension or phone number
to – an array of the callee’s user extensions or phone numbers
type – Strings: “1” or “2”
Type “1” will simultaneously call the “from” number and “to” number and then connect them.
Type “2” will call the “from” number first. When the “from” number picks up, it calls the “to” number then connects the two calls.

**Returns:**
A success message or an error message.

### 5.28. park

**Permissions:**
admin or user

**Description:**
Parks a call.
Parameters:

- tenant – tenant name
- tid – talker ID whose call will be parked
- number – the number for retrieving the parked call

Returns:

A success message or an error message.

Related methods:

unpark

5.29. **pnmanage**

Permissions:

admin or user

Description:

Adds or removes device information used for Push Notification.

Parameters:

- command – add or remove
- username – The SIP user ID. If you are not an admin user, this ID must be one of the user’s Phone IDs.
- service_id – 1: APNS, 2: GCM/FCM, 3: WebPush (only for command: add)
- application_id – the Application ID/Sender ID for the application (in case of command: add)
- device_id – the device ID (in case of service_id: 1 or 2)
- endpoint – the endpoint (in case of service_id: 3)
- key – the data encryption key (in case of service_id: 3)
- auth_secret – the authentication secret (in case of service_id: 3)
- user_agent – the user-agent of the browser (in case of service_id: 3)

Returns:

If the method succeeds, it returns “Succeeded:” in the result field. If not, it returns a string that starts with “Error:.”

Version:

3.7.5.6 or later
5.30.  **pnsend**

**Permissions:**
admin or user

**Description:**
Sends Push Notification messages.

**Parameters:**
username – user extension if type="user" is set. When type="phone_id" is set, It is the SIP user ID. If you are not an admin user, this ID must be one of the user's Phone IDs.
type – “user” or “phone_id”. When “user” is set, the notifications will be sent all of the user's phone.
message – message
title – title
button – button
sound_file – the sound file
image_file – the image file
expires – expiration in seconds
counter – the counter
priority – the priority (1-10)

**Version:**
3.8 or later

5.31.  **remoteControl**

**Permissions:**
admin or user

**Description:**
Sends NOTIFY event to the user's phone.

**Parameters:**
tenant – tenant name
talker_id – talker ID of the call
action – “talk” which takes a phone off-hook or “hold” which will place the call on hold

**Version:**
3.6 or later

5.32. setContact

Permissions:
admin or user

Description:
Sets a contact.

Parameters:
- aid – the ID of the contact
- phonebook – name of the phonebook
- shared – set true if the phonebook is shared
- info – all field data of the contact (Refer to the [Phone book] > [Import/Export] page for the details of the standard field of the data.)

Returns:
If the method succeeds, it returns an object with a property aid like { “aid” : 6636 }. (Available only in version 3.8.3.5 or later)

Related methods:
- getContact

Version:
3.8 or later

5.33. setDid

Permissions:
admin

Description:
Sets the Route Local Variables within the tenant’s Route Templates that are configured as DID on the Field settings page.

✓ Fields that do not have the Tenant Access (Edit) checkbox selected in Field settings cannot be returned.

Parameters:
- template – the name of an ARS route template
name – Route Name

enabled – 1: Enabled, 0: Disabled

<Variable> - <value> – The variable can be v1~v9, w1~w9, or x1~x9.

Sample request:

```
{"jsonrpc":"2.0", "method":"setDid", "params":{"template":"templ","name":"n1", "v1":"value1", "v2":"value2"}, "id":1}
```

Returns:

If the method succeeds, it returns "OK" in the result field.

Related methods:

getAllDids
5.34. setExtensionProperties

Permissions:
admin or user

Description:
Sets the values specified in the properties associative array to the extension properties.

Parameters:
tenant – the name of the tenant company
extension – a string representing a username as defined in Brekeke PBX Admintool
properties – an associative array containing the properties to be configured

Returns:
If the method succeeds, it returns “OK” in the result field.
If the method fails, it returns an associative array in the error field.

Related methods:
createExtension, deleteExtension, getExtensionProperties, getExtensions

5.35. setNote

Permissions:
admin or user

Description:
Sets information for a note in the defined tenant.

Parameters:
tenant – tenant name
name – note name
description – description
useraccess – access levels are No access: 0; Read only: 1; Read/Write: 2.
note – note content

Returns:
If the method succeeds, it returns “OK” in the result field.
If the method fails, it returns an associative array in the error field.

Related methods:
deleteNote, getNote, getNoteNames
5.36. **setPlanSwitchTimer**

**Permissions:**

user

**Description:**

Sets timer information.

**Parameters:**

- **extension** – a string representing an extension. It can only be user or schedule extensions.
- **number** – 1 or 2 represents Timer 1 or Timer 2 in the extension setting. Default setting is 1.
- **schedule** – a string with format “P<start-end-date>A<days-of-week>W<weeks>D<date-pattern>T<times>.” For details about the returned string format, refer to the “Schedule Extension” table in section 7, “Extension Settings Properties.”

**Returns:**

If the method succeeds, it returns “OK” in the result field.

**Related methods:**

getPlanSwitchTimer

5.37. **setTenantProperties**

**Permissions:**

sa

**Description:**

Sets the values specified in the properties associative array to the Tenant properties.

**Parameters:**

- **tenant** – tenant name
- **properties** – an associative array containing the properties to be configured
  
  ✓ Valid property names are: **tenantid, desc, maxrecordingsessions, maxusers, and maxsessions.**

**Returns:**

If the method succeeds, it returns “OK” in the result field.

If the method fails, it returns an associative array in the error field.

**Related methods:**

createTenant, getTenantProperties
5.38. startRecording

Permissions: 
admin or user

Description: 
Records a conversation.

Parameters: 
tenant – tenant name
tid – talker ID whose conversation will be recorded

Returns: 
A success message or an error message.

Related methods: 
stopRecording

5.39. stopRecording

Permissions: 
admin or user

Description: 
Stops recording.

Parameters: 
tenant – tenant name
tid – talker ID whose recording will be stopped

Returns: 
A success message or an error message.

Related methods: 
startRecording
5.40. transfer

Permissions:
admin or user

Description:
Transfers a call.

Parameters:
tenant – tenant name
user – the extension to transfer the call to
tid – the talker ID of the caller whose conversation will be transferred
mode – when mode is set as blind, a blind transfer will be performed; otherwise, it will be an attended transfer

Returns:
A success message or an error message.

5.41. unpark

Permissions:
admin or user

Description:
Unparks a call.

Parameters:
tenant – tenant name
user – the user extension number with which the call will be unparked
number – the retrieve number

Returns:
A success message or an error message.

Related methods:
Park
5.42. updateRouteVariables

Permissions:
sa

Description:
Sets a route and its Route Local Variables values to a specified route template.

Parameters:
template – the name of an ARS route template
update_password – true or false
values – a string array containing route name and related Route Local Variables settings

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 an empty string for any variables that are not needed.

Returns:
If the method succeeds, it returns “true” in the result field.

Related methods:
deleteRouteVariables, getRouteTemplateNames, getRouteVariables, insertRouteVariables
6. Phonebook utility for javascript

Besides above mentioned APIs (getPhonebooks(), getContactList(), getContact(), setContact()), using javascript API (phonebook.js) may be desired in the cases of implementing the phonebook feature into an application.

Javascript API (phonebook.js) allows:

- Retrieving standard field information
- Creating a display name

Load the http(s)://<host>:<port>/pbx/common/js/brekeke/phonebook/phonebook.js file to use those functions.

6.1. Retrieving standard field information

Example:

```javascript
var manager = Brekeke.Phonebook.getManager('en');
var item = manager.item;
```

Specify a language (en, ja) to retrieve manager object. The item property of the manager object is an array of the field information. The properties of the field information should be as shown below:

- id – the field ID. If an id starts with $, the caption properties will be used. Otherwise, id will be used as a caption.
- caption – the caption of the field. It is omitted when the id does not start with $.
- onscreen – if true, the field is always shown
- type – type of the field (phone or address)

For details of the standard fields, refer to the [Phone book] > [Import/Export] page of the product admintool.
6.2. Creating a display name

Example:

```javascript
var manager = Brekeke.Phonebook.getManager('en');
var display_name = manager.toDisplayName( contact.info );
```

Using this function, when editing a contact information, the application can show the contact's display name like the [Phone book] > [Contact] page of the product admintool.
7. Extension Settings Properties

The tables below contain the property names that may be viewed or altered using the WebSocket methods. These lists are not comprehensive:

### 7.1. Callback Extension

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Extension type</td>
<td>callback</td>
</tr>
<tr>
<td>callback.callee</td>
<td>Callback callee</td>
<td>Extension number</td>
</tr>
<tr>
<td>desc</td>
<td>Description</td>
<td>Text</td>
</tr>
<tr>
<td>noanswerforward</td>
<td>Forwarding destination (No answer)</td>
<td>Extension number</td>
</tr>
<tr>
<td>ringertime</td>
<td>Ringer time</td>
<td>Number</td>
</tr>
</tbody>
</table>

### 7.2. Conference Extension

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Extension type</td>
<td>conference</td>
</tr>
<tr>
<td>callback.callee</td>
<td>Callback callee</td>
<td>Extension number</td>
</tr>
<tr>
<td>conf.accept</td>
<td>Applied to (Caller numbers)</td>
<td>Wildcard pattern</td>
</tr>
<tr>
<td>desc</td>
<td>Description</td>
<td>Text</td>
</tr>
<tr>
<td>exit.on.host.disconnected</td>
<td>Exit all when host leaves</td>
<td>true or false</td>
</tr>
<tr>
<td>phoneforward</td>
<td>Forwarding destinations</td>
<td>Comma-separated extension numbers</td>
</tr>
</tbody>
</table>

### 7.3. Groups Extension

#### Simultaneous Ring

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Extension type</td>
<td>ringgroup</td>
</tr>
<tr>
<td>stype</td>
<td>Specific type</td>
<td>sr</td>
</tr>
<tr>
<td>desc</td>
<td>Description</td>
<td>Text</td>
</tr>
<tr>
<td>noanswerforward</td>
<td>Forwarding destination (No answer)</td>
<td>Extension number</td>
</tr>
<tr>
<td>phoneforward</td>
<td>Forwarding destinations</td>
<td>Comma-separated extension numbers</td>
</tr>
<tr>
<td>ringertime</td>
<td>Ringer time</td>
<td>Number</td>
</tr>
</tbody>
</table>
### Call Hunting

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Extension type</td>
<td>ringgroup</td>
</tr>
<tr>
<td>stype</td>
<td>Specific type</td>
<td>rr</td>
</tr>
<tr>
<td>desc</td>
<td>Description</td>
<td>Text</td>
</tr>
<tr>
<td>noanswerforward</td>
<td>Forwarding destination (No answer)</td>
<td>Extension number</td>
</tr>
<tr>
<td>switchmode</td>
<td>Mode</td>
<td>- cyclic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- ascending</td>
</tr>
<tr>
<td>phoneforward</td>
<td>Hunt group extensions</td>
<td>Comma-separated extension</td>
</tr>
<tr>
<td>queuingcallinterval</td>
<td>Call interval (msec)</td>
<td>Number</td>
</tr>
<tr>
<td>queuingmax</td>
<td>Max number of calls in the queue</td>
<td>Number</td>
</tr>
<tr>
<td>queuingtime</td>
<td>Waiting time in the queue</td>
<td>Number</td>
</tr>
<tr>
<td>ringertimes</td>
<td>Ringer time</td>
<td>Numbers separated by comma</td>
</tr>
<tr>
<td>round.tryonce</td>
<td>Single attempt</td>
<td>true or false</td>
</tr>
</tbody>
</table>

#### 7.4. IVR Extension

**Auto Attendant**

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Extension type</td>
<td>ivr</td>
</tr>
<tr>
<td>stype</td>
<td>Specific type</td>
<td>aa</td>
</tr>
<tr>
<td>desc</td>
<td>Description</td>
<td>Text</td>
</tr>
<tr>
<td>ex.autotransfer</td>
<td>Default operator</td>
<td>User extension</td>
</tr>
<tr>
<td>ex.calltimesec</td>
<td>Ring timeout (sec)</td>
<td>Number</td>
</tr>
<tr>
<td>ex.digitmaxlength</td>
<td>Max input digits</td>
<td>Number</td>
</tr>
<tr>
<td>ex.dtmfwaittimesec</td>
<td>DTMF timeout (sec)</td>
<td>Number</td>
</tr>
<tr>
<td>ex.maxretry</td>
<td>Max retry count</td>
<td>Number</td>
</tr>
<tr>
<td>ex.speeddial</td>
<td>Speed dial</td>
<td></td>
</tr>
<tr>
<td>ex.useronly</td>
<td>Transfer to unregistered users</td>
<td>true or false</td>
</tr>
<tr>
<td>language</td>
<td>Language</td>
<td>en or ja</td>
</tr>
</tbody>
</table>
### Add/Remove Forwarding Destinations

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Extension type</td>
<td>ivr</td>
</tr>
<tr>
<td>stype</td>
<td>Specific type</td>
<td>fm</td>
</tr>
<tr>
<td>desc</td>
<td>Description</td>
<td>Text</td>
</tr>
<tr>
<td>language</td>
<td>Language</td>
<td>en or ja</td>
</tr>
<tr>
<td>fm.targetusers</td>
<td>Target groups</td>
<td>Group extension numbers, separated by commas</td>
</tr>
</tbody>
</table>

### Switch Plan

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Extension type</td>
<td>ivr</td>
</tr>
<tr>
<td>stype</td>
<td>Specific type</td>
<td>ptn</td>
</tr>
<tr>
<td>desc</td>
<td>Description</td>
<td>Text</td>
</tr>
<tr>
<td>language</td>
<td>Language</td>
<td>en or ja</td>
</tr>
<tr>
<td>ptn.index</td>
<td>Plan number</td>
<td>Plan number</td>
</tr>
<tr>
<td>ptn.toggle</td>
<td>On/Off</td>
<td>true or false</td>
</tr>
</tbody>
</table>

### Script

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Extension type</td>
<td>ivr</td>
</tr>
<tr>
<td>stype</td>
<td>Specific type</td>
<td>scr</td>
</tr>
<tr>
<td>desc</td>
<td>Description</td>
<td>Text</td>
</tr>
<tr>
<td>ivr.script.autoanswer</td>
<td>Auto Answer</td>
<td>true or false</td>
</tr>
<tr>
<td>ivr.script.function</td>
<td>Function name</td>
<td>Text</td>
</tr>
<tr>
<td>ivr.script.note</td>
<td>Note name</td>
<td>Text</td>
</tr>
<tr>
<td>ivr.script.parameter</td>
<td>Parameter</td>
<td></td>
</tr>
<tr>
<td>language</td>
<td>Language</td>
<td>en or ja</td>
</tr>
</tbody>
</table>

### Flow

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Extension type</td>
<td>ivr</td>
</tr>
<tr>
<td>stype</td>
<td>Specific type</td>
<td>fl</td>
</tr>
<tr>
<td>desc</td>
<td>Description</td>
<td>Text</td>
</tr>
<tr>
<td>Property Name</td>
<td>Description</td>
<td>Value</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td>ivr.flow.autoanswer</td>
<td>Auto Answer</td>
<td>true or false</td>
</tr>
<tr>
<td>ivr.flow.name</td>
<td>Flow name</td>
<td>Text</td>
</tr>
<tr>
<td>ivr.flow.parameter</td>
<td>Properties</td>
<td></td>
</tr>
<tr>
<td>language</td>
<td>Language</td>
<td>en or ja</td>
</tr>
</tbody>
</table>

### 7.5. Schedule Extension

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Extension type</td>
<td>conditional</td>
</tr>
<tr>
<td>desc</td>
<td>Description</td>
<td>Text</td>
</tr>
<tr>
<td>pln[n]_d_noanswerforward</td>
<td>Plan[n] default Forwarding Schedule Call Forwarding destination</td>
<td>Extension number</td>
</tr>
<tr>
<td>pln[n]_d_p[n]_paging</td>
<td>Plan[n] phone[n] default paging</td>
<td>true or false</td>
</tr>
<tr>
<td>pln[n]_ptn[n]_callerroute</td>
<td>Route selection</td>
<td>- any - external - internal</td>
</tr>
<tr>
<td>pln[n]_ptn[n]_filtertext</td>
<td>Plan[n] Forwarding Schedule[n] Filter setting</td>
<td>Regular expression</td>
</tr>
<tr>
<td>pln[n]_ptn[n]_filtertype</td>
<td>Plan[n] Forwarding Schedule[n] Filter match radio box</td>
<td>match or unmatch</td>
</tr>
</tbody>
</table>
| pln[n]_ptn[n]_timeschedule | Plan[n] Forwarding Schedule[n] time schedule setup | Format: P<start-end-date>A<days-of-week>W<weeks>D<date-pattern>T<times>  
Sample setup:  
- P<start-end-date>  
Pyyyyymmdyyyyymmd  
- T<times>  
Thhmmhhmmhhmmhhmm  
- D<date-pattern>  
[Include] or [Exclude] days separated by comma  
- A<days-of-week> and W<weeks>  
Decimal value equals the binary number represented by bit flag of selected fields. Right side holds higher-value |
<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>pln[n]_ptn[n]_noanswerforward</td>
<td>Plan[n] Forwarding Schedule[n] Call Forwarding destination</td>
<td>Extension number</td>
</tr>
<tr>
<td>pln[n]_ptn[n]_p[n]_paging</td>
<td>Plan[n] Forwarding Schedule[n] phone[n] paging</td>
<td>true or false</td>
</tr>
</tbody>
</table>

### 7.6. User extension

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin</td>
<td>If an admin user</td>
<td>true or false</td>
</tr>
<tr>
<td>allowjoin</td>
<td>Allow others to join my conversation</td>
<td>true or false</td>
</tr>
<tr>
<td>automonitor</td>
<td>Automatic Monitoring</td>
<td>Comma-separated user extensions</td>
</tr>
<tr>
<td>busyforward.voicemail</td>
<td>If forward call to voicemail when user is busy</td>
<td>true or false</td>
</tr>
<tr>
<td>canjoin</td>
<td>Join other's conversation</td>
<td>true or false</td>
</tr>
<tr>
<td>defaultpickup</td>
<td>Call pickup group</td>
<td>A group extension number</td>
</tr>
<tr>
<td>desc</td>
<td>User description</td>
<td>Text</td>
</tr>
<tr>
<td>email</td>
<td>Email address</td>
<td>Email addresses where voicemails will be forwarded, separated by comma</td>
</tr>
<tr>
<td>emailattachment</td>
<td>Attach WAV file to email</td>
<td>true or false</td>
</tr>
<tr>
<td>emailnotification</td>
<td>Enable email notification or not</td>
<td>true or false</td>
</tr>
<tr>
<td>greetingtype</td>
<td>Greeting message</td>
<td>1 = Personal 2 = Alternative 3 = Default System</td>
</tr>
<tr>
<td>language</td>
<td>Language</td>
<td>en or ja</td>
</tr>
<tr>
<td>login.password</td>
<td>User login password</td>
<td>Text</td>
</tr>
<tr>
<td>maxsessioncount</td>
<td>Max inbound sessions</td>
<td>-1 means unlimited, or 0 thru 6</td>
</tr>
<tr>
<td>messageforward</td>
<td>Message forwarding</td>
<td>A list of user extensions, separated by comma</td>
</tr>
<tr>
<td>name</td>
<td>Display name of this user</td>
<td>Text</td>
</tr>
<tr>
<td>noanswerforward.voicemail</td>
<td>If forward call to user</td>
<td>true or false</td>
</tr>
<tr>
<td>Property Name</td>
<td>Description</td>
<td>Value</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>voicemail when user does not answer the call</td>
<td></td>
<td></td>
</tr>
<tr>
<td>password</td>
<td>User voicemail box password</td>
<td>Text</td>
</tr>
<tr>
<td>pln[n]_d_anothercall.beep</td>
<td>Beep on Incoming Call</td>
<td>true or false</td>
</tr>
<tr>
<td>pln[n]_d_busyforward</td>
<td>[Call Forwarding] &gt; [Forwarding Destination (Busy)]</td>
<td>Text</td>
</tr>
<tr>
<td>pln[n]_d_callnext</td>
<td>Call next phone if phone stops ringing</td>
<td>true or false</td>
</tr>
<tr>
<td>pln[n]_d_knockknock</td>
<td>Knock Knock period</td>
<td>Number</td>
</tr>
<tr>
<td>pln[n]_d_knockknock_onlyinternal</td>
<td>Only from internal extension</td>
<td>true or false</td>
</tr>
<tr>
<td>pln[n]_d_noanswerforward</td>
<td>[Call Forwarding] &gt; [Forwarding Destination (No answer)]</td>
<td>Text</td>
</tr>
<tr>
<td>pln[n]_d_phoneforward</td>
<td>[Call Forwarding] &gt; [Other Forwarding Destinations]</td>
<td>Comma-separated list of phone numbers</td>
</tr>
<tr>
<td>pln[n]_d_ringertime</td>
<td>[Call Forwarding] &gt; [Ringer time (sec)]</td>
<td>Number</td>
</tr>
<tr>
<td>pln[n]_d_p[n]_delay</td>
<td>Plan[n] phone[n] [Delay (sec)] setting</td>
<td>Number</td>
</tr>
<tr>
<td>pln[n]_d_p[n]_enabled</td>
<td>If enabled, plan[n] phone[n]</td>
<td>true or false</td>
</tr>
<tr>
<td>pln[n]_d_p[n]_paging</td>
<td>Plan[n] phone[n] default paging</td>
<td>true or false</td>
</tr>
<tr>
<td>pln[n]_d_p[n]_ringertime</td>
<td>Plan[n] phone[n] ringer time</td>
<td>Number</td>
</tr>
<tr>
<td>pnumber[n]</td>
<td>Phone[n] ID</td>
<td>Text</td>
</tr>
<tr>
<td>p[n]_ptype</td>
<td>Phone[n] Type</td>
<td>Phone type set at [Options] &gt; [Phone Type]</td>
</tr>
<tr>
<td>recording</td>
<td>Call Recording</td>
<td>True or false</td>
</tr>
<tr>
<td>recording.pattern</td>
<td>Call Recording Patterns</td>
<td>Comma-separated list of patterns</td>
</tr>
<tr>
<td>recordlength</td>
<td>Message recording length</td>
<td>Number</td>
</tr>
<tr>
<td>resourcemap</td>
<td>Resource map</td>
<td>Text</td>
</tr>
<tr>
<td>type</td>
<td>User type</td>
<td>User</td>
</tr>
<tr>
<td>userclass</td>
<td>User class</td>
<td>User class options set in [Options] &gt; [User Access Settings]</td>
</tr>
<tr>
<td>Property Name</td>
<td>Description</td>
<td>Value</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>---------------------</td>
<td>------------</td>
</tr>
<tr>
<td>voicemail.readcallerid</td>
<td>Talking caller ID</td>
<td>true or false</td>
</tr>
<tr>
<td>voicemail.password.enter.type</td>
<td>Skip password from my phone</td>
<td>0: no 1: yes</td>
</tr>
</tbody>
</table>

✓ `pln[n]_d_`xxx are properties’ names for the user extension [Inbound] page plan [n] Default Forwarding Schedule settings.

✓ `pln[n]_ptn[n]_`xxx are properties’ names for the user extension [Inbound] page plan [n] Forwarding Schedule [n] settings.

✓ User Forwarding Schedule [n] properties’ names are the same as those in Default Forwarding Schedule settings, but with a different prefix. For the Forwarding Schedule [n] Conditions properties’ names, please check the Schedule Extension table.
8. Sample Programs

8.1. Example 1

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

```javascript
<script>
var socket = null;
var host = 'ws://192.168.200.10:18080/pbx/ws?tenant=test&login_user=1000&login_password=1000';

if('WebSocket' in window){
    socket = new WebSocket(host);
} else if('MozWebSocket' in window){
    socket = new MozWebSocket(host);
}

socket.onmessage = function(event){
    var response = JSON.parse(event.data);
    var properties = ['name', 'desc', 'language', 'login.password', 'admin '];
    var request = '{"jsonrpc": "2.0", "method": "getExtensionProperties", "params": {"extension": "0001", "property_names":properties}, "id": 1};
    var json_request = JSON.stringify(request);
    socket.send(json_request);

    switch(response['id']){
        case '1':
            for(var i = 0; i < properties.length; i ++)
                console.log(properties[i] + ' = ' + response['result'][i]);
            break;
    }
</script>
```
8.2. Example 2

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

```html
<script>
  var socket = null;
  var host = 'ws://192.168.200.10:18080/pbx/ws?tenant=test&login_user=1000&login_password=1000';
  if('WebSocket' in window){
    socket = new WebSocket(host);
  }else if('MozWebSocket' in window){
    socket = new MozWebSocket(host);
  }
  socket.onmessage = function(event){
    var response = JSON.parse(event.data);
    var properties = {
      "name":"0001", "desc":"admin", "language":"en",
      "login.password":"0001", "admin":"true"};
    var request = '{"jsonrpc": "2.0", "method": "setExtensionProperties",
      "params": {"extension": "0001", "properties": properties}, "id":1};
    var json_request = JSON.stringify(request);
    socket.send(json_request);

    switch(response['id']){
      case '1':
        console.log(response['result']);
        break;
    }
  }
</script>
```