# Genesys Pulse Deployment Guide

## Genesys Pulse Web Service API

5/4/2025

# Contents

# Genesys Pulse Web Service API

Display external data in Genesys Pulse (or to display Pulse data in an external system) by using the Genesys Pulse RESTful (Representational State Transfer) Web Service API.

> **Important**
>
> - The API is subject to change at any time without notice.
> - Make sure you are authenticated before executing API methods. See the POST method under `/api/session/login` for details.

## /api/session/login

Methods:

### [+] POST

Authenticates a user.

Request body:

```
{
 "username": "your_username",
 "password": "your_password"
}
```

Available response representations:

- 204 - Successfully authenticated.
- 401 - Authentication failed, application/json with details.

## /api/session/logout

Methods:

### [+] GET

Invalidates the user's session.

Available response representations:

- 204 - Successfully invalidated.
- 401 - Authentication failed.

## /api/wbrt/templates

Methods:

**[+] GET**

Returns array of templates by specified parameters.

Example request URI(s):

- /api/wbrt/templates
- /api/wbrt/templates?uscn=1
- /api/wbrt/templates?uscn=1&type=ltGENERIC

Request query parameters

| name | required | type | description |
|---|---|---|---|
| uscn | no | long | specifies uscn, for filtering templates changed after this USCN (non inclusive) |
| type | no | string, available values:<br><br>ltGENERIC<br>ltPCREGULAR<br>ltPCPERFORMANCE<br>ltDATADEPOT<br>ltIFRAME<br>ltOTHER | specifies type |

Required permission:

- PULSE_READ or PULSE_READ_RESTRICTED or PULSE_READ_ALL_LAYOUTS

Available response representations:

- 200 - application/json with array of template.
- 400 - Request is not valid, application/json with details:
  - `{ "message": "INVALID_URL" }` - incorrect request parameters specified.
- 403 - User does not have privileges.

**[+] POST**

Creates new template.

Request query parameters

| name | required | type | description |
|---|---|---|---|
| overwrite | no | boolean | if true, then templates with the same name and saved to the same location will be removed |

By default template is created with proxy_access_object.dbid = 0 and shared for everyone without creating proxy access object.

If you want to control access, then specify empty proxy_access_object.dbid for creating proxy access object:

```
{
    "definition": {
        ...
        "proxy_access_object": {
        }
    }
}
```

You can specify dbid of folder where proxy access object must be created by proxy_access_object.folder_dbid. For folders in other tenant you have to specify proxy_access_object.tenant_dbid as well.
Instead of folder_dbid and tenant_dbid you can just specify proxy_access_object.path, for example: "path":"\\Configuration\\Environment\\Scripts"

Available request representations:

- application/json with template configuration. See LayoutInfo example.

Required permission:

- PULSE_WRITE_TEMPLATE

Available response representations:

- 200 - New template was successfully created, redirect to newly created template.
- 400 - Provided template configuration is not valid, application/json with details:
  - `{ "message": "JSON_PARSE_ERROR" }` - template has invalid format
  - `{ "message": "INVALID_LAYOUT_DEFINITION" }` - template definition is invalid
  - `{ "message": "PROXY_ACCESS_OBJECT_REQUIRED" }` - no proxy access object specified
  - `{ "message": "OBJECT_NAME_CONFLICT" }` - template with same name already exists in the folder. Retry with overwrite = true to remove duplicate templates.
- 403 - User does not have privileges.

## /api/wbrt/templates/<guid>

Methods:

**[+] GET**

Return template with specified id.

Required permission:

- PULSE_READ or PULSE_READ_RESTRICTED or PULSE_READ_ALL_LAYOUTS

Available response representations:

- 200 - application/json with requested template.
- 403 - User does not have privileges.

**[+] PUT**

Updates template with specified id.

> **Important**
> The request does not create new template if it does not exist.

Request query parameters

| name | required | type | description |
|---|---|---|---|
| overwrite | no | boolean | if true, then templates with the same name and saved to the same location are removed |

The method can be used to move template between folder. Execute with proxy_access_object.dbid = 0 to make it as shared for everyone, or specify target folder in proxy_access_object.folder_dbid. For moving between tenants also specify proxy_access_object.tenant_dbid.

Available request representations:

- application/json with template configuration.

Required permission:

- PULSE_WRITE_TEMPLATE

Available response representations:

- 200 - Template was updated

- 400 - Provided template configuration is not valid, application/json with details:
  - `{ "message": JSON_PARSE_ERROR }` - template has invalid format
  - `{ "message": INVALID_LAYOUT_DEFINITION }` - new template definition is invalid
  - `{ "message": "OBJECT_NAME_CONFLICT" }` - template with same name already exists in the folder. Retry with overwrite = trueto remove duplicate templates.
  - `{ "message": "CONCURRENT_DB_MODIFICATION" }` - uscn value provided in the request is older than uscn of the modifing object on the server. Remove the uscn from the request to overwrite the saved object or set the correct uscn value and retry.
- 403 - User does not have privileges.
- 404 - Template does not exist, application/json with details:
  - `{ "message": "TEMPLATE_NOT_FOUND" }` - template not exists

**[+] DELETE**

Delete template with specified id.

Required permission:

- PULSE_WRITE_TEMPLATE

Available response representations:

- 200 - Template was deleted
- 204 - Template was already deleted
- 403 - User does not have privileges.

## /api/wbrt/layouts

Methods:

**[+] GET**

Returns array of layouts by specified parameters.

Example request URI(s):

- /api/wbrt/layouts
- /api/wbrt/layouts?uscn=1
- /api/wbrt/layouts?uscn=1&type=ltGENERIC

Request query parameters

| name | required | type | description |
|---|---|---|---|
| uscn | no | long | specifies uscn, for |

| name | required | type | description |
|---|---|---|---|
| | | | filtering layouts changed after this USCN (non inclusive) |
| type | no | string, available values:<br><br>ltGENERIC<br>ltPCREGULAR<br>ltPCPERFORMANCE<br>ltDATADEPOT<br>ltIFRAME<br>ltOTHER | specifies layout type |

Required permission:

- PULSE_READ_ALL_LAYOUTS

Available response representations:

- 200 - application/json with array of layout.
- 400 - Request is not valid, application/json with details:
  - `{ "message": "INVALID_URL" }` - incorrect request parameters specified.
- 403 - user doesn't have privileges.

**[+] POST**

Creates new layout.

Available request representations:

- application/json with layout configuration. See LayoutInfo example.

Required permission:

- PULSE_WRITE_LAYOUT

Available response representations:

- 200 - New layout was successfully created, redirect to newly created layout.
- 400 - Provided layout configuration is not valid, application/json with details:
  - `{ "message": "JSON_PARSE_ERROR" }` - layout has invalid format
  - `{ "message": "INVALID_LAYOUT_DEFINITION" }` - layout definition is invalid
  - `{ "message": "CONCURRENT_DB_MODIFICATION" }` - uscn value provided in the request is older than uscn of the modifing object on the server. Remove the uscn from the request to overwrite the saved object or set the correct uscn value and retry.
- 403 - User does not have privileges.

## /api/wbrt/layouts/<guid>

Methods:

**[+] GET**

Return layout with specified id.

Required permission:

- PULSE_READ or PULSE_READ_RESTRICTED or PULSE_READ_ALL_LAYOUTS

Available response representations:

- 200 - application/json with requested layout.
- 403 - User does not have privileges.
- 404 - Layout does not exist or current user doesn't have any widgets for this layout, but there are some widgets of other users (layout owned by some other user)

**[+] PUT**

Updates layout with specified id. If there are more than one widget for this layout, then the new layout is created and returned in response.

> **Important**
> The request does not create new template if it does not exist.

Available request representations:

- application/json with layout configuration. See LayoutInfo example.

Required permission:

- PULSE_WRITE_LAYOUT

Available response representations:

- 200 - Layout was updated, redirect to new newly created layout if any
- 400 - Provided layout configuration is not valid, application/json with details:
  - `{ "message": "JSON_PARSE_ERROR" }` - layout has invalid format
  - `{ "message": "INVALID_LAYOUT_DEFINITION" }` - new layout definition is invalid
  - `{ "message": "CONCURRENT_DB_MODIFICATION" }` - uscn value provided in the request is older than uscn of the modifing object on the server. Remove the uscn from the request to overwrite the saved object or set the correct uscn value and retry.

- 403 - User does not have privileges.
- 404 - Layout does not exist, application/json with details:
  - `{ "message": "LAYOUT_NOT_FOUND" }` - layout does not exist

**[+] DELETE**

Delete layout with specifiifed id.

Required permission:

- PULSE_WRITE_LAYOUT

Available response representations:

- 200 - Layout was deleted
- 204 - Layout was already deleted
- 400 - could not remove specified layout, application/json with details:
  - `{ "message": "FIRST_DELETE_RELATED_WIDGETS" }` - try to remove layout before related widgets.
- 403 -In case if user doesn't have privileges.

## /api/wbrt/layouts/<guid>/snapshot

Methods:

**[+] GET**

Returns recent snapshot for specified layout.

NOTE: If user does not have "Pulse Read All Layouts" privilege then this method performs additional rows filtering based on user access restrictions for snapshots with layout_type = **ltPCREGULAR** and layout_type = **ltPCPERFORMANCE**. Rows with objects not accessible for the user are filtered out. It must work as follows: if there is column _Object$CfgType, then use pair (_Object$CfgType, _Object$ID) to check permissions, otherwise attempt the usual combination (_Object$Type, _Object$ID).

Required permission:

- PULSE_READ or PULSE_READ_RESTRICTED or PULSE_READ_ALL_LAYOUTS

Available response representations:

- 200 - application/json snapshot.
- 204 - no content, snapshot not exists
- 403 - User does not have privilegess.
- 404 - Related resource does not exist, application/json with details:

- `{ "message": "LAYOUT_NOT_FOUND" }` - layout does not exist

**[+] POST**

Saves the snapshot for the layout with the specified id. Uses the timestamp property from the LayoutSnapshot to distinguish a different snapshot. If there is already a saved snapshot with the same timestamp, then it is overwritten.

Available request representations:

- application/json with snapshot. See LayoutSnapshot example

Required permission:

- PULSE_WRITE_SNAPSHOT

Available response representations:

- 200 - snapshot was successfully saved
- 400 - snapshot is not valid, application/json with details:
  - `{ "message": "SNAPSHOT_PARSE_ERROR" }` - snapshot has invalid format
  - `{ "message": "INVALID_BODY_HASH" }` - state.body_hash_1 in layout does not match state.body_hash_1 in the snapshot
- 403 - User does not have privileges.
- 404 - Related resource does not exist, application/json with details:
  - `{ "message": "LAYOUT_NOT_FOUND" }` - layout does not exist

## /api/wbrt/layouts/<guid>/snapshots

Methods:

**[+] GET**

Returns array of snapshots for specified layout and matched filter period. If start and end are not specified then returns only latest snapshot.

Example request URI(s):

- /api/wbrt/layouts/1/snapshots
- /api/wbrt/layouts/1/snapshots?start=1411720605
- /api/wbrt/layouts/1/snapshots?start=1411720605&columns=Inbound_Talk_Time,Internal_Talk_Time
- /api/wbrt/layouts/1/snapshots?start=140000000&frequency=10

Request query parameters

| name | required | type | description |
|---|---|---|---|
| start | no | long | Unix epoch timestamp indicating the start of period for which snapshots should be returned. If not specified, then all snapshots generated before end time are returned. |
| end | no | long | Unix epoch timestamp indication end of period for which snapshots should be returned. If not specified, then all snapshots generated after the start time are returned. |
| frequency | no | integer | Minimum interval between two snapshots in history in seconds. Needed for reducing amount of snapshots returned by this request. Default value is 0, so all existing snapshots inside specified period are returned. |
| columns | no | Array of Strings | Array of Column.id that is to be included in the snapshot. If not specified, then all columns are included. |

> **Important**
>
> This method performs additional row filtering based on user access restrictions for snapshots with layout_type = **ltPCREGULAR** and layout_type = **ltPCPERFORMANCE**. Rows with objects that are not accessible for a user are filtered out. It must work as follows: if there is column _Object$CfgType, then use pair (_Object$CfgType, _Object$ID) to check permissions; otherwise, attempt usual combination (_Object$Type, _Object$ID).

Required permission:

- PULSE_READ or PULSE_READ_RESTRICTED or PULSE_READ_ALL_LAYOUTS

Available response representations:

- 200 - application/json with array of snapshots. See LayoutSnapshot example.

- 403 - User does not have privileges.

## /api/wbrt/widgets

Methods:

**[+] GET**

Returns array of widget by specified parameters.

Example request URI(s):

- /api/wbrt/widgets
- /api/wbrt/widgets?uscn=1

Request query parameters

| name | required | type | description |
|---|---|---|---|
| uscn | no | long | specifies uscn, for filtering widgets changed after this USCN (non inclusive) |

Required permission:

- PULSE_READ or PULSE_READ_RESTRICTED

Available response representations:

- 200 - application/json with array of widgets. See Widget example.
- 403 - User does not have privileges.

**[+] POST**

Create new widget for specified layout.

Available request representations:

- application/json with widget configuration. See Widget example.

Request query parameters:

| name | required | type | default | description | |
|---|---|---|---|---|---|
| brief | no | boolean | true | when set to `false`, the related layout is expected to be in the widget request; | |

| name | required | type | default | description | |
|---|---|---|---|---|---|
| | | | | also allows saving server-side alerts related to the widget | |

Required permission:

- PULSE_WRITE_WIDGET

Available response representations:

- 200 - New widget was successfully created, redirect to newly created widget.
- 400 - Provided widget configuration is not valid, application/json with details:
  - `{ "message": "JSON_PARSE_ERROR" }` - widget has invalid format
  - `{ "message": "WIDGETS_LIMIT_EXCEEDED" }` - widgets limit exceeded
  - `{ "message": "LAYOUT_DEFINITION_REQUIRED" }` - brief = `false` and related layout is not presented in the widget request
  - `{ "message": "LAYOUT_NOT_EXPECTED_IN_BRIEF_MODE" }` - brief = `true` and related layout is presented in the widget request
  - `{ "message": "ALERTS_NOT_EXPECTED_IN_BRIEF_MODE" }` - brief = `true` and server-side alert is presented in the widget request
- 403 - User does not have privileges.
- 404 - Widget configuration data does not exist, application/json with details:
  - `{ "message": "LAYOUT_NOT_FOUND" }` - specified widget layout does not exist

## /api/wbrt/widgets/<guid>

Methods:

**[+] GET**

Returns widget configuration for widget with specified id and belonging to user or default widget.

Request query parameters:

| name | required | type | default | description | |
|---|---|---|---|---|---|
| brief | no | boolean | true | When set to `false`, the widget is returned with layout definition and server-side | |

| name | required | type | default | description | |
|---|---|---|---|---|---|
| | | | | alerts related to the widget. Otherwise just widget is returned. | |

Required permission:

- PULSE_READ or PULSE_READ_RESTRICTED

Available response representations:

- 200 - application/json with widget configuration. See Widget example.
- 403 - User does not have privileges.
- 404 - Widget with requested id does not exist, application/json with details:
  - `{ "message": "WIDGET_NOT_FOUND" }`

**[+] PUT**

Update widget configuration for specified widget.

> **Important**
> The request does not create new widget if it does not exist

Request query parameters:

| name | required | type | default | description | |
|---|---|---|---|---|---|
| brief | no | boolean | true | When set to `false`, the related layout is expected to be in the widget request; also allows saving server-side alerts related to the widget | |

Available request representations:

- application/json with widget configuration. See Widget example.

Required permission:

- PULSE_WRITE_WIDGET

Available response representations:

- 200 - Widget was successfully updated
- 400 - Provided widget configuration is not valid, application/json with details:
  - `{ "message": "JSON_PARSE_ERROR" }` - widget has invalid format
  - `{ "message": "INVALID_LAYOUT_HASH" }` - specified layout has differ body_hash than provided in snapshot
  - `{ "message": "CONCURRENT_DB_MODIFICATION" }` - uscn value provided in the request is older than uscn of the modifing object on the server. Remove the uscn from the request to overwrite the saved object or set the correct uscn value and retry.
  - `{ "message": "LAYOUT_DEFINITION_REQUIRED" }` - brief = `false` and related layout is not presented in the widget request
  - `{ "message": "LAYOUT_NOT_EXPECTED_IN_BRIEF_MODE" }` - brief = `true` and related layout is presented in the widget request
  - `{ "message": "ALERTS_NOT_EXPECTED_IN_BRIEF_MODE" }` - brief = `true` and server-side alert is presented in the widget request
- 403 - User does not have privileges.
- 404 - Widget configuration data or widget itself does not exist, application/json with details:
  - `{ "message": "WIDGET_NOT_FOUND" }` - specified widget not exist
  - `{ "message": "LAYOUT_NOT_FOUND" }` - specified widget layout does not exist

**[+] DELETE**

Delete widget with specified guid.

Required permission:

- PULSE_WRITE_WIDGET

Available response representations:

- 200 - Widget was deleted
- 204 - Widget was already deleted
- 403 - User does not have privileges.

## /api/wbrt/tabs

Methods:

**[+] GET**

Returns array of tabs for current user.

Request query parameters:

| name | required | type | default | description | |
|---|---|---|---|---|---|
| brief | no | boolean | true | When set to `true`, returns only the set of tabs without widgets. Otherwise, returns tabs with the `widget` field containing full widget definition, including layout and related server-side alerts for all widget's presented on the tab. | |
| shared | no | boolean | false | When set to `true`, returns only shared dashboards (tabs with type `ttDashboard`). Otherwise, returns user tabs.<br>User cannot change or delete widgets placed to shared dashboard unless he is owner of this shared dashboard. | |
| type | no | string | | Optional filter for filtering by type, possible values: `ttDashboard`, `ttWidget`, `ttWallboard`. | |

Required permission:

- PULSE_READ or READ_RESTRICTED

Available response representations:

- 200 - application/json with array of tabs.

- 403 - User does not have privileges.

**[+] POST**

Creates new tab.

Request query parameters:

| name | required | type | default | description |
|---|---|---|---|---|
| brief | no | boolean | false | When set to `true`, widgets are not created according to widgets definitions (only positions are saved). Otherwise creates new widgets according to widgets definitions, each widget contains the related layout definition. |
| shared | no | boolean | false | When set to `true`, creates shared tab, which does not belong only to the current user. Otherwise, creates the tab which is available only for current user. |
| overwrite | no | boolean | false | When set to `true`, removes tabs with the same name and saved to the same location (only for shared tabs). |

If you are going to create shared tab, then proxy_access_object field must be specified:

- proxy_access_object.dbid = 0 for sharing for everyone without creating proxy access object.
- empty proxy_access_object.dbid for creating proxy access object to control access:

```
{
    "body": {
        ...
        "proxy_access_object": {
        }
    }
}
```

You can specify dbid of folder where proxy access object must be created by proxy_access_object.folder_dbid. For folders in other tenant you have to specify proxy_access_object.tenant_dbid as well.

Available request representations:

- application/json with tab.

Required permission:

- PULSE_WRITE_TAB - for personal tab
- PULSE_SHARE_TAB - for shared tab

Available response representations:

- 200 - Tab was successfully created. Redirect to newly created tab. See Tab example.
- 400 - Provided tabs configuration is not valid, application/json with details:
  - `{ "message": "JSON_PARSE_ERROR" }` - request has invalid format.
  - `{ "message": "PROXY_ACCESS_OBJECT_REQUIRED" }` - request with shared = true but has no proxy access object specified.
  - `{ "message": "OBJECT_NAME_CONFLICT" }` - tab with same name already exists in the folder. Retry with overwrite = `true` to remove duplicate tabs.
  - `{ "message": "TABS_LIMIT_EXCEEDED" }` - tabs limit exceeded.
- 403 - User does not have privileges.

**[+] PUT**

> **Important**
> Only for use with personal tabs. Please read method behaviour before using.

Stores array of tabs for current user in the following way:

1. Remove all tabs belonging to current user but not presented in the request.
2. Remove all widgets related to the removed tabs.
3. Update existing tabs belonging to current user and specified in the request.
4. Remove all existing but not presented in the request widgets.
5. Create not existing tabs but specified in the request.
6. If brief != `ture` create/update widgets with widget definitions.

Request with empty array removes all personal tabs. Request query parameters:

| name | required | type | default | description | |
|---|---|---|---|---|---|
| brief | no | boolean | false | When set to `true`, widgets are not created/ updated with widgets definitions (only positions are saved). Otherwise creates/ updates widgets according to widgets definitions, each widget is expected to contain related layout definition. | |

Available request representations:

- application/json with tab **array**.

Required permission:

- PULSE_WRITE_TAB - for personal tab
- PULSE_SHARE_TAB - for shared tab

Available response representations:

- 200 - User's tabs were saved. Redirect to to the updated tab. See Tab example.
- 400 - Provided tabs configuration is not valid, application/json with details:
  - `{ "message": "JSON_PARSE_ERROR" }` - the request has an invalid format
  - `{ "message": "LAYOUT_DEFINITION_REQUIRED" }` - brief = `false` and related layout is not presented in the request
  - `{ "message": "LAYOUT_NOT_EXPECTED_IN_BRIEF_MODE" }` - brief = `true` and related layout is presented in the request
  - `{ "message": "ALERTS_NOT_EXPECTED_IN_BRIEF_MODE" }` - brief = `true` and server-side alert is presented in the request
  - `{ "message": "WIDGETS_NOT_EXPECTED_IN_BRIEF_MODE" }` - brief = `true` and widget is presented in the request
- 403 - User does not have privileges.

## /api/wbrt/tabs/<guid>

Methods:

**[+] GET**

Returns tab with specified guid.

Request query parameters:

| name | required | type | default | description | |
|---|---|---|---|---|---|
| brief | no | boolean | false | When set to `true`, returnes tab without widgets definition (only positions property filled). Otherwise, returns tab with field `widget`, containing full widget definitions (including related layout and related server-side alerts) for all widget's presented on the tab. | |

Required permission:

- PULSE_READ or READ_RESTRICTED

Available response representations:

- 200 - application/json with tab. See Examples.
- 403 - User does not have privileges.
- 404 - Tab does not exist or user doesn't have access to it.

**[+] PUT**

Update tab with specified guid. This method removes widgets and their layouts from old tab if newer tab doesn't have references to them.

> **Important**
> The request does not create new tab if it does not exist.

| name | required | type | default | description |
|---|---|---|---|---|
| brief | no | boolean | false | If specified and "true" then widgets will not be created\updated with widgets definitions (only positions will be saved).<br><br>Otherwise will create\ update widgets according to widgets definitions. |
| overwrite | no | boolean | false | if true then tabs with same name and saved to same location will be removed (only for shared tabs) |

The method can be used to move tab between folder. Execute with proxy_access_object.dbid = 0 to make it as shared for everyone, or specify target folder in proxy_access_object.folder_dbid. For moving between tenants also specify proxy_access_object.tenant_dbid.

Available request representations:

- application/json with tab.

Required permission:

- PULSE_WRITE_TAB - for personal tab
- PULSE_SHARE_TAB - for shared tab

Available response representations:

- 200 - Tab was updated successfully.
- 400 - Provided tabs configuration is not valid, application/json with details:
  - `{ "message": "JSON_PARSE_ERROR" }` - request has invalid format
  - `{ "message": "CONCURRENT_DB_MODIFICATION" }` - uscn value provided in the request is older than uscn of the modifing object on the server. Remove the uscn from the request to overwrite the saved object or set the correct uscn value and retry.
  - `{ "message": "OBJECT_NAME_CONFLICT" }` - tab with same name already exists in the folder. Retry with overwrite = `true` to remove duplicate tabs.
  - `{ "message": "LAYOUT_DEFINITION_REQUIRED" }` - brief = `false` and related layout is not presented in the request.

  - `{ "message": "LAYOUT_NOT_EXPECTED_IN_BRIEF_MODE" }` - brief = `true` and related layout is presented in the request.
  - `{ "message": "ALERTS_NOT_EXPECTED_IN_BRIEF_MODE" }` - brief = `true` and server-side alert is presented in the request.
  - `{ "message": "WIDGETS_NOT_EXPECTED_IN_BRIEF_MODE" }` - brief = `true` and widget is presented in the request.
- 403 - User does not have privileges or access to change this tab.
- 404 - Tab with specified guid does not exist or user does not have read access to this tab.

**[+] DELETE**

Delete tab with specified guid. This method removes related widgets and their layouts before removing the tab.

Required permission:

- PULSE_WRITE_TAB

Available response representations:

- 200 - Tab was successfully deleted.
- 204 - Tab is already deleted or user doesn't have read access to this tab.
- 403 - User does not have privileges or access to delete this tab.

## /api/wbrt/users

Methods:

**[+] GET**

Returns all Genesys Pulse users.

Required permission:

- PULSE_WRITE_USER

Available response representations:

- 200 - Application/json with users.
- 403 - User does not have privileges.

## /api/wbrt/users/<guid>

Methods:

**[+] GET**

Returns user with specified guid.

Request query parameters:

| name | required | type | default | description | |
|---|---|---|---|---|---|
| brief | no | boolean | true | When set to `false`, the user is returned with the `tab` field containing full personal tabs definitions. When not specified or set to `true`, the user is returned without personal tabs definitions. | |

Required permission:

- PULSE_WRITE_USER

Available response representations:

- 200 - Application/json with user.
- 403 - User does not have privileges.
- 404 - User does not exist.

## Example

**brief**=`true`

```
{
    "body": {
        "guid": "5efe4b7c07b3-b636-11ea-fe84-6719e62a",
        "username": "default",
        "tab_guid": [
            "961f0facee94-a9d4-11ea-fe84-681b5081",
            "f22c1537b09b-a43b-11eb-0d5e-69cb9b37"
        ],
        "widgets_number": 1,
        "preferences": {
            "language": "en-us",
            "time_zone": ""
        }
    }
}
```

### [+] DELETE

Delete a user with the specified guid. This method removes related tabs, widgets, and their layouts before removing the user.

Required permission:

- PULSE_WRITE_USER

Available response representations:

- 200 - User was successfully deleted.
- 204 - User is already deleted.
- 403 - User does not have privileges.

## /api/wbrt/import

Methods:

### [+] POST

Import entities provided in request body.

Example of request body for importing data exported with the export service:

```
{
    "data": <result of export>
}
```

Example of request body for importing templates:

```
{
    "data": {
                "template": [{
                        "definition": <template_definition>
                }, {
                        "definition": <template_definition>
                }, ...]
    }
}
```

Example of request body for importing tabs:

```
{
    "data": {
                "tab": [{
                        "body": <template_body>
                }, {
                        "body": <template_body>
                }, ...],
                "widget": [{
                        "body": <widget_body>
                }, {
```

```
                "body": <widget_body>
            }, ...],
            "layout": [{
                "definition": <layout_definition>
            }, {
                "definition": <layout_definition>
            }, ...]
    }
}
```

Required permission:

- PULSE_WRITE_TEMPLATE - for importing templates
- PULSE_SHARE_TAB - for importing tabs

Available response representations:

- 200 - application/json with imported entities.
- 400 - Provided import data is not valid, application/json with details:
  - `{ "message": "JSON_PARSE_ERROR" }` - request has invalid format
  - `{ "message": "OBJECT_NAME_CONFLICT" }` - entity with same name already exists in the folder. Retry with specified overwrite:
    opOverwrite - overwrite old entity in case of name conflict
    opSkip - skip in case of conflict
    opRename - rename new entity in case of name conflict
- 403 - User does not have privileges or access permissions to import one or more entity to the destination folder.

## /api/wbrt/export

Methods:

**[+] POST**

Export entities according to the query provided in request body.

Example of request body for exporting all Wallboards:

```
{
    "tab": [{"type": "ttWallboard"}]
}
```

Example of query for exporting all shared Dashboards and Templates:

```
{
    "tab": [{"type": "ttWallboard", "proxy_access_object": {}}],
    "template": [{}]
}
```

Example of query for exporting Dashboards\Wallboards with given guids:

```
{
```

```
    "tab": [{"guid": "<guid_1>"}, {"guid": "<guid_2>"}]
}
```

Example of query for exporting Templates with given guids:

```
{
    "template": [{"guid": "<guid_1>"}, {"guid": "<guid_2>"}]
}
```

Example of query for exporting all templates with certain type:

```
{
    "template": [{"layout_type": "ltPCREGULAR"}]
}
```

Required permission:

- PULSE_READ or PULSE_READ_RESTRICTED

Available response representations:

- 200 - application/json with export result.
- 403 - User does not have privileges.

## /api/plugins/wbrt/health

Methods:

**[+] GET**

Checks Genesys Pulse application's health. Genesys Pulse is determined as healthy when:

- There is a working connection to Genesys Pulse Database.
- There is a working connection to Configuration Server.

This service caches the previous check result and keeps it according to the value of the **pulse/health_expire_timeout** option.

Health of Genesys Pulse Collectors is not checked here, use use /api/plugins/wbrt/health/detail to get health information about

Required permission:

- no permissions required

Available response representations:

- 200 - Genesys Pulse application works fine.
- 503 - Genesys Pulse health check failed, please, use use /api/plugins/wbrt/health/detail service to get

more information.

## /api/plugins/wbrt/health/detail

Methods:

**[+] GET**

Returns Genesys Pulse application health details.

Genesys Pulse application health details include:

- State of connection to Configuration Server.
- State of connection to the Genesys Pulse Database.
- State of Genesys Pulse Collectors and their health details, including:
  - State of connection to Configuration Server.
  - State of connection to the Genesys Pulse Database.
  - State of connection to Stat Servers.
  - Starting with release 9.0.004.03, state of Stat Server utilization:
    - Total Statistic Count - total number of statistic requests that are either already opened, or pending, or not sent to Stat Server yet.
    - Opened Statistic Count - the number of requests that are successfully opened.
    - Failed Statistic Count - the number of requests that failed.

This service caches the previous check result and keeps it according to the value of the **pulse/health_expire_timeout** option.

Required permission:

- no permissions required

Available response representations:

- 200 - application/json with Genesys Pulse application health details. Including:
  - **snapshotWritingStatus**—Snapshots are written to the file system. The snapshot generation and writing process itself does not depend on present/missing connections. If a connection to Stat Server is missing, the data may be all N/A, but snapshots are still generated and written.
  - **pendingLayoutStatusChangesCount**—The number of pending layouts to activate, deactivate, or recheck. It depends on the connectivity to DB Server. If the connection is present, this number changes, as layout are being activated/deactivated/rechecked. Stalled nonzero number may indicate missing connection to DB Server. Genesys Pulse Collector needs to read changed layouts and new layout definitions, which requires active connection to DB Server.
  - **maxStatisticValueDelay**—This value depends on the number of layouts, their refresh period, performance of CPU, sufficient memory availability (no swap operations caused by Genesys Pulse Collector), performance of the file system. There is no recommended value; however, most layouts

typically have the same refresh interval. If the setting of the `maxStatisticValueDelay` is higher than some part of this interval (say 25-50%), then you want to consider increasing resources. Values do not indicate that something is totally wrong, but may indicate that Genesys Pulse Collector can perform better if it runs with more elevated settings (more processing threads) and faster drives/file systems for the snapshots storage.

**Example of the service response:**


```
{
 "configServer":{
  "connected":true
 },
 "database":{
  "connected":true
 },
 "collectors":{
  "Collector_9.0.006":{
   "currentTimestamp": "1600369952",
   "startupTimestamp": "1600159972",
   "uptime": "209980",
   "connectionStatus": {
    "configServer": {
     "connected": true,
     "instance": "primary",
     "connectionTimestamp": "1600159973",
     "disconnectTimestamp": "0"
    },
    "statServer0": {
     "present": true,
     "connected": false,
     "connectionTimestamp": "0",
     "disconnectTimestamp": "1600159974",
     "totalStatisticCount": "60",
     "openedStatisticCount": "0",
     "failedStatisticCount": "0"
    },
    "statServer1": {   "present": true,   "connected": false,   "connectionTimestamp": "0",
"disconnectTimestamp": "1600159974",   "totalStatisticCount": "60",   "openedStatisticCount":
"0",   "failedStatisticCount": "0"
    },
    "dbServerConnection": "online",
    "dbConnection1": {
     "connected": true,
     "connectionTimestamp": "1600159975",
     "disconnectTimestamp": "0"
    },
     "dbConnection2": {
      "connected": true,
      "connectionTimestamp": "1600159975",
      "disconnectTimestamp": "0"
     }
    },
   "snapshotWritingStatus": true,
   "pendingLayoutStatusChangesCount": "0",
   "maxStatisticValueDelay": 0,
   "version": "2019.10.10.00"
  }
 }
}
```

# Troubleshooting

## [+] Snapshot was populated, but related Widget shows only spinner instead of data from the snapshot.

LayoutState message has special fields body_hash_1/body_hash_2.

These fields are intended to indicate compatibility between LayoutDefinition and LayoutSnapshot.

So body_hash_1/body_hash_2 are populated in LayoutInfo.state (with hash value calculated for LayoutDefinition) each time when LayoutInfo is saved (via POST/PUT or via Widget wizard).

Same values must be inserted to LayoutSnapshot.state by data producer. This lets Genesys Pulse know that data from LayoutSnapshot is legal for current state of LayoutDefinition.

### Important

Starting with **release 8.5.104.05**, Genesys Pulse validates value of body_hash_1 in snapshot against value in the layout.

So when you trying to post snapshot with incorrect body_hash_1 value you will get 400 BAD REQUEST with message INVALID_BODY_HASH in the response body.

You can ommit body_hash_1 value in the snapshot and it will not be used for detecting layout change.

Then widget will continue to show data from such snapshot even something changed in the layout.

For example if you select/unselect some objects widget will show these objects until new snapshot without them is generated by Genesys Pulse Collector.

The widget is updated too infrequently or too frequently

LayoutDefinition message has special refresh_interval field indicates how often data producer must provide new data.

The same refresh_interval field is presented in the LayoutSnapshot and indicates how often data producer provides data actually.

Genesys Pulse UI uses LayoutSnapshot.refresh_interval for calculation of delay before each next data request:

### Refresh Delay Calculation

```
var delay = snapshot.refresh_interval || 60, //use refresh interval from data provider or 60
sec
    gap = 2,
    genTime = snapshot.timestamp, //must be filled by data provider
        readTime = snapshot.read_timestamp; //filled by Genesys Pulse BE on each snapshot
request

var d = readTime - genTime - gap;
if ((d > 0) && (d < delay)) {
```

```
    delay = delay - d;
}
```

> **Important**
>
> LayoutSnapshot.refresh_interval field must be provided by data producer. It can be smaller/greater than in related LayoutDefinition.

## Examples

**[+] LayoutInfo Example**

```
{
    "definition": {
        "guid": "fe1157d5-d707-4135-be83-fd79f3243ddc",
        "refresh_interval": 15,
        "layout_type": "ltGENERIC",
        "column": [
            {
                "category": "ccDIMENSION",
                "is_delta_key": true,
                "id": "_Object$ID"
            },
            {
                "category": "ccDIMENSION",
                "id": "_Object$Name",
                "format": "string"
            },
            {
                "category": "ccMEASURE",
                "vt": "vINT",
                "id": "First_column",
                "type": "ctGENERIC",
                "format": "integer",
                "label": "First"
            },
            {
                "category": "ccMEASURE",
                "vt": "vINT",
                "id": "Second_column",
                "type": "ctGENERIC",
                "format": "time",
                "label": "Second"
            },
            {
                "category": "ccMEASURE",
                "vt": "vSTR",
                "id": "Third_column",
                "type": "ctGENERIC",
                "format": "string",
                "label": "Third"
            }
        ],
        "default_widget": {
            "threshold": [
```

```
                {
                    "value": [
                        0,
                        0,
                        0
                    ],
                    "direction_up": false,
                    "column_id": "First_column"
                }
            ],
            "view": [
                {
                    "column_selector": [
                        "Second_column"
                    ],
                    "sorting": [
                        {
                            "is_asc": true
                        }
                    ],
                    "type": "BarView"
                }
            ],
            "size_x": 1,
            "size_y": 4,
            "label": "Test  Widget"
        },
        "name": "Third party source",
        "description": "Example of third party source"
    },
    "state": {
        "current_status": "stACTIVE",
        "body_hash_1": 583854940,
        "body_hash_2": 583854940
    },
    "record": {
        "timestamp": 1443695496,
        "username": "pulse"
    }
}
```


### [+] LayoutSnapshot Example


```
{
    "layout_guid": "fe1157d5-d707-4135-be83-fd79f3243ddc",
    "col": [
        {
            "v": [
                101,
                102,
                103,
                104,
                105,
                106,
                107,
                108,
                109,
                110
            ],
            "col": {
                "category": "ccDIMENSION",
                "is_delta_key": true,
                "vt": "vINT",
```

```
            "id": "_Object$ID"
        }
    },
    {
        "v": [
            "Agent1",
            "Agent2",
            "Agent3",
            "Agent4",
            "Agent5",
            "Agent6",
            "Agent7",
            "Agent8",
            "Agent9",
            "Agent10"
        ],
        "col": {
            "category": "ccDIMENSION",
            "vt": "vSTR",
            "id": "_Object$Name"
        }
    },
    {
        "v": [
            60,
            120,
            180,
            234,
            123,
            531,
            521,
            231,
            541,
            634
        ],
        "col": {
            "category": "ccMEASURE",
            "vt": "vINT",
            "id": "First_column",
            "type": "ctGENERIC",
            "format": "integer",
            "label": "First"
        }
    },
    {
        "v": [
            523,
            0,
            312,
            543,
            631,
            434,
            423,
            642,
            743,
            135
        ],
        "col": {
            "category": "ccMEASURE",
            "vt": "vINT",
            "id": "Second_column",
            "type": "ctGENERIC",
            "format": "time",
```

```
                "label": "Second"
            }
        },
        {
            "v": [
                "Value1",
                "Value2",
                "Value3",
                "Value4",
                "Value5",
                "Value6",
                "Value7",
                "Value8",
                "Value9",
                "Value10"
            ],
            "col": {
                "category": "ccMEASURE",
                "vt": "vSTR",
                "id": "Third_column",
                "type": "ctGENERIC",
                "format": "string",
                "label": "Third"
            }
        }
    ],
    "state": {
        "current_status": "stACTIVE",
        "body_hash_1": 583854940,
        "body_hash_2": 583854940
    },
    "layout_type": "ltGENERIC",
    "timestamp": 1409066836
}
```

### [+] Widget Example

```
{
        "body": {
                "guid": "28cb32f0-6d8c-4be9-9ee5-1e2fa5f91db2",
            "layout_guid": "fe1157d5-d707-4135-be83-fd79f3243ddc",
            "label": "MyWidget",
            "size_x": 1,
            "size_y": 4,
            "view": [{
                "type": "BarView",
                "column_selector": ["Login_Duration"],
                "sorting": [{
                    "is_asc": false
                }]
            }]
    }
}
```

### [+] Tab Example

```
{
    "body": {
        "guid": "c8cb42f0-1d8c-4be9-9ee5-1e2fa5f91db2",
        "title": "Dashboard",
        "type": "ttDashboard",
        "position": [{
```

```
            "widget_guid": "c4bff0ce-d842-4070-be48-36a2cf6e0630",
            "col": 1,
            "row": 1
        }],
        "widget": [{
            "body": {
                "guid": "c4bff0ce-d842-4070-be48-36a2cf6e0630",
                "layout_guid": "fe1157d5-d707-4135-be83-fd79f3243ddc",
                "label": "MyWidget",
                "size_x": 1,
                "size_y": 4,
                "view": [{
                    "type": "BarView",
                    "column_selector": ["Login_Duration"],
                    "sorting": [{
                        "is_asc": false
                    }]
                }]
            }
        }]
    },
    "record": {
        "timestamp": 1443695496,
        "username": "pulse"
    }
}
```


## [+] Tab Example (brief=true)


```
{
    "body": {
        "guid": "c8cb42f0-1d8c-4be9-9ee5-1e2fa5f91db2",
        "title": "Dashboard",
        "type": "ttDashboard",
        "position": [{
            "widget_guid": "c4bff0ce-d842-4070-be48-36a2cf6e0630",
            "col": 1,
            "row": 1
        }]
    },
    "record": {
        "timestamp": 1443695496,
        "username": "pulse"
    }
}
```