



# Universal Shop Floor Connectivity

Version 5.11

---

## *Product Description*



Document: Product Description -  
Universal Shop Floor  
Connectivity.docx



Release date: 2020-10-20



Document version: 1



Author: STernes

---

## Product Description

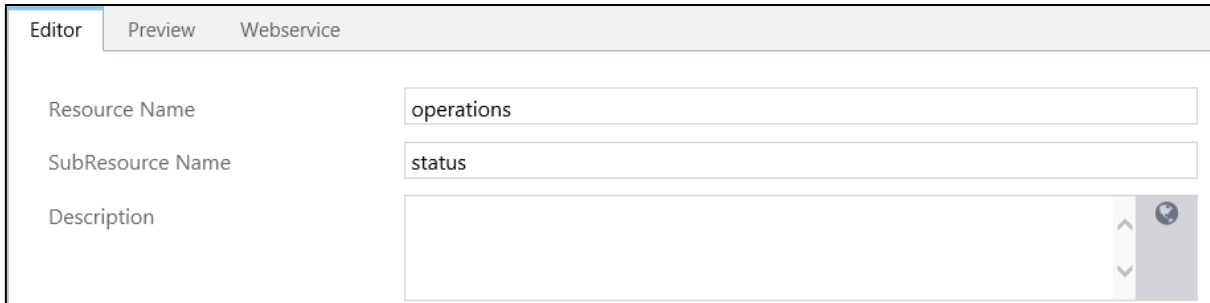
With the introduction of a machine data collection/plant data collection (MDC/PDC) system, production enterprises usually aim at making the data collected from the shop floor available to all applications within the production site. Universal Shop Floor Connectivity enables third-party systems to make the data collected in FORCAM FORCE™ available by a web service via a freely configurable REST interface. For this purpose, the appropriate data are retrieved from the database using SQL and represented as a *network resource* which can be accessed using the *Hypertext Transfer Protocol* (HTTP).

### Resource Name and URL

When configuring a resource, a {Resource} name is specified for it. Each resource can be unambiguously identified, located and addressed by a *Uniform Resource Locator* (URL) as follows:

`http://{host}:{port}/ffwebservices/customized/v1/{Resource}/`

The placeholders {host} and {port} need to be specified as appropriate for the actual system installed. The {Resource} placeholder represents the name specified when configuring the resource. The resources configured in this way can be extended with subresources that have to be assigned a unique {Subresource} name in turn (see figure 1).



The screenshot shows a configuration window with three tabs: 'Editor' (selected), 'Preview', and 'Webservice'. Below the tabs are three input fields: 'Resource Name' with the value 'operations', 'SubResource Name' with the value 'status', and 'Description' which is empty. A small globe icon is visible in the bottom right corner of the description field.

**Figure 1: Configuration of resource and subresource names**

Subresources can then be addressed as follows:

`http://{host}:{port}/ffwebservices/customized/v1/{Resource}/{Subresource}/`

## Server Response

All web services configured return a JSON-based representation of a specific resource in *Hypermedia Application Language* (HAL). The information returned is referred to as a *Collection*, i.e. a list of objects returned. The individual data fields of the return objects are referred to as items. The name, content and data format of the items returned can be freely defined in the resource configuration. The user can also preset the names of the return objects to be used in the server response. In default of such a presetting, it is generated automatically by concatenating {Resource} and {Subresource}, with the subresource – if any – starting with a capital letter.

Collection	operationsStatus	
Items	Column Identifier (SQL)	Response Field (API)
	OPERATION_ID	operationId
	OPERATION_STATUS_ID	statusId
	UUID	phaseUuid
	CODE	phaseCode
	SCHEDULED_START_DATE	scheduledStartDate
	TARGET_START_DATE	plannedStartDate
	OPERATION_NUMBER	operationNumber
	OPERATION_SPLIT	operationSplit

**Figure 2: Configuration of the response**

A date type and data format must be configured for each item (see figure 3). You can use the **formatted** argument to change the data format used by default in the BRIDGE API for the specific data type to the data format configured.

### Edit Columns

Column Identifier (SQL)	OPERATION_ID
Response Field (API)	operationId
Data Type	<div style="border: 1px solid #ccc; padding: 2px;">             Operation           </div>
Data Format	<div style="border: 1px solid #ccc; padding: 2px;">             Workplace              Material              Operation              Date/Time           </div>

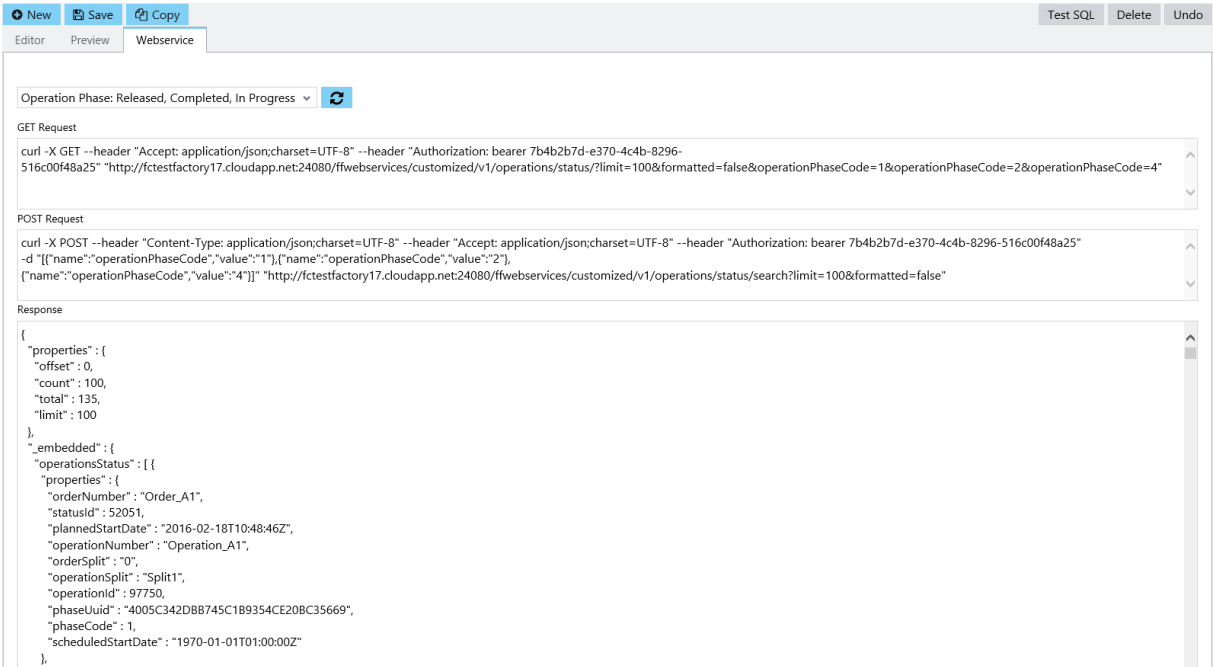
**Figure 3: Configuration of the individual items of a response**

If the return field data type is workplace, material, operation, operating state, order or status class, links to the corresponding resources of the BRIDGE API are automatically generated in the web service response. These can be embedded by the calling client by specifying the linked resources with the **embed=** argument. You can select one of the formats configured in the data format editor or the default format configured there for the above listed data types.

## Client Request

For resource requests by clients, the HTTP methods GET and POST are supported. Generally the following arguments are available, which usually exist for a collection request: **limit**, **offset** and **embed**. The additional **formatted** argument allows overriding the default formatting of the data types according to the configuration. Other arguments can be configured as additional filters.

Each web service configured is documented in a technical layout in a separate tab. You can also test there which client arguments produce which server response. In most cases, you can conveniently select the arguments from the drop-down menu of the filter. When all filter values are set, the client request to access the web service is output including all arguments for the GET and POST methods. This is to demonstrate to applications programmers how to call the web service from the client. The contents of the fields can be copied and pasted directly into a command shell to call the HTTP CURL client. The complete response of the web service for the respective arguments is output below.



The screenshot shows a web service configuration interface with tabs for 'Editor', 'Preview', and 'Webservice'. The 'Webservice' tab is active, displaying a dropdown menu for 'Operation Phase' with options: Released, Completed, In Progress. Below this, there are sections for 'GET Request', 'POST Request', and 'Response'.

**GET Request**

```
curl -X GET --header "Accept: application/json;charset=UTF-8" --header "Authorization: bearer 7b4b2b7d-e370-4c4b-8296-516c00f48a25" "http://fctestfactory17.cloudapp.net:24080/ffwebservices/customized/v1/operations/status/?limit=100&formatted=false&operationPhaseCode=1&operationPhaseCode=2&operationPhaseCode=4"
```

**POST Request**

```
curl -X POST --header "Content-Type: application/json;charset=UTF-8" --header "Accept: application/json;charset=UTF-8" --header "Authorization: bearer 7b4b2b7d-e370-4c4b-8296-516c00f48a25" -d [{"name":"operationPhaseCode","value":"1"},{"name":"operationPhaseCode","value":"2"}, {"name":"operationPhaseCode","value":"4"}] "http://fctestfactory17.cloudapp.net:24080/ffwebservices/customized/v1/operations/status/search?limit=100&formatted=false"
```

**Response**

```
{
  "properties": {
    "offset": 0,
    "count": 100,
    "total": 135,
    "limit": 100
  },
  "_embedded": {
    "operationsStatus": [ {
      "properties": {
        "orderNumber": "Order_A1",
        "statusId": 52051,
        "plannedStartDate": "2016-02-18T10:48:46Z",
        "operationNumber": "Operation_A1",
        "orderSplit": "0",
        "operationSplit": "Split1",
        "operationId": 97750,
        "phaseUid": "4005C342DBB745C1B9354CE20BC35669",
        "phaseCode": 1,
        "scheduledStartDate": "1970-01-01T01:00:00Z"
      }
    }
  ]
}
```

**Figure 4: Documentation and test of the web services configured**

---

## Scope of Functions

- Representational State Transfer (REST) based on HTTP
- Hypermedia representation format: Hypermedia Application Language & JSON (hal+json)
- Authorization with OAuth 2.0 (Open Authentication)
- Resources and subresources can be freely configured
- Return objects of the collections can be freely configured using SQL statements
- Items of the return objects can be freely compacted within the scope of SQL
- Name and data format of the individual items can be freely configured
- Arguments for the request can be freely configured
- Automatic generation of links to the resources of the official Bridge API
- Embedding of resources of the Bridge API into the response
- Requests of configured resources using the GET or POST methods
- Documentation and test of the web services configured including complete request and response