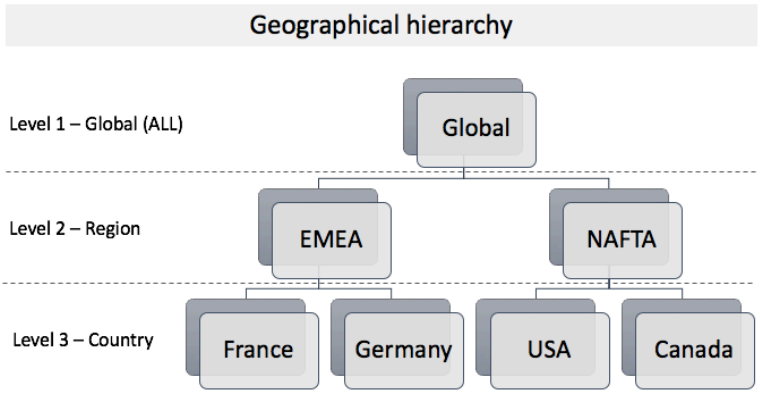
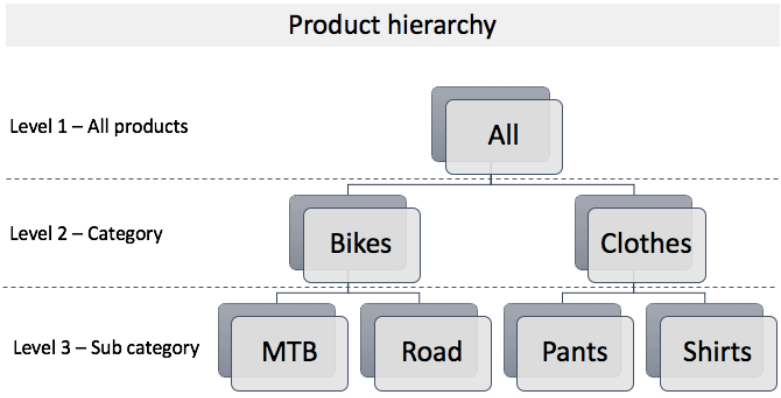
Klywa Access

Access Key Guide

An application/data model need to be adjusted and populated with a row-level access key for full functionality as a multidimensional target in Klywa Access. This document will explain the details why this is necessary and how it is done.

The examples in this document will be based on global sales data where the company wants to slice the data both geographically and product wise. Two hierarchies are identified and looks like this.





A Target can be sliced on any level in the included hierarchies. Even if an application has data on Country level, Regional level might be enough to fulfill the slicing needs. Therefore, each Target is configured in Klywa Access with information at which level the slicing will operate.

Update of access rights for a Target is done by requesting Klywa Access for current access keys. This is easily done with the PowerShell Scrips supplied from Klywa. (Alternatively build a customer specific solution using the REST API.)

By sending a Target-ID in the request, Klywa Access will use the current Target settings and construct and return the complete set of current access keys.

An order line access key for a mountain bike sold in France could look like "MTB|FR". (Product Code followed by a pipe-sign and finally a country code.)

Joker sign

A very powerful feature is the joker-sign included in the access keys from Klywa Access. Imagine a new sub category "Caps" added in the source system. A manager given access to all products do want to see all products – not only those currently configured in the access system.
This is solved in a two-step process:

1) Klywa Access will automatically add extra access keys to all users given complete access (to one or both of the included hierarchies). These extra keys will have a joker sign, "¤" meaning "all currently unknown id's" assigned in the position of the complete hierarchy.

2) A request for new access keys will also have a complete set of currently known id's in return. When adding the access keys in the local application, replace all sub-category id's not known to Klywa Access with the joker-sign.

The access keys giving the manager access to the new Cap sales would look like: "¤|FR", "¤|DE", "¤|US", "¤|CA" – four keys, one for each country.

Implementation example

The first thing to check is that the data model includes data fields suitable for reduction of the content for the end users. These fields should be part of the hierarchies added into Klywa Access.
Even if the data model has fields with details on lower level in the hierarchies, access level should be selected based on reducing needs – hierarchy level to be used is defined per target.

If the identified fields are not part of a current hierarchy two options are available. Either add a new Domain in Klywa Access with a hierarchy in line with the fields, or use a local mapping file to translate the field data according to an existing hierarchy.

The picture below illustrates parts of a common star schema data model with technical keys connecting the fact table with the dimensional tables.
It is always preferred, but not mandatory, with a clean data model including a single fact table.

Product Dim

PID	*Product*	*GroupCode*
29	MTB 420 x2	MA
52	Road 25b	R2

Fact Table

PID	*CID*	*Currency*	*Value*
29	43	USD	1 200
52	34	USD	980
29	54	USD	1 220

Customer Dim

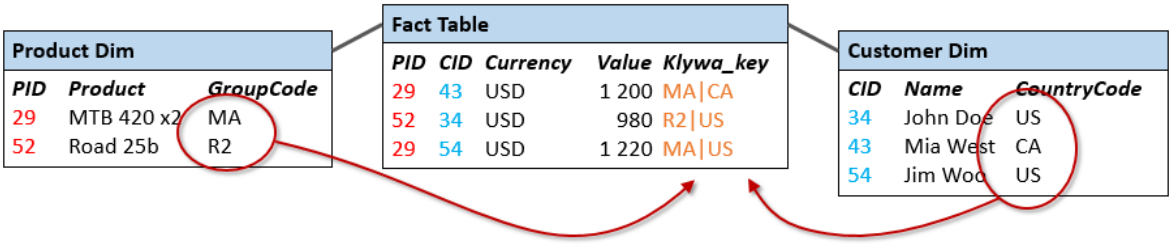
CID	*Name*	*CountryCode*
34	John Doe	US
43	Mia West	CA
54	Jim Woo	US

In this example Klywa Access has a Product hierarchy with a suitable level named Product Group that will perfectly match the GroupCode in Product table and an Organizational hierarchy with a level called Country that will match the CountryCode-field in the Customer-table.

Klywa_key

The access key "Klywa_key" connects the Targets data model with the access tables generated by Klywa access. This key is best added in the central fact table.
When a User is accessing a Target powered by Klywa Access, the data model will be filtered based on the specific values in the Klywa_key-field that is connected to the authenticated User.

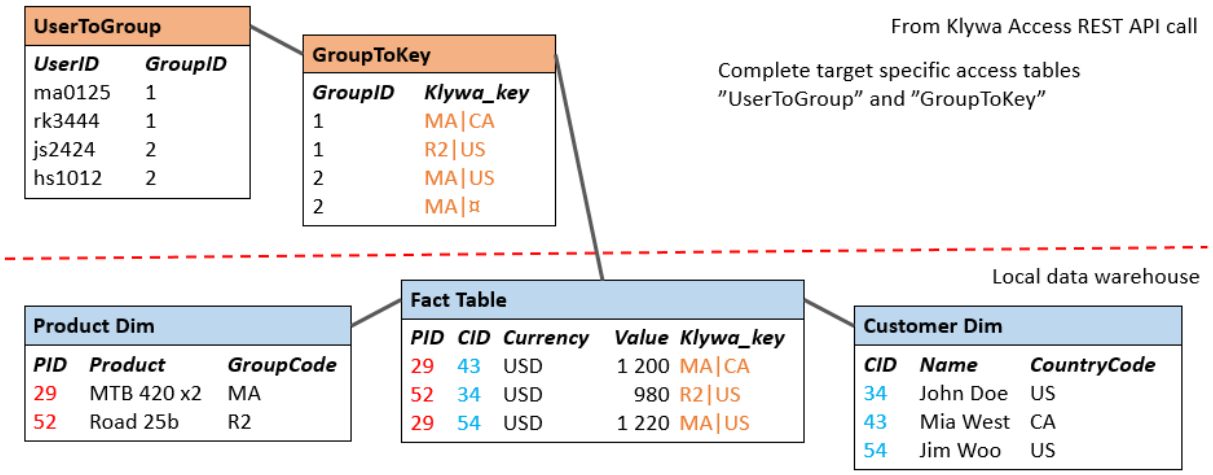
In this example "GroupCode" is joined from Product Dim and "CountryCode" is joined from Customer Dim into the fact table to generate the "Klywa_key" -field.



The Klywa_key-field is a concatenated string with the two values separated by a pipe-character.

Klywa tables

Klywa access will return two ready built tables when requested for updated access information. The request is unique per Target, and the returned data is based on ID's from the hierarchies as configured in the access system. The first table will include the "Klywa_key"-field that will match the generated access field as specified above. The tables are connected via a Group-filed and in the second table is the end user identification field.

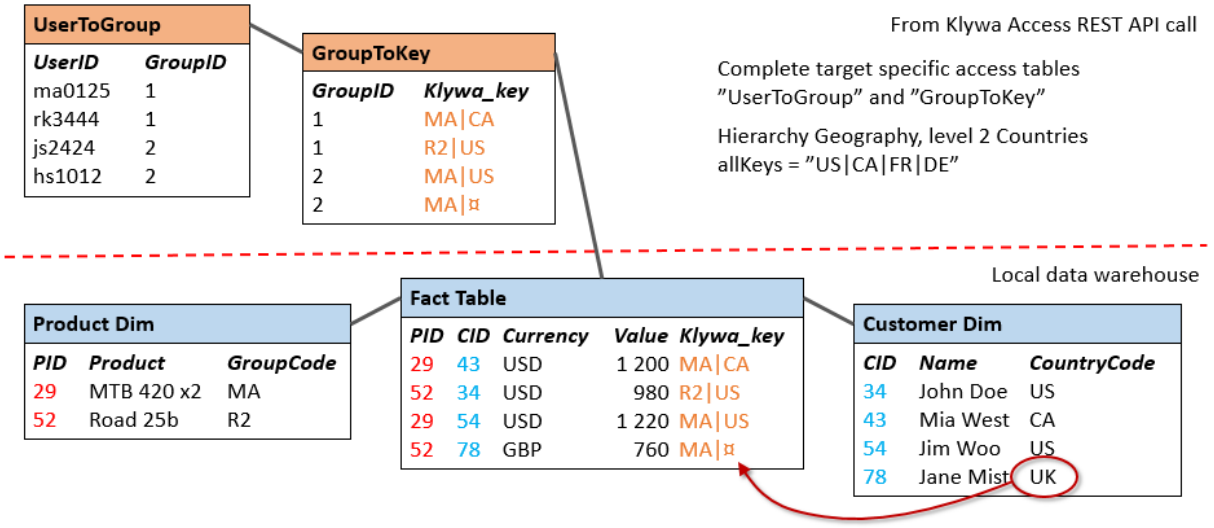


Joker-sign example

What if the data model includes new values (in the access fields) that are not known by Klywa Access? The joker-sign will guarantee that all Users with complete access to a specific hierarchy always will see all transactions regardless value in that hierarchy.

Again - How does this work?
When a Target calls Klywa Access for the current access keys, the system will also return the currently known complete set of values used to build the access key. If a value for a transaction

is included in the known set, the value is kept; otherwise the joker-sign will be used in the access key instead.



The joker-sign is the same as a "system-unknown-value" and all users with access to a complete hierarchy will be assigned special access keys covering these "system-unknown-transactions".

While processing the joker-sign, it will also be possible to monitor these unknown values and thereby inform the access administrator that Klywa Access should be updated.

Klywa access is configured with a geographical hierarchy that includes a level with countries. The complete list of countries is USA, Canada, France and Germany, with corresponding ID´s; US, CA, FR and DE.

This list is sent as an array as part of the REST API response.

```
"hierarchies": [
    {
      "hierarchyId": 365,
      "name": "Geography",
      "description": "Customer location",
      "coverageDepth": 1,
      "depths": [
        {
          "depth": 0,
          "name": null
        },
        {
          "depth": 1,
          "name": "Country"
        },
      ],
      "allKeys": "US|CA|FR|DE"
```

This list need to be matched to the distinct set of ID's in the local data base before generating the final "Klywa_key"-field in the fact table. All local ID's not represented in the array from Klywa access is replaced (or represented) with a joker-sign in the key-field.

In a Qlik loading script this is efficiently solved with the ApplyMap-function. In SQL-server temporary tables is one good option.