iWD REST API Reference Guide

intelligent Workload Distribution 9.0.0

4/21/2024

Table of Contents

IWD REST API Reference Guide

Welcome to the IWD REST API Reference. This guide provides information about how you can use the IWD REST API to incorporate Genesys IWD Manager features into custom applications and integrations with third-party software.

This API is based on HTTP and has some properties of REST.

Resources

The API defines the following resources:

- Tasks
- Task Attributes (for filter definitions)
- Filters
- Media Icons
- Tenants
- Business Structure (Solutions, Departments, Processes, Capture Points)
- Security (for login/logout)

Verbs

Standard REST verbs are used:

- GET
- POST
- PUT
- DELETE

Messages

If not written explicitly, the Content-Type for requests and responses is "application/json". Error responses may contain a list of messages, for example:

```
[
  {
    "severity" : "ERROR",
    "message_id" : "MSG_COULD_NOT_LOGIN_TO_PLACE",
```

```
    "args" :
     {
      "<name>": "<value>",
      ...
     },
    "message": "English message to log/debug"
  }
]
```

Severity may be one of:

- ERROR
- WARNING
- INFO.

message_id is a key that should be replaced with a localized message in the front end.

Date Format

All date/time values are in ISO8601 format with time zone. The exact supported variants of ISO8601 format are:

- yyyy-MM-dd'T'HH:mm:ss.SSSZZ
- yyyy-MM-dd'T'HH:mm:ssZZ

Time zone offset is mandatory. Examples:

- 2015-12-16T09:10:50.000+01:00
- 2015-10-01T00:00:00.000Z
- 2015-12-16T07:10:50-01:00

Additionally, for quick filters in GTL, a shortened, date-only format is allowed:

- yyyy-MM-dd['T'ZZ]

Time zone offset is optional in this case; if it is missing, the default configured time zone is used. The time returned from the server should be in the configured time zone.

Priority of time zones

1. User profile time zone
2. Solution time zone
3. UTC

Base Address

The base address is:

```
http(s)://<host>:<port>/<path/to/application>/api
```

where <path/to/application> is a path to the iWD Manager web application. By default it is `iwd_manager`, but can be different if deployed under another path.

All addresses in this document are based on this address. Addresses returned from the server must contain the whole path, including the `/<path/to/application>/api` prefix.

Tasks

Fetch paginated list of tasks

Method

Up to 9.0.012

GET

From 9.0.013

POST

> **Important**
> The new request has exactly the same parameters as the old one and allows you to pass the ql_expression in the request body in JSON format.

Syntax

Up to 9.0.012

```
GET /gtl/tasks/<solution_dbid>/<page>?entity_dbid=<dbid>&entity_type=department/process/capturePoint/
solution>&filter=<filter_name>o_by=qualified_attribute_name>o_direction=<ascending/
descending>&snapshot_id=snapshot_id>&release_snapshot=<snapshot_id>
```

From 9.0.013

```
POST /gtl/tasks/<solution_dbid>/<page>/search?entity_dbid=<dbid>&entity_type=department/process/capturePoint/
solution>&filter=<filter_name>o_by=qualified_attribute_name>o_direction=<ascending/
descending>&snapshot_id=snapshot_id>&release_snapshot=<snapshot_id>
```

Request Body

The Request body should contain a valid JSON data with array of <ql_items>.

```
[<ql_item>, <ql_item>, ...]
```

<ql_item> is a JSON object with the following fields:

Key	Is Mandatory	Comment
attribute	-	Database column name.
operator	+	"LIKE", "IN", "=", ">", "<", ">=", "<=", "OR", "AND", "(", ")"
value	-	String, array of strings or numbers.

Important

1. Any string or date value must be wrapped in single quotes.
2. Operators are case insensitive.
3. Date format is "YYYY-MM-DDTHH:mm:ssZ", for example: 2020-06-18T14:10:12Z

Here is a JSON example of query "Queue IN ('iWD_Rejected' , 'iWD_Canceled') and 'Business Value' in (1, 2, 3) and 'Completed Date' <=

2020-06-16" :

```
[
    {
        "operator": "IN",
        "value": [
            "'iWD_Rejected'",
            "'iWD_Canceled'"
        ],
        "attribute": "queue"
    },
    {
        "operator": "AND"
    },
    {
        "operator": "IN",
        "value": [
            1,
            2,
            3
        ],
        "attribute": "IWD_businessValue"
    },
    {
        "operator": "AND"
    },
    {
        "operator": "<=",
        "value": "'2020-06-16T23:59:59Z'",
        "attribute": "completed_at"
    }
]
```

Parameters

Parameter Name	Description	Default Value
entity_dbid	DBID of solution, department, process or capture point	The same value as solution_dbid in the path)

Parameter Name	Description	Default Value
entity_type	solution, department, process or capturePoint; if solution then entity_dbid must be the same as solution_dbid in the path	solution
filter	Filter name	-
order_by	Qualified attribute name used to sort the results (qualified attribute name is a name with core/ext/ data prefix and a dot delimiter)	core.createdDateTime
order_direction	ascending or descending	descending
Custom Filter Query		
core.{any core task attribute}	attribute value	-
ext.{any extended task attribute}	attribute value	-
data.{any custom task attribute}	attribute value	-
any_id	A value of id or captureId	-
Snapshot Management		
snapshot_id	Reuse a previously taken snapshot (Interaction Server's query result) if still valid. Create a new snapshot otherwise.	-
release_snapshot	If present, a new snapshot in Interaction Server is requested. The snapshot with ID given in this parameter is released.	-

- `snapshot_id` and `release_snapshot` are optional and mutually exclusive.
- `entity_dbid` and `entity_type` must be both present or both omitted.

Response Body

```
{
  "page": <page number>,
```

```
  "tasks_per_page": <number of tasks per page>,
  "total_tasks": <number of tasks>,
  "snapshot_id": "<snapshot ID>",
  "columns":
  [
    {
      "name": "<column name>",
      "label": "<localized column label>",
      "type": "<string/date/int/img>",
      "sortable": <true/false>,
      "sorted": <ascending/descending>,
      "category": "<core/ext/data>"
    },
    ...
  ],
  "tasks":
  [
    {
      "core":
      {
        <attribute definitions>
      },
      "ext":
      {
        <attribute definitions>
      },
      "data":
      {
        <attribute definitions>
      }
    },
    ...
  ]
}
```

Attribute Definitions

```
"<attribute name>":
{
```

```
  "value": <attribute value>,
  "display_value": <value_to_display>,
  "tooltip": "<tooltip>"
},
...
```

- Returned tasks contain only attributes selected by the filter plus task ID, which is always present, regardless of whether it is selected by the filter or not.
- `"snapshot_id"` is not the same as the Interaction Server's snapshot ID. In order to be unique across the application, `"snapshot_id"` is combined from a connection ID and Interaction Server's snapshot ID.
- If a selected attribute is not set for a task, it is omitted in the returned record.
- If the attribute type is "img", the value is a URL of the image. The URL is relative to the host base address, i.e. it contains the whole path, including the path to the application and the api prefix. For example: `"value":` **"/iwd_manager/api/gtl/icons/102/workitem"**. Currently there is only one attribute of this type - `mediaIcon`. `"tooltip"` for this attribute contains the associated media type.
- The list of tasks may be empty. Pages are numbered starting from 1.
- At most one column is expected to contain the field `"sorted"`.
- `"tooltip"` and `"display_value"` are present only if different to the value. `"display_value"` is currently used only for channel.

HTTP Status Codes

- 404 Not Found—Solution, department, process, capture point or filter does not exist. Page number is `0` (zero) or negative
- **Up to release 9.0.012**: 302 Found—Requested page is out of range. Location header contains a URL of the last valid page
  From release 9.0.013: 307 Temporary Redirect
- 400 Bad Request—One of the following:
  Wrong value of `order_direction`

  Both `release_snapshot` and `snapshot_id` are present in the request

  The snapshot identified by `snapshot_id` was taken with different query parameters (solution DBID, filter, order etc.) from those used for the current request

  Any other request error

- 403 Forbidden—The snapshot identified by `release_snapshot` does not belong to the current user session

Fetch list of task IDs from current snapshot

Method

GET

Syntax

```
GET /gtl/task_ids/<solution_dbid>?snapshot_id=<snapshot_id>
```

- `snapshot_id` is mandatory

Response Body

```
[
  "<task_id_1>",
  "<task_id_2>",
  ...
]
```

- According to the Interaction Server configuration, there may be up to 2000 task IDs for one snapshot (See: the configuration option))
- Or, when a snapshot or solution with the given ID not found:

```
HTTP 404 Not Found
```

Export selected tasks

Method

Up to 9.0.012

GET

From 9.0.013

POST

> **Important**
> The new request has exactly the same parameters as the old one and allows you to pass the ql_expression in the request body in JSON format.

Syntax

Up to 9.0.012

```
GET /gtl/tasks/{solutionDbid}/{page}
```

From 9.0.013

```
POST /gtl/tasks/{solutionDbid}/{page}/search
```

Headers

```
Accept: application/xml,*/*
```

Request Body: up to 9.0.012

```
{
   "include": [ <task_id1>, <task_id2>, ...]
```

```
 }
```

Or:

```
{
  "exclude": [ <task_id1>, <task_id2>, ...]
}
```

Request Body: from 9.0.013

```
{
    "content": {
        "include": [ <task_id1>, <task_id2>, ...]
    },
    "qlExpression": [<ql_item>, <ql_item>, ...]
}
```

Or:

```
{
    "content": {
        "exclude": [ <task_id1>, <task_id2>, ...]
    },
    "qlExpression": [<ql_item>, <ql_item>, ...]
}
```

- It is not allowed to send both `"include"` and `"exclude"` lists.
- `snapshot_id` is mandatory.
- The returned format is XML.
- It is necessary to set the **Accept** header with `application/xml` as the first value. The following "*/*" is needed to correctly return an error message in JSON.
- The value for the **qlExpression** field is the same object as request body from Fetch paginated list of tasks.

Fetch single task by task ID

Method

GET

Syntax

```
GET /gtl/task/<solution_dbid>/<task_id>
```

Response Body

```
{
  "attribute_definitions":
  [
    {
      "name": "<attribute name>",
      "label": "<localized attribute label>",
      "type": "<string/date/int/img/list/age>",
      "category": "<core/ext/data>"
    },
    ...
  ],
  "task":
    {
      "core":
      {
        "<attribute_name>":
        {
           "value": <value>,
           "display_value": <display value>,
           "tooltip": <tooltip>,
           "missing": true
        },...
      },
      "ext":
      {
        ...
      },
      "data":
      {
        ...
      }
    }
 }
```

- "data" attribute definitions are sorted ascending by the localized attribute label. "core" and "ext" attribute definitions are in the same order as they were in previous releases of iWD Manager.

- "age" type has the following format:

```
"value":
  {
    "start": "<start_date_time>",
    "end": "<optional_end_date_time>"
  }
```

- If "end" is present and not null, the UI should display the duration between "start" and "end". Otherwise, it should display the duration between "start" and the current time.
- The process, department and capturePoint attributes contain names of business structure objects found by IDs (RTID) from the task: processId, departmentId and capturePointId, respectively. If an ID is set in a task but the corresponding object is missing, the attribute value is set to a default value (RTID) and a flag—missing: true—is set for such an attribute. The UI should display the default value in a specific way; for example crossed out and with a localized tooltip. Example: The attribute is missing.
- "tooltip" and "display_value" are present only if different to the value. "display_value" is currently used only for channel.
- HTTP 404 Not Found if the task was not found.

Fetch history

Method

GET

Syntax

```
GET /gtl/task_history/<solution_dbid>/<task_id>
```

Response Body

```
{
 "task_id" : "<task ID>",
 "rows_limited_to" : <number>,
 "columns":
 [
  {
    "name": "<column_ID>",
    "label": "<localized column label>",
    "type": "<string/date>"
  }, ...
 ],
 "history":
 [
   {
     "date_time": "<date_time>",
     "actor": "<actor>",
     "event_code": "<event_code>",
     "event": "<localized event message>"
   }, ...
 ]
}
```

- `"rows_limited_to"` is added only when the configured maximum number of events has been achieved. Its value is the number of events.

Task operations

Method

POST

Syntax

```
POST /gtl/task/<solution_dbid>/<task_id>/hold
POST /gtl/task/<solution_dbid>/<task_id>/resume
POST /gtl/task/<solution_dbid>/<task_id>/cancel
```

Payload is empty as all information is in the URL.

Single task modification - fetch task modifiable data

Method

GET

Syntax

```
GET /gtl/task/modifiable/<solution_dbid>/<task_id>
```

Response Body

```
{
 "attribute_definitions":
 [
   {
     "name": "<attribute name>",
     "label": "<localized attribute label>",
     "type": "<string/date/int/list>",
     "values":
      [
       {
        "value": "<name or value>",
        "label": "<display name>"
       },
     "editable_list": <true/false>,
       ...
    ],
     "category": "core/ext/data",
   },
   ...
 ],
 "attribute_values":
   {
     "core":
     {
       "<attribute_name>":
       {
          "value": <value>
       },...
     },
     "ext":
     {
       ...
     },
```

```
     "data":
     {
       ...
     }
   }
}
```

- Only modifiable attributes are included in the content,
- `"editable_list"` applies to the `"list"` type attributes only. If `true`, it is allowed to enter a value out of the `"values"` list. Currently, the `"list"` type is treated as the `"string"` type; support for the `"list"` type is not implemented.

Single task modification

Method

POST

Syntax

```
POST /gtl/task/solution_dbid>/<task_id>/modify
POST /gtl/task/solution_dbid>/<task_id>/modify_restart
```

Request Body

```
{
  "core":
  {
   "<attribute_name>":
   {
      "value": <value>
   },...
  },
  "ext":
  {
   ...
  },
  "data":
  {
   ...
  }
}
```

- Only modified attributes are included in the content.
- Attributes of type `int`—`priority`, `businessValue`—can have either a new numeric value, or a relative change, such as: `+10`, `-5`.

Response Body

Either:

```
HTTP/1.1 204 No Content
```

Or:

```
HTTP/1.1 200 Ok
```

plus:

```
[
  <message>
]
```

A warning message may be returned when the operation is successful, but an update to Universal Contact Server (UCS) failed. For example:

```
[ {
  "severity" : "WARNING",
  "message_id" : "CANNOT_OPEN_CONNECTION_TO_UCS",
  "message" : "CANNOT_OPEN_CONNECTION_TO_UCS",
  "args" : { }
} ]
```

Get common task attributes for modification

Method

POST

Syntax

```
POST /gtl/tasks/common_data/<solution_dbid>?entity_dbid=<dbid>&entity_type=<department/
process/capturePoint/
solution>&filter=<filter_name>₀_by=<qualified_attribute_name>₀_direction=<ascending/
descending>&snapshot_id=<snapshot_id>
```

Request Body

Up to 9.0.012

Either:

```
{
        "include": [ <task_id1>, <task_id2>, ...]
}
```

Or:

```
{
        "exclude": [ <task_id1>, <task_id2>, ...]
}
```

From 9.0.013

Either:

```
{
    "content": {
        "include": [ <task_id1>, <task_id2>, ...]
    },
    "qlExpression": [<ql_item>, <ql_item>, ...]
}
```

Or:

```
{
    "content": {
        "exclude": [ <task_id1>, <task_id2>, ...]
    },
    "qlExpression": [<ql_item>, <ql_item>, ...]
}
```

Notes

- It is not allowed to send both `"include"` and `"exclude"` lists.
- `snapshot_id` is mandatory.
- The value for the **qlExpression** field is the same object as the request body from Fetch paginated list of tasks.

Response Body

```
{
  "attribute_definitions":
  [
    {
      "name": "<attribute name>",
      "label": "<localized attribute label>",
      "type": "<string/date/int/img/list>",
      "values":
       [
        {
         "value": "<name of value>",
         "label": "<display name>"
        },
        ...
     ],
      "category": "core/ext/data",
    },
    ...
  ],
  "attribute_values":
  {
    "core":
    {
        <attribute definitions>
    },
    "ext":
    {
        <attribute definitions>
    },
    "data":
    {
        <attribute definitions>
    }
  }
}
```

- `"attribute_definitions"` contains only editable attributes.
- `"attribute_values"` contains only attributes that have the same value for all the selected tasks.

Bulk operations

Summary

1. A first user issues POST with the operation specification.
2. The API responds with the URL of the job status.
3. The user can query this URL with GET to check the operation status (for example to update progress bar).
4. When the user recognizes that the operation has completed (processed=total), resources should be released with DELETE.
5. Issuing DELETE when a bulk operation is still in progress aborts it, but does not roll back already modified interactions.

Methods

- POST
- GET
- DELETE

POST

Syntax

```
POST /gtl/tasks/<solution_dbid>/<page>?entity_dbid=<dbid>&entity_type=<department/process/
capturePoint/
solution>&filter=<filter_name>o_by=<qualified_attribute_name>o_direction=<ascending/
descending>&snapshot_id=<snapshot_id>
```

Request Body: up to 9.0.012

```
{
  "action": "<hold/resume/cancel/modify/modify_restart>",
  "include": [ <task_id1>, <task_id2>, ...],
  "exclude": [ <task_id1>, <task_id2>, ...],
  "attributes":
    {
      "core":
      {
        <attribute definitions>
      },
      "ext":
```

```
      {
        <attribute definitions>
      },
      "data":
      {
        <attribute definitions>
      }
    },
}
```

Request Body: from 9.0.013

```
{
    "content": {
            "action": "<hold/resume/cancel/modify/modify_restart>",
            "include": [ <task_id1>, <task_id2>, ...],
            "exclude": [ <task_id1>, <task_id2>, ...],
            "attributes":
            {
                "core":
                {
                    <attribute definitions>
                },
                "ext":
                {
                    <attribute definitions>
                },
                "data":
                {
                    <attribute definitions>
                }
            }
    },
    "qlExpression": [<ql_item>, <ql_item>, ...]
}
```

Notes

- `include` and `exclude` cannot both be present in one request.
- If `include` is present, only the selected tasks from the query will be affected.
- If `exclude` is present, all tasks from the query except for those selected will be affected.
- If neither `include` nor `exclude` is present, then all tasks from the snapshot will be affected.
- The `attributes` section should contain new values of attributes to be modified. It will be ignored for actions other than `modify` or `modify_restart`.
- The <page> path attribute is ignored in this case. All tasks that matches the query and the include/exclude list will be affected, regardless of the page.
- `snapshot_id` is mandatory.
- The value for the **qlExpression** field is the same object as request body from Fetch paginated list of tasks.

Result

```
HTTP 202 Accepted
Location: <URL of the job status>
```

Response Body

```
{
  "location": "<URL of the job status>"
}
```

Example URL:

```
/gtl/tasks/jobs/statuses/1234
```

GET

Syntax

```
GET <job status URL>
```

Response Body

```
HTTP 202 Accepted
 {
   "processed": <number of processed tasks>,
   "total": <total number of tasks to process>,
   "wait": <number of milliseconds>
 }
```

- `"wait"` is a suggested number of milliseconds to wait before asking again for the status.

```
HTTP 200 OK
 [
   {
     "task_id": "<task_id>",
     "status": "OK",
     "message": <message definition>
   },
   {
     "task_id": "<task_id>",
     "status": "ERROR",
     "message": <message definition>
   },
   ...
 ]
```

Notes

- `<message definition>` has the same format as message. Can be skipped if there is no message.
- For status = `"OK"` there may be a warning message when the operation succeeded, but an update to UCS failed.
- Or, if there is no remembered snapshot ID for the requested query:

```
HTTP 404 Not Found
```

The last option may occur if the request was sent without first sending a request for a tasks list, or the snapshot was released in the meantime. When this happens, the client should download the tasks list again, allow the user to review the tasks selection and submit the modification request again.

DELETE

Syntax

When bulk operation results have been read, they should be removed to release resources:

```
DELETE <job status URL>
```

Response Body

```
HTTP 204 No Content
```

regardless of whether the job was found or not.

If the given job is still running, calling DELETE will stop it as soon as possible. Results can also be removed automatically after some (configurable) time after finishing.

Task Attributes

- Fetch list of all available task attributes for criteria in filter definitions
- Fetch list of all available task attributes for columns in filter definitions
- Fetch list of all available task attributes for Custom Filter Query in GTL

Fetch list of all available task attributes for criteria in filter definitions

Method

GET

Syntax

```
GET /filter/attributes/<tenant_dbid>
```

Reponse body

```
[
  {
    "name" : "<attribute name>",
    "column_label": "<localized column label>",
    "filter_label" : "<localized attribute label used in filter conditions>",
    "type" : "<string/date/int/img/list>",
    "values" :
      [
        {
          "value": "<name or value>",
          "label": "<display name>"
        }
      ],
    "category" : "core/ext/data",
    "editable_list" : true/false,
    "column_name": "<column name used for mapping>"
    "filterable" : true/false
  }
]
```

Notes

- Only attributes with **filterable** = `true` will be returned in response.
- Only attributes with **type** = `list` will have properties **editable_list** and **values**.

Fetch list of all available task attributes for columns in filter definitions

Method

GET

Syntax

```
GET /filter/columns/<tenant_dbid>
```

Response body

```
[
  {
    "name" : "<attribute name>",
    "column_label": "<localized column label>",
    "filter_label" : "<localized attribute label used in filter conditions>",
    "type" : "<string/date/int/img/list>",
    "values" :
      [
        {
          "value": "<name or value>",
          "label": "<display name>"
        }
      ],
    "category" : "core/ext/data",
    "editable_list" : true/false,
    "column_name": "<column name used for mapping>"
    "filterable" : true/false
  }
]
```

Notes

- Only attributes with **type** = `list` will have properties **editable_list** and **values**.

Fetch list of all available task attributes for Customer Filter Query in GTL

Method

GET

Syntax

```
GET /gtl/attributes/<solution_dbid>
```

Response Body

```
[
  {
    "name": "<attribute name>",
    "column_label": "<localized column label>",
    "filter_label": "<localized attribute label used in filter conditions>",
    "type": "<string/date/int/img/list>",
    "values":
      [
        {
          "value": "<name or value>",
          "label": "<display name>"
        },
        ...
      ],
    "category": "core/ext/data",
    "filterable": true/false,
    "editable_list": true/false
    "column_name": "<column name used for mapping>"
  },
  ...
]
```

Notes

- The data format is the same for all the three types of requests, but the lists are different.
- Lists of attributes for filter definitions will be selected for a given tenant. Available values for "list" attributes are also tenant-specific.

- The list of attributes for Custom Filter Query in GTL is specific for a selected solution. Available values are also solution-specific.
- All the lists of attributes should be sorted by `filter_label` or `column_label`.
- All the lists of available values should be sorted by label.
- `filter_label` should be used to display the attribute in filter criteria and in customer filter query on GTL. column_label should be used to define a column in a filter. In most cases both labels are the same, but some attributes (of type `date`) should be displayed with different labels as a column title and in filter criteria. For example, `"Activated D/T"` and `"Activated Date"`, respectively.
- `values` will be used only for the type `"list"`. If `true`, a user should be allowed to type in a value that is not on the list.
- `editable_list` will be used only for the type `"list"`.
- Attribute names are those used by iWD Manager internally (for example, `createdDateTime` and not `received_at`). They may differ from those used by Interaction Server. The databse attribute names, defined in **Interaction Custom Properties/Values/**<*property*>/**annex/translation/translate-to**, should not be exposed by the API and will be used only internally to execute queries.
- `filterable` indicates if an attribute can be used in filter criteria or in Customer Filter Query. All attributes received in response to queries **/gtl/attributes** and **/filter/attributes** have filterable set to `true`. Some attributes received in response to the query **/filter/columns** have `filterable` set to `false`.
- `filter_label`, `column_label` and `column_name` can be used in Custom Filter Query in GTL: they all provide equal functional capabilities.

Filter Operations

Fetch a list of filters available for GTL

Method

GET

Syntax

```
GET /gtl/filters/<tenant_dbid>
```

Response Body

```
[
  "<filter name>",
  ...
]
```

Notes

The list contains names of filters that are owned by the current user or are public (the list may be empty). Private filters owned by other users are not displayed. This is the default behavior for Global Task List.

Fetch a list of filters accessible by the current user

Method

GET

Syntax

```
GET /filters/<tenant_dbid>?invalidateCache=<true/false>
```

Parameters

Parameter Name	Description	Default Value
invalidateCache	Optional. Can be either `true` or `false`. With value `true`, the iWD Manager backend does not use the inner cache and must reload filters from Configuration Server. Value `false` means that iWD Manager can use the inner cache for better performance. If the parameter is not present, the inner cache is used (same as for invalidateCache=false).	false

Response Body

```
[
  "<filter name>",
  ...
]
```

Notes

All filters accessible by the current user are returned. This is used by the **Filters** page to allow editing or deleting of other users' filters by users with appropriate permissions.

Fetch filter

Method

GET

Syntax

```
GET /filters/<tenant_dbid>/<filter_name>
```

Response Body

```
{
  "name": "<filter name>",
  "owner": "<owner user name>",
  "public": <true/false>,
  "columns": [
    <column definition>,
    ...
  ],
  "criteria": [
    <criterion definition>,
    ...
  ]
}
```

Where:

- `<column definition>`:

{

```
  "category": "core/ext/data",
  "name": "<attribute name>"
}
```

And:

- <criterion definition>:

```
{
  "template_expression": "<template_expression>",
  "template_key": "<template label key for i18n>",
  "elements":
  [
    {
      "type": "INPUT_INTEGER/INPUT_DATE/INPUT_TEXT/INPUT_ATTRIBUTE_NAME/INPUT_ATTRIBUTE_VALUE/PLAIN_TEXT",
      "name": "<element name>",
      "label": "<label key for i18n>",
      "value": "<current value>",
      "editable_list": <true/false>,
      "values":
      [
        {
          "value": "<available value>",
          "label": "<value label key>"
        }
        ...
      ]
    },
    ...
  ]
}
```

Notes

- values define a list of items that can be picked from a drop-down menu.
- A list of values is empty for elements of INPUT_DATE type. It can be non-empty for INPUT_TEXT and INPUT_INTEGER types. It is typically empty for INPUT_ATTRIBUTE_NAME and INPUT_ATTRIBUTE_VALUE - a list of available attributes and their values should be taken by a separate request (see:

Attributes). If the list of values exists and is not empty, it has precedence over the list of all attributes.

- Internal element types IS_IS_NOT, OPERATOR, TIME_UNIT, WAS_WAS_NOT, MORE_LESS, QUEUE are being converted to and from INPUT_TEXT with appropriate lists of values.
- editable_list indicates whether the element value input component is or is not editable, i.e. if a user is able to put a value out of the given list. For INPUT_ATTRIBUTE_VALUE the value of this property has precedence over the same property on the attribute definition.
- Labels for elements have a form of "CRITERION_LABEL_<CAPITALIZED_NAME>", for example CRITERION_LABEL_IS_IS_NOT.
- Actual labels for the criterion, elements and available values should be loaded from resources (localized). If there are not found, the element name should be used instead.
- template_key is a key to find a localized criterion template in resources. The localized template can contain parameters as element names in braces (for example {attributeName}). These parameters are replaced by either a proper input GUI component in edition mode or element’s localized labels for displaying the criterion template. In case the template_key is not found in resources the criterion template is built of elements directly – one by one (elements of PLAIN_TEXT type – not used usually in localized templates - fill the text space then). All dynamic elements from template_expression must be included in the localized string to allow correct creation of a query string.
- INPUT_ATTRIBUTE_VALUE is an input element which data type should be dynamically determined basing on the type of attribute, chosen with INPUT_ATTRIBUTE_NAME, unless the element’s values list exists and is not empty, because it has the precedence over the attribute values.
- template_expression is a legacy template expression, which is used here as the template criterion identifier only.
- A list of available columns is available under a separate address.

Fetch filter criteria templates

Method

GET

Syntax

```
GET /filter/criteria/<tenant_dbid>
```

Response Body

```
[
  {
     "template_expression": "<filter expression template>",
     "template_key": "<template expression identifier>",
     "elements":
       [
         {
           "name": "<element name>",
           "value": "<element value>",
           "type": "<element type>",
           "label": "<element label>",
           "supported_types": ["<string/date/integer>", ...]
         },
         ...
       ]
  }
  ...
]
```

Notes

- A format of the criterion definition is the same as for criteria in filter definitions.
- `value` of an element is a default value in this case.

Create filter

Method

POST

Syntax

```
POST /filters/<tenant_dbid>
```

Response Body

```
{
  "name": "<filter name>",
  "public": <true/false>,
  "columns": [
    <column definition>,
    ...
  ],
  "criteria": [
    <short criterion definition>,
    ...
  ]
}
```

Where short criterion definition is defined as the following:

```
{
  "template_expression": "<template_expression>",
  "elements":
  [
    {
      "name": "<element name>",
      "value": "<current value>"
    },
    ...
  ]
}
```

Notes

- A filter is required to contain at least one column. If there is no column, an error status `HTTP 400 Bad Request` will be returned.

- The list of criteria may be empty or missing.
- If an element name is `INPUT_ATTRIBUTE_NAME`, its value must be a qualified attribute name. If the attribute is not recognized in the requested category, the data category is assumed, regardless of its prefix.

Update filter

Method

PUT

Syntax

```
PUT /filters/<tenant_dbid>/<filter_name>
```

Notes

The payload contains a modified filter definition. The format is the same as for Create filter.

Delete filter

Method

DELETE

Syntax

```
DELETE /filters/<tenant_dbid>/<filter_name>
```

Response Body

Whether the filter is found or not:

```
HTTP 204 No Content
```

Media Icons

Fetch list of media types

Method

GET

Syntax

```
GET /media_types/<tenant_dbid>
```

Response Body

```
[
    "alert",
    "any",
    "appsharing",
    "auxwork",
    "busevent",
    "callback",
    "chat",
    "cobrowsing",
    "email",
    "fax",
    "imchat",
    "mms",
    "mmssession",
    "outboundpreview",
    "smail",
    "sms",
    "smssession",
    "trainingitem",
    "video",
    "vmail",
    "voice",
    "voip",
    "webform",
    "whiteboard",
    "workitem"
]
```

Fetch list of media icons

Method

GET

Syntax

```
GET /icons/<tenant_dbid>?invalidateCache=<true/false>
```

Parameters

Parameter Name	Description	Default Value
invalidateCache	Optional. Can be either `true` or `false`. With value `true`, the iWD Manager backend does not use the inner cache and must reload icons from Configuration Server. Value `false` means that iWD Manager can use the inner cache for better performance. If the parameter is not present, the inner cache is used (same as for `invalidateCache=false`).	false

Response Body

```
[
    {
        "media_type": "<media_type>",
        "media_icon":
        {
            "content_type": "<content_type>",
            "name": "<icon_file_name>",
            "encoding": "base64",
            "data": "<icon_data_encoded_with_encoding_method>"
        }
    },
    ...
]
```

Notes

- There may be icons with `media_type` that is missing in the media types list. They should be returned as well.
- Currently only `"base64"` is a valid value of `"encoding"`.

Fetch media icon to display in GTL

Method

GET

Syntax

```
GET /gtl/icons/<tenant_dbid>/<media_type>
```

Response Body

```
<raw image data>
```

Notes

- **Content-Type** is `image/png`, `image/bmp`, `image/jpg` or `image/gif`.
- If icon is missing or its content-type is invalid, status `HTTP 404` is returned.

Upload media icon

Method

PUT

Syntax

```
PUT /icons/<tenant_dbid>/<media_type>
```

Request Body

```
{
    "content_type": "<content_type>",
    "name": "<icon_file_name>",
    "encoding": "base64",
    "data": "<icon_data_encoded_with_encoding_method>"
}
```

Notes

- Animated GIF images may be uploaded, but they are not guaranteed to work correctly. Especially if resizing is necessary on the server, the image is assumed to be static and saved as PNG.
- `"name"` is optional. It must be up to 255 characters long. In case of resizing, a new name is being created.
- Depending on a back-end database management system a missing name may be returned later as an empty string or omitted.
- `"encoding"` is optional—if it is missing, `"base64"` is assumed, as that is currently the only option.
- `"content_type"` must be one of the supported image content types.
- `"media_type"` must be one of media types defined on the configuration server and also it must be up to 255 characters long.

Delete media icon

Method

DELETE

Syntax

```
DELETE /icons/<tenant_dbid>/<media_type>
```

Notes

There is no separate POST operation for media icons, because the list of media types is fixed in IWD and new one cannot be created by GTL. PUT is used for both create and update operations.

Media icons export

Method

GET

Syntax

```
GET /icons/xml/<tenant_dbid>
```

Output

XML

Media icons import

Method

POST

Syntax

```
POST /icons/xml/<tenant_dbid>
```

Response Body

The format of imported/exported icons is compatible with iWD 8.1+ configuration, including icons encoded with Base64. Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<EvoConfiguration>
 <iWDVersion>(version)</iWDVersion>
 <MediaIcons>
   <Icon name="webform.png" contentType="image/png" mediaType="webform"
data="iVBORw0KGgoAAAANSUhEUgAAABAAAAAQCAYAAAAf8/9hAAAABGdBTUEAAK/INwWK6QAAABl0RVh0&..." />
   ...
 </MediaIcons>
</EvoConfiguration>
```

Tenants

Fetch Business Structure

Method

GET

Syntax

```
GET /gtl/business_structure
```

Response Body

Returns all available tenants, including "Environment", with business structure.

```
[
  {
    "dbid": 1,
    "name": "Environment",
    "type": "tenant",
    "rules_authoring_tool_url": "http://server.example.com:8380/genesys-rules-authoring",
    "iwd_web_url": ""http://server.example.com:8080/iwd_web",
    "tenants":
    [
      {
        "dbid": 102,
        "name": "SubTenant",
        "type": "tenant",
        "tenants": [],
        "solutions": []
      }
    ],
    "solutions":
    [
       {
         "dbid": 1001,
         "name": "Solution1",
         "type": "solution",
         "timezone": "<timezone name>",
         "first_day_of_week": "<MONDAY|SUNDAY>",
         "departments":
         [
           {
             "dbid": 102,
             "name": "Department 1",
             "type": "department",
             "processes":
             [
               {
```

```
                "dbid": 1111,
                "name": "Process 1",
                "type": "process"
              }
            ]
          }
        ],
        "capture_points":
        [
          {
            "dbid": 222,
            "name": "Capture Point 1",
            "type": "capturePoint"
          }
        ]
      },
      {
        "dbid": 104,
        "name": "Solution 2",
        "type": "solution",
        "departments": [],
        "capture_points": []
      }
    ],
  }
]
```


Notes

- `rules_authoring_tool_url` is taken from Configuration Server; location: **Business Structure/**<*path to tenant*>**/annex/iWD/rules-authoring-tool-url**
- `iwd_web_url` is taken from Configuration Server; location: **Business Structure/**<*path to tenant*>**/annex/iWD/iwd-web-url**. (Configuration of this option is to be added to iWD GAX Plugin in the future).

Fetch tenants tree

Method

GET

Syntax

```
GET /tenants
```

Response Body

Response—thetree of all available tenants without business structure.

```
[
  {
    "dbid": 1,
    "name": "Environment",
    "type": "tenant",
    "rules_authoring_tool_url": "http://server.example.com:8380/genesys-rules-authoring",
    "iwd_web_url": "http://server.example.com:8080/iwd_web",
    "tenants":
    [
      {
        "dbid": 102,
        "name": "SubTenant",
        "type": "tenant",
        "tenants": [],
        "solutions": []
      }
    ],
    "solutions": []
  }
]
```

Notes

- `"solutions"` may be present for convenience; they are always empty.

User Settings

Fetch user settings

Method

GET

Syntax

```
GET /user_settings
```

Response Body

From 9.0.016

logged_with_saml and **slo_enabled** are removed.

```
{
  "user_dbid": <dbid>,                // for example 1234
  "user": "<short_user_name>",        // for example "django"
  "user_first_name": "<first_name>",  // for example "Django"
  "user_last_name": "<last_name>",    // for example "Freeman"
  "timezone": {
    "value": "<current_TZ_value>",    // for example "America/Chicago"
    "values": [
      {
        "value": "<TZ_value>",        // for example "America/Chicago"
        "label": "<TZ_label>"         // for example "America/Chicago (GMT-6)(+DST)"
      },
      ...
    ]
  },
  "language": "<language_ID_string>",               // for example "fr-CA" (IETF notation
recommended)
  "date_time_format": "<date_time_format_string>",   // for example "yyyy/MM/dd HH:mm:ss Z"
  "date_format": "<date_format_string>",             // for example "yyyy/MM/dd"
  "first_day_of_week": "<SUNDAY|MONDAY>",            // default: MONDAY
  "last_logged_in": "<date_time_of_last_logging_in>",// for example
"2016-05-03T12:30:55+02:00"
  "default_filter": {
    "selected_filter": <filter_name>,               // for example Assigned
    "available_filters": [
      "<filter_name_1>",
      "<filter_name_2>",                            // for example Assigned
      ...
    ]
  },
  "user_privileges": [
```

```
    {
      "tenant_dbid": <tenant_dbid>,
      "privileges": [
        "<privilege_1>",  // for example "GLOBAL_TASK_LIST_MODIFY"
        "<privilege_2>",
        ...
      ]
    },
    ...
  ],
  "system_timezone": "<timezone>",                  // time zone from the server; for
example "America/Chicago"
  "external_authentication": <true|false>,          // shows whether external authentication
is enabled on Config Server.
}
```

Up to 9.0.015

```
{
  "user_dbid": <dbid>,                // for example 1234
  "user": "<short_user_name>",        // for example "django"
  "user_first_name": "<first_name>",  // for example "Django"
  "user_last_name": "<last_name>",    // for example "Freeman"
  "timezone": {
    "value": "<current_TZ_value>",    // for example "America/Chicago"
    "values": [
      {
        "value": "<TZ_value>",        // for example "America/Chicago"
        "label": "<TZ_label>"         // for example "America/Chicago (GMT-6)(+DST)"
      },
      ...
    ]
  },
  "language": "<language_ID_string>",               // for example "fr-CA" (IETF notation
recommended)
  "date_time_format": "<date_time_format_string>",   // for example "yyyy/MM/dd HH:mm:ss Z"
  "date_format": "<date_format_string>",             // for example "yyyy/MM/dd"
  "first_day_of_week": "<SUNDAY|MONDAY>",            // default: MONDAY
  "last_logged_in": "<date_time_of_last_logging_in>",// for example
"2016-05-03T12:30:55+02:00"
  "default_filter": {
    "selected_filter": <filter_name>,               // for example Assigned
    "available_filters": [
      "<filter_name_1>",
      "<filter_name_2>",                            // for example Assigned
      ...
    ]
  },
  "user_privileges": [
    {
      "tenant_dbid": <tenant_dbid>,
      "privileges": [
        "<privilege_1>",  // for example "GLOBAL_TASK_LIST_MODIFY"
        "<privilege_2>",
        ...
      ]
    },
    ...
  ],
  "system_timezone": "<timezone>",                  // time zone from the server; for
example "America/Chicago"
  "external_authentication": <true|false>,          // shows whether external authentication
```

```
is enabled on Config Server.
  "logged_with_saml": false,                    // In 9.0.015: deprecated. Always
returns false.
  "slo_enabled": false                          // In 9.0.015: deprecated. Always
returns false.
}
```

Notes

- `system_timezone` is taken from the server running the iWD Manager application. May be empty if the time zone is not compatible with Joda Time.
- `external_authentication` shows using external authentication by Management Framework.

Update user settings

Method

PUT

Syntax

```
PUT /user_settings
```

Request Body

```
{
    "timezone": {
        "value": "<current_TZ_value>"    // for example "America/Chicago"
    },
    "language": "<language_ID_string>",                // for example "fr-CA" (IETF notation
recommended)
    "date_time_format": "<date_time_format_string>",   // for example "yyyy/MM/dd HH:mm:ss Z"
    "date_format": "<date_format_string>",             // for example "yyyy/MM/dd"
    "first_day_of_week": "<SUNDAY|MONDAY>"            // default: MONDAY
}
```

Notes

- It is also possible to use the full data format as returned by the GET method. In this case, all the unnecessary properties are ignored.
- `current_tenant_dbid` is considered a read-only property; it can be changed by a separate request.

Update current tenant

Method

PUT

Syntax

```
PUT /user_settings/current_tenant
```

Request Body

- **Content-Type**: `application/json`
- **Payload**: single decimal number - tenant DBID. For example:

```
101
```

Update current entity

Method

PUT

Syntax

```
PUT /user_settings/current_entity
```

- **Content-Type**: `application/json`—for example:

```
{
 "current_tenant_dbid": "<current_tenant_dbid>",
 "current_solution_dbid": "<current_solution_dbid>",
 "current_entity_dbid": "<current_entity_dbid>"
 "current_entity_type": "<current_entity_type>"      // for example solution, department,
process
 }
```

Notes

- current_solution_dbid, current_entity_dbid, current_entity_type are being reset to null upon a tenant change

Change password on demand

Method

PUT

Syntax

```
PUT /user_settings/password
```

Request Body

```
{
 "old_password": "<plain_text_or_base64_encoded_old_password>",
 "new_password": "<plain_text_or_base64_encoded_new_password>",
 "password_encoded": "<true/false>"
}
```

Notes

- "password_encoded" is optional—if missing, plain text old and new password are assumed.
- When it is set to true, both passwords must be encoded with Base64.

Security

Login via POST form

Method

POST

Syntax

```
POST /login
```

Request Body

Body is url-encoded and contains the following data:

```
username=<username>&password=<password>[&passwordEncoded]
```

Login via GET parameters

Method

GET

Syntax

```
GET /login?username=<username>&password=<password>[&passwordEncoded]
```

or

```
GET /ui/login?username=<username>&password=<password>[&passwordEncoded]
```

or

```
GET /ui/
login.jsf?username=<username>&password=<password>[&passwordEncoded][≈plication=<application_name>]
```

Login request with GET parameters must be syntactically backward-compatible with previous versions of iWD Manager. In both cases (POST and GET), passwordEncoded is optional; if present, the password must be encoded with Base64. The application parameter is optional and is ignored; it is allowed for compatibility only.

Response

```
HTTP 302 Found
```

The Location header contains either to the GUI starting page (on success) or to the login page with an error message (on failure).

Logout

Method

- POST
- GET (no longer supported from release 9.0.014)

Syntax

From 9.0.014

```
POST /logout.jsf
```

Up to 9.0.013

```
POST|GET /logout.jsf
```

Response

From 9.0.014

```
HTTP 302 (only returned for logged in user).
HTTP 401 (otherwise).
```

Up to 9.0.013

```
HTTP 302 Found
```

Notes

The Location header redirects to the login page.

Automatic logout

The iWD Manager server session expires after a time period set up in the configuration. Every http request sent to API resets the session expiration timer.

Method

- GET
- POST

Syntax

There is a special request just to keep the session alive:

```
GET /api/session/idle
```

Response

Result:

```
HTTP 204 No content
```

There is also another API request to automatic logout:

```
POST /api/session/autologout
```

Result:

```
HTTP 302 Found
```

Notes

- The Location header redirects to the login page (with reason=sessionExpired).
- The difference between automatic and the normal logout requests is in a logout reason responding back to the front-end. The automatic logout request redirects to the login screen with the reason "session expired" and the normal logout - with "logged out successfully".
- In every response a special cookie "SESSIONLIFETIME=<timestamp>_<lifetime>" is applied where the front-end is able to find necessary information to use the automatic logout and keeping alive mechanism.

Content Security Policy

Header Value

All server responses contains a Content-Security-Policy header:

```
default-src 'self'; img-src 'self' data:; frame-ancestor 'self'
```

Using CSRF/XSRF tokens

Overview

Every POST, PUT and DELETE request in iWD's REST API, as well as GET requests described in Login via GET parameters, should include a CSRF/XSRF token.

> **Important**
> All such requests sent without a CSRF/XSRF token result in a HTTP code 403.

Procedure

1. Send any GET request to iWD Manager (such as **GET/iwd_manager**).
2. Read the value of a token from the XSRF-TOKEN cookie in the received response.
3. Use the token value in subsequent REST API requests by setting it up in either the **X-XSRF-TOKEN** header or the **_csrf** query parameter.

History Node

- Download event history for single Interaction ID
- Download range of events for given Solution ID.
- Delete range of events for given Solution Id.
- Return maximum available event sequence number internal to History Node.

Download event history for single Interaction ID

This query is used by iWD Manager for fetching task history. It contains a list of event records sorted from newest to oldest.

Method

GET

Syntax

```
GET /gtl/events/{InteractionId}
```

Response Body


```
{
 "actor": "urs/Prioritization",
 "eventKey": "QUEUE",
 "eventSeqNum": 1052197,
 "eventTime": 1490878831000,
 "id": 1910,
 "interactionId": "028GB2TRUM0NV04G",
 "param1": null,
 "param2": null,
 "param3": null
}
```

Download range of events for given Solution ID

Downloads a range of events for a given Solution Id. Parameters (from) and (to) are event sequence numbers internal to History Node and are treated inclusively. Contains list of event records.

Method

GET

Syntax

```
GET /datamart/events/{solutionId}/{from}/{to}
```

Response Body

The record has the following structure with the "'properties"' field containing serialized Interaction Server reporting events:

```
{
 "eventKey": "NEW",
 "eventSeqNum": 1048579,
 "id": 1,
 "interactionId": "028GB2TRUM0NV000",
 "properties": "FgAAAKAAA .... DEAAACQaQAAAAAA",
 "queueKey": "NEW",
 "solutionId": "SLT1"
},
```

Delete range of events for given Solution Id

Method

DELETE

Syntax

```
DELETE /datamart/events/{solutionId}/{from}/{to}
```

Description

Parameters from and to are event sequence numbers internal to History Node and are treated inclusively. Only the Data Mart part of the History Node database is affected.

Return maximum available event sequence number internal to History Node

Method

GET

Syntax

```
GET /datamart/events/id
```