Web Services - Code Examples

Simple example of a wsCustomerInfo Web Service that brings a line of data for a given instance

The following Web Service gets an input LUI for the CUSTOMER LU and returns data from the CUSTOMER table in the CUSTOMER LU. Output data is returned in DB.Rows structure. It can also be returned as an Object which is then converted by Fabric into DB.Rows structure.

```
String sql = "SELECT CUSTOMER_ID, SSN, FIRST_NAME, LAST_NAME FROM CUSTOMER";


Db.Rows rows = ludb("Customer", i_id).fetch(sql);

reportUserMessage("WS executed Succesfully for Customer ID :" + i_id);

return rows;
```

Output:


```
[
  {
    "CUSTOMER_ID": "1",
    "SSN": "5153527856",
    "FIRST_NAME": "Tali",
    "LAST_NAME": "Griffin"
  }
]
```

Example of a wsCustomerInfo2 Web Service that brings a Db.Rows structure as an output for a given instance

The following Web Service gets an input LUI for the CUSTOMER LU and returns several rows of data by running a Join query on several tables in the CUSTOMER LU. Output data is returned in DB.Rows structure. It can also be returned as an Object which is then converted by Fabric into DB.Rows structure.

```
String sql = "select cust.CUSTOMER_ID,cust.SSN,cust.FIRST_NAME||'
'||cust.LAST_NAME CUSTOMER_NAME,
cont.CONTRACT_ID,cont.CONTRACT_DESCRIPTION,sub.SUBSCRIBER_ID,sub.MSISDN,sub.I
MSI,sub.SIM,sub.SUBSCRIBER_TYPE " +
               "from CUSTOMER cust, CONTRACT cont, SUBSCRIBER sub where
cont.CONTRACT_ID=sub.SUBSCRIBER_ID";

Db.Rows rows = ludb("Customer", i_id).fetch(sql);

reportUserMessage("WS executed Succesfully for Customer ID :" + i_id);

return rows;
```

Output:


```
[
  {
    "CUSTOMER_ID": "1",
    "SSN": "5153527856",
    "CUSTOMER_NAME": "Tali Griffin",
    "CONTRACT_ID": "1.0",
    "CONTRACT_DESCRIPTION": "5G tether",
    "SUBSCRIBER_ID": "1",
    "MSISDN": "9614867860",
    "IMSI": "531015353732639",
    "SIM": "2988735759833578",
    "SUBSCRIBER_TYPE": "1"
  },
  {
    "CUSTOMER_ID": "1",
    "SSN": "5153527856",
    "CUSTOMER_NAME": "Tali Griffin",
    "CONTRACT_ID": "2.0",
    "CONTRACT_DESCRIPTION": "10G 3G",
    "SUBSCRIBER_ID": "2",
    "MSISDN": "7997099409",
    "IMSI": "457470703125000",
    "SIM": "3751389217697811",
    "SUBSCRIBER_TYPE": "3"
  },
  {
    "CUSTOMER_ID": "1",
    "SSN": "5153527856",
    "CUSTOMER_NAME": "Tali Griffin",
    "CONTRACT_ID": "3.0",
    "CONTRACT_DESCRIPTION": "Roaming special",
```

```
    "SUBSCRIBER_ID": "3",
    "MSISDN": "3924663547",
    "IMSI": "759668646918403",
    "SIM": "2395410334354832",
    "SUBSCRIBER_TYPE": "1"
  },
  {
    "CUSTOMER_ID": "1",
    "SSN": "5153527856",
    "CUSTOMER_NAME": "Tali Griffin",
    "CONTRACT_ID": "4.0",
    "CONTRACT_DESCRIPTION": "450 min",
    "SUBSCRIBER_ID": "4",
    "MSISDN": "7042855196",
    "IMSI": "345797729492187",
    "SIM": "9009227816906040",
    "SUBSCRIBER_TYPE": "4"
  },
  {
    "CUSTOMER_ID": "1",
    "SSN": "5153527856",
    "CUSTOMER_NAME": "Tali Griffin",
    "CONTRACT_ID": "5.0",
    "CONTRACT_DESCRIPTION": "Unlimited call",
    "SUBSCRIBER_ID": "5",
    "MSISDN": "7183304985",
    "IMSI": "734794108072916",
    "SIM": "5671433642523324",
    "SUBSCRIBER_TYPE": "4"
  }
]
```

Example of versioning

Both the wsCustomerInfo and wsCustomerInfo2 Web Services in the examples share the same URL path named test/getCustomerInfo. The version property of wsCustomerInfo is set to 1 and the version property of wsCustomerInfo1 is set to 2.

- To invoke a call to wsCustomerInfo the following URL should be called: http://localhost:3213/api/v1/test/getCustomerInfo?i_id=1&token=ABC&format=json
- To invoke a call to wsCustomerInfo2 the following URL should be called: http://localhost:3213/api/v2/test/getCustomerInfo?i_id=1&token=ABC&format=json

Example of a complex Java input structure

A complex JSON format can also be sent as input to a Fabric Web Service using the POST verb. Data is automatically serialized according to the input structure defined as a part of the Web Service's markup.

For example:

Requested body

```
{
  "ID":[{"id":"78999", "company":"Telco1"},{"id":"z34","company":"Telco2"}],
  "parent_customer_id":"456",
  "company":"Telco International"
}
```

Web Service markup

```
public static String wsExample(List<Map<String,String>> ID, String
parent_customer_id, String company){
}
```

Web Service inside logic

```
    Map<String,String> m = ID.get(0);
    String id = m.get("id"); // will return 78999
    String company = m.get("company"); // will return Telco1
```

Example of a complex Customized input structure

Same as the example above, however with a customized Java classes.

For example:

Requested body

```
{
  "person": {
    "address": [
      {
        "number": 10,
        "city": "Net York",
        "street": "5th Ave."
      }
    ],
    "name": "Lion",
    "id": "1234",
    "age": 45
  }
}
```

Web Service markup

```
static class Person {
   String name;
   String id;
   int age;
   List<Address> address;
}

static class Address {
   String city;
   String street;
   int number;
}

        @webService(path = "", verb = {MethodType.GET, MethodType.POST,
MethodType.PUT, MethodType.DELETE}, version = "1", isRaw =
                                        false, produce = {Produce.XML,
Produce.JSON})
        public static Address CustomClassExample(Person person) throws
Exception {
```

Web Service inside logic

```
     return person.address.get(0);
```

Web Service response

```
{
  "city": "Net York",
```

```
  "street": "5th Ave.",
  "number": 10
}
```

Example of a complex TDM Web Service

The wsGetTaskExeStatsForEntity Web Service used by TDMGUI brings a map of all entity lists related to a given LUI that are related to the same business entity. That is, all instances related to all LUT under the same task execution that are defined as a parent or child of the given input LUI, call recursive functions to get a full hierarchy path.

```
String sqlGetEntityData = "select lu_name luName, target_entity_id targetId,
entity_id sourceId, " +
        "execution_status luStatus from TDM.task_Execution_link_entities  " +
        "where lu_name <> ? and target_entity_id = ? and entity_id = ?";

String sqlGetParent = "select parent_lu_name, target_parent_id from
TDM.task_Execution_link_entities " +
        "where lu_name= ? and target_entity_id = ? and parent_lu_name <> ''";

Map <String, Object> mainOutput = new HashMap<>();
Map <String, Object> childHierarchyDetails = new HashMap<>();
Map <String, Object> parentHierarchyDetails = new HashMap<>();

Db.Row entityDetails = null;
Boolean countChildren = false;

fabric().execute( "get TDM." + taskExecutionId);

//Get the Hierarchy starting from the given entity and below
childHierarchyDetails = fnGetChildHierarchy(luName, targetId);

String parentLuName = "";
String parentTargetId = "";

// Get the parent of the given LU, to see if there is a reason to get the
ancestors or not
Db.Row parentRec = fabric().fetch(sqlGetParent, luName, targetId).firstRow();

if (!parentRec.isEmpty()) {
        //log.info("There is a parent: " + parentRec.cell(0));
        parentLuName = "" + parentRec.cell(0);
        parentTargetId = "" + parentRec.cell(1);
}
//If the the input entity has parents get the hierarchy above it
if (parentLuName != null && !"".equals(parentLuName)) {
        //Starting for the parent as the details of the input entity is
already included in the children part
        //Sending the chilren hierarchy in order to add it to the ancestors as
child hierarchy
        parentHierarchyDetails = fnGetParentHierarchy(parentLuName,
parentTargetId, childHierarchyDetails);
} else {// Given inputs are of a root entity
        parentHierarchyDetails =  childHierarchyDetails;
}

String rootLUName = "" + parentHierarchyDetails.get("luName");
String rootTargetID = "" + parentHierarchyDetails.get("targetId");
String rootSourceID = "" + parentHierarchyDetails.get("sourceId");
```

```
mainOutput.put(rootLUName, parentHierarchyDetails);
//If there are other root entities with the same root entity ID get them,
//they will be added to output as standalone (even if they have their own
hierarchy)
Db.Rows otherRootRecs = fabric().fetch(sqlGetEntityData, rootLUName,
rootTargetID, rootSourceID);
for (Db.Row rootRec : otherRootRecs) {
        Map <String, Object> rootDetails = new HashMap<>();
        String currRootLuName = "" + rootRec.cell(0);
        rootDetails.put("luName", currRootLuName);
        rootDetails.put("targetId", "" + rootRec.cell(1));

//Get instance ID from entity id
Object[] splitId = fnSplitUID("" + rootRec.cell(2));
String instanceId = "" + splitId[0];
rootDetails.put("sourceId", "" + instanceId);

rootDetails.put("entityStatus", "" + rootRec.cell(3));

mainOutput.put(currRootLuName, rootDetails);

}
if (otherRootRecs != null) {
        otherRootRecs.close();
}

return mainOutput;
```