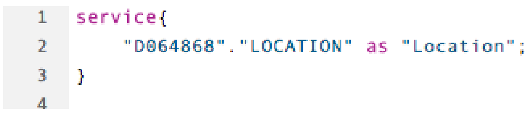
Technical Background Information

This document gives a basic idea of how XSODATA, HANA Sequences and XSJSLIB-Exits work together to implement CRUD operations using ODATA.

XSODATA Services

XSODATA Services are used to expose HANA tables and views using the industry-standard protocol ODATA (Video). In the HANA Workbench Editor or HANA Studio we can define a *service* that another application like a web-app or UI5-Application can consume to populate tables or lists with HANA DB data.
A basic services implementation looks as follows:



```
service{
    "D064868"."LOCATION" as "Location";
}
```

To expose a table or view you have to define it as an entity of the service. In this example the table “D064868”.”LOCATION” is defined as the Entity *Location*.

Action (CRUD)	**Request Type**	**Pattern**	**Additional Information**	**Authentication**
Read	GET	serviceURL/Entity	----	----
Create	POST	serviceURL/Entity	- content type - data to body	Basic Auth
Update	PUT	serviceURL/Entity(id)	- content type - changed data to body	Basic Auth
Delete	DELETE	serviceURL/Entity(id)	----	Basic Auth

ODATA Basic Tutorial

XSJSLIB-Exits

XSJSLIB-Exits are special configurations within a XSODATA definition that tell the XSODATA Interpreter to jump from the XSODATA definition into foreign XSJS coding when certain event are fired or CRUD-operations executed.

HANA Sequences

A HANA sequence is a database object that automatically generates incremented numeric values according the defined rules e.g. the next identifier of a table that can be used for a Create.
HANA Sequences are created by using the HANA Studio or SQL Console of the Web-Workbench (Tutorial / Reference)

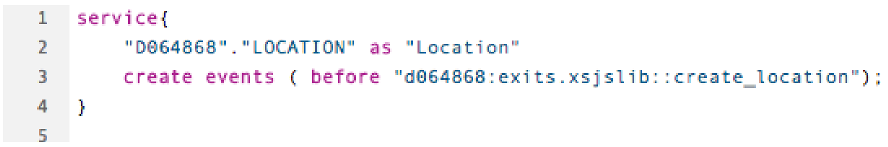
Why XSJSLIB-Exits in our case?

By using HANA Sequences to get an auto-incremented *identifier* on a Create, we have to tell the XSODATA Service Configuration to use the HANA Sequence whenever a Create takes place.

The first naive approach is simply to hand over the HANA Sequence Name as the value the identifier in the body. In the case the service will causes an type error because the value of the identifier is *bigInt* and not *String.* This tells us that the Service is unable to interprete the incoming string as a HANA Sequence Name to use.

The next approach might be not to hand over any value for the identifier to the service. This would be the easiest way but unfortunately it causes in an error telling us that the service requires an identifier value.

We use the XSJSLIB Modification Exit on the create to stop the XSODATA Service when called before executing the Create-Operation to use the parameters handed over but execute the Create manually via XSJS in which we are able to use the HANA Sequences.



```
service{
    "D064868"."LOCATION" as "Location"
    create events ( before "d064868:exits.xsjslib::create_location");
}

```

The service definition above shows how to use this XSJSLIB Modification Exit for our service example. Before the create is executed an event is fired and this configuration catches this event and executes the function *create_relation* from the script *exits.xsjslib* instead with access to the Create-Parameters.

Sources / Further articles

General Introduction
https://blogs.sap.com/2012/11/29/sap-hana-extended-application-services/

General XSODATA
https://blogs.sap.com/2012/12/21/hana-development-xs-odata-services/

XSJSLIB Modification Exit
https://blogs.sap.com/2015/09/21/how-to-return-generated-value-in-xsodata-using-xsjslib-modification-exit/

Create HANA Sequences
https://help.sap.com/viewer/4fe29514fd584807ac9f2a04f6754767/2.0.01/en-US/20d509277519101489029c064d468c5d.html

XSODATA CRUD (Video)
https://www.youtube.com/watch?v=c41anxrDleg