ORANGE FT GROUP

User Guide NodeJS SDK

SDK Datavenue platform

18/05/2015

This document is the user guide for the Datavenue NODEJS SDK.

Version history

Version	04/15/2015
1.0.0	
Version	05/18/2015
1.0.1	

orange

CONTENTS TABLE

Version 1.0.1

GLOSSARY

Key terms used in the document.

A

Account: A customer account. The creation of an account in the Datavenue portal is a prerequisite to use the SDK NODE JS.

API key: An API Key allows an application to access the operations of a single Data-source.

C

Authenticate keys: Using the SDK requires two separate keys, one to access the service (access key Orange Partner) and to be authenticated on the service (primary Master Key, Master key or key API).

D

Datavenue: Set of services offered by Orange for the Internet of Things.

Device: This is an "object" (connected object, server, data aggregator ...) producing data.

Data-source: This is the set of elements associated with a device. This includes "streams", the each data stream, and APIs keys and meta-data device.

R

Resource Identifier: In the structures used in the SDK, there are link type fields. It is a REST resource identifier.

CHAP 1: GENERAL SERVICE DESCRIPTION

1. A use case

Before presenting in detail the main concepts of Datavenue service, here is a use case that helps place the service in its context.

1.1. Context

As a company, I developed connected objects that provide atmospheric data (temperature and humidity). I want to store all my objects data to a centralized platform.

To do that, I'll have to:

- Join the Datavenue service.
- Create an account on Datavenue portal with my login.
- Get my passkey on the Orange Partner web site (OPEkey)
- Create an authentication key (Master Key)
- Create a prototype data source, this datasource represents by object. On this datasource, I'll define two streams, one for each kind of data (temperature and humidity)
- Test the communication between my prototype and Datavenue platform by sending data (values) in each of these "streams ".
- If I am satisfied with my prototype. I'll fix the structure of my data model by creating a "Template" datasource. I'll use this template to massively create datasources (one datasource by item sold). Each "datasource" will have two "streams" of data to collect data emitted by the objects.

1.2. Conception phase

During conception, I create my prototype in which I will define two "streams": temperature and humidity.

Once I created my streams, I can also create several API Keys, for example an API Key for the object, one for the mobile app, in order to identify my actors and control their access.
If I define a specific API Keys for my actors (objects, my applications - e.g. mobile), these keys will be re instantiated (key values remain unique and will not be duplicated between data-sources) automatically for each model that inherits the datasource template
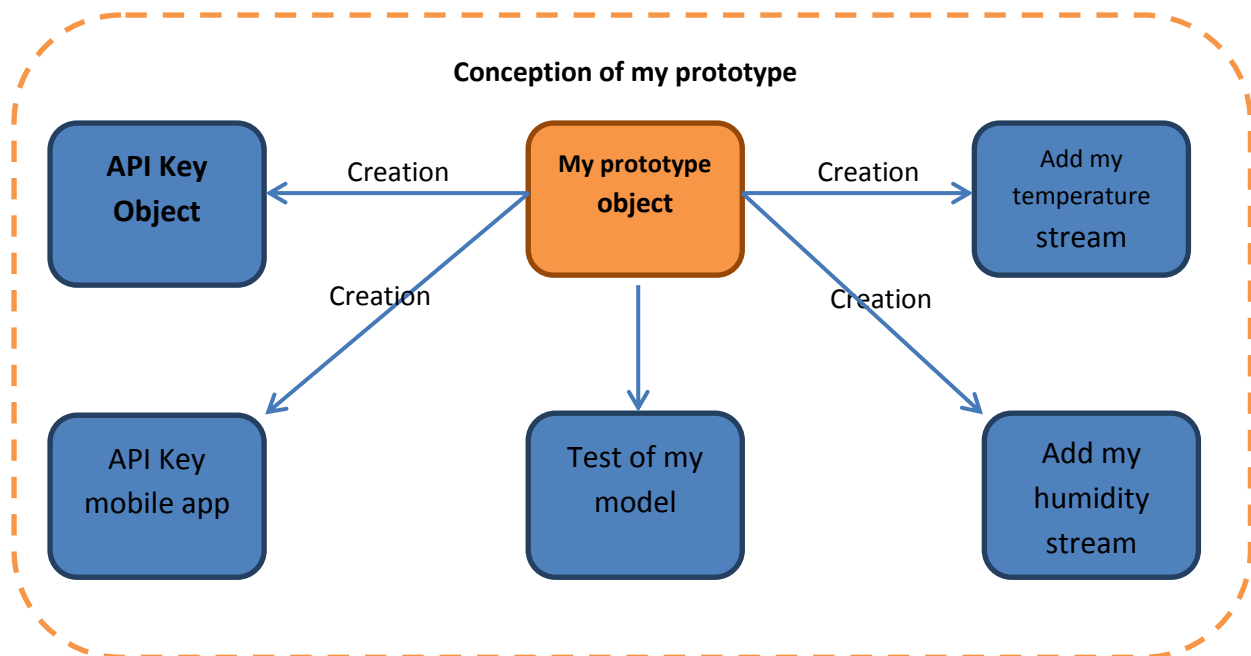
Conception of my prototype

| API Key Object | ← Creation | My prototype object | Creation → | Add my temperature stream |

Creation → API Key mobile app

↓ My prototype object → Test of my model

Creation → Add my humidity stream

Figure 1: Prototype conception

My data model being finalized, I can begin to test before validation

1.3. Template creation

Once my prototype is working and satisfying the requirements, I can extract a "template" with the current data model, this template will be used to generate datasource, and my prototype will be able to evolve following the objects evolutions

The datasource will inherit from the template (data can only be added to a datasource, not a template), so all datasource will have the same characteristics as the template (streams, units, number of API Keys, callback parameters…) but will instantiate some data (like API Key for example)
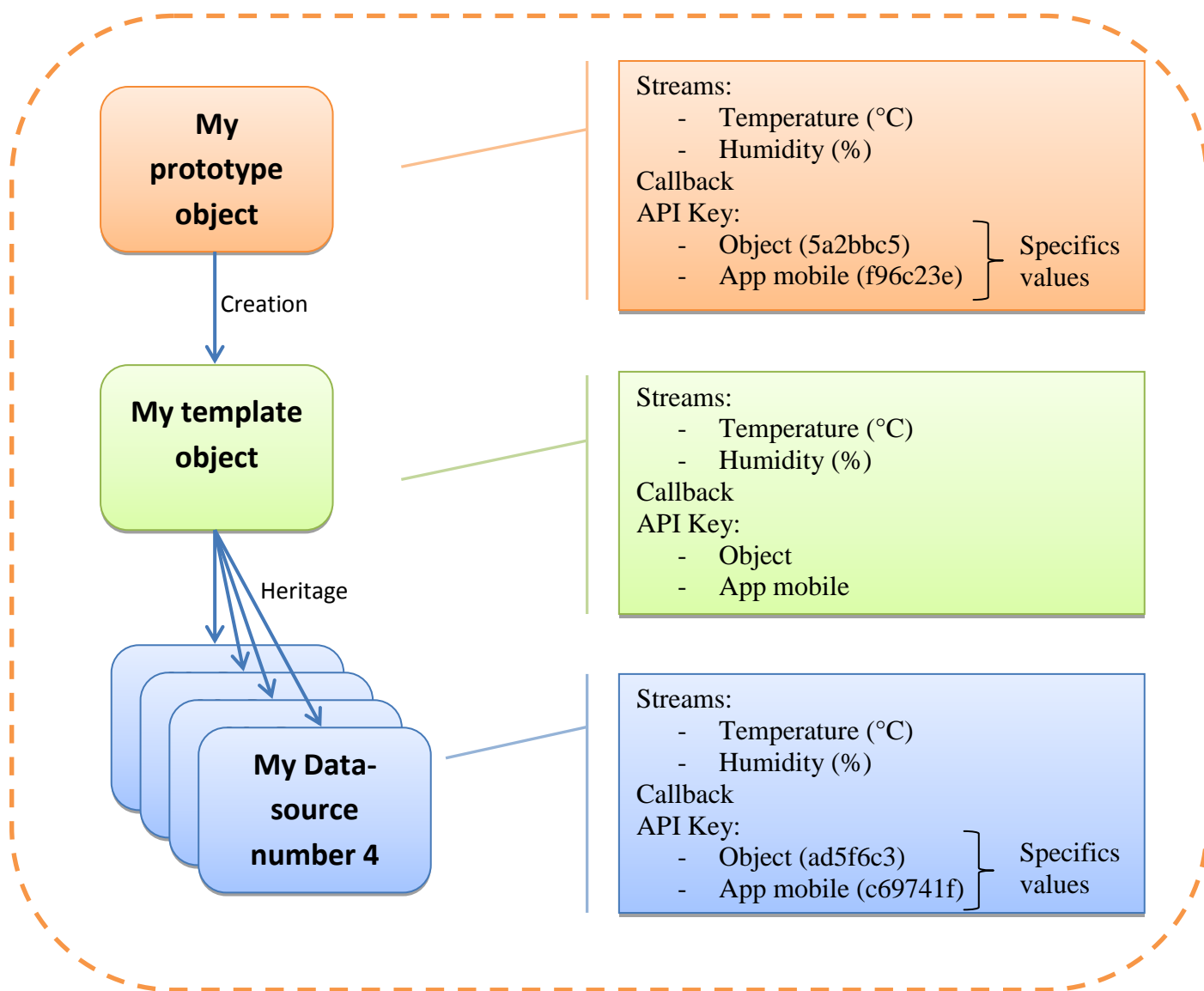
Version 1.0.1

1.4. Data-source evolution

Once the data-source created from a template, I can enrich it with new streams, new API Keys, or delete keys, of course the modifications on a datasource won't impact other datasource, the template or the prototype.

1.5. Template and prototype evolution

After creating the template, the prototype can evolve without changing the templates already created.

For example if on the V2 version of my object there is a new captor, I can add a new stream to my prototype in order to store the data uploaded by the new captor. The prototype will evolve, following objects evolution, and a template will be a fixed version of the prototype. This is why the datasource are based on the template and not on the prototype

Version 1.0.1

2. Presentation of the main concept

2.1. Context

The IoT (Internet of Things) service has been designed to help IoT companies to develop services around the connected objects. For this, Datavenue helps to centralize data produced by different devices on the same platform, accessible from the Internet via APIs.

The aim is to assist IoT developers by offering a centralized storage, and all the helpful APIs to access the data

2.2. Service access

The service is accessible from the internet. Access URL is:
https://datavenue.orange.com/api/v1

Access to the API Datavenue is secured by Orange Partner. To use the API, you must have previously subscribed to Datavenue and retrieved your Orange Partner id. This id will be used in API requests, with the authentication Key provided by Datavenue.

2.3. Resource model

The following diagram defines Datavenue resource model with its main entities (account, prototype, key, data-source, template, and stream). The relationships between these entities are specified (UML convention).
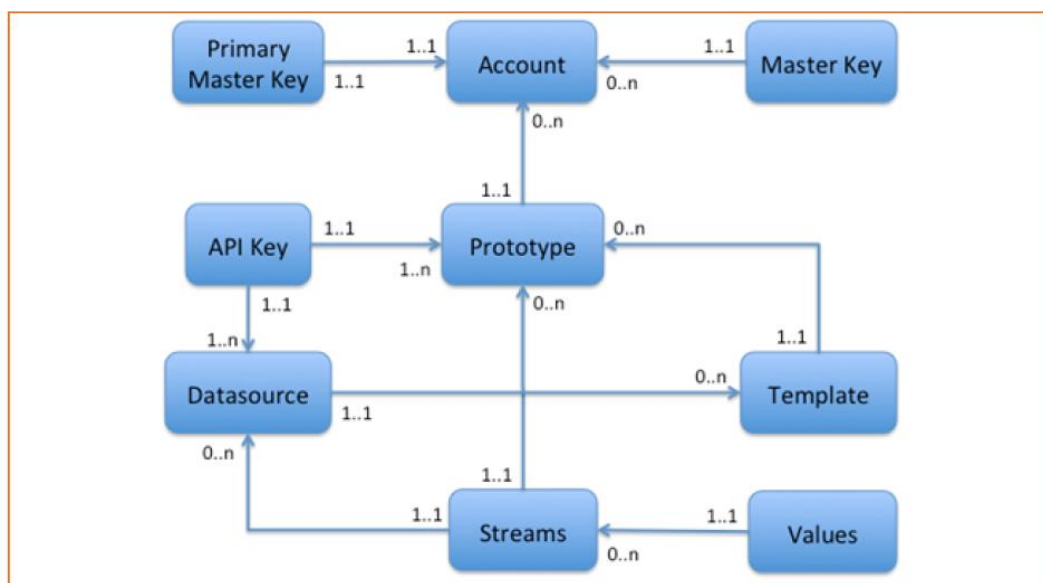

Figure 3: Datavenue resource model

Version 1.0.1

An access key to Datavenue services may be a 'Primary Master Key ', ' Master key' or ' API Key ' with specific privileges. For example, the key ' Primary Master Key' is required to create key ' Master key'. and a key ' Master Key' is needed to create prototypes and data-sources.

2.4. Resource relationship

The following diagram shows the dependencies between the components used in Datavenue.
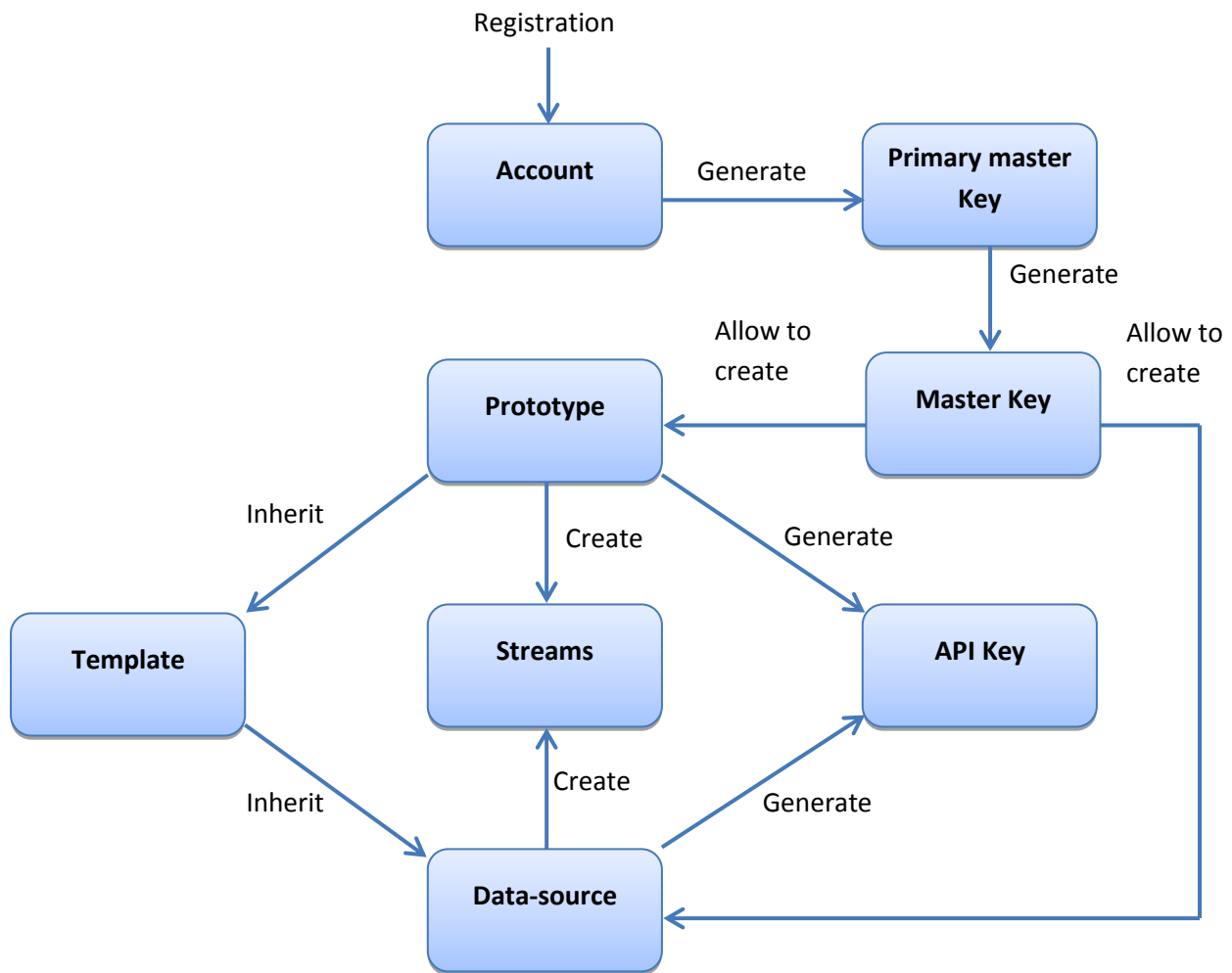
2.5. Authentication keys

2.5.1. Generality

The keys are used for authentication and resources access management. They are needed to perform requests to Datavenue.

2.5.2. The differetn types of Key

There are 3 types of keys that can be used to make calls on Datavenue API:

- **Primary master key:** key to the administration and management of your account. This key is created during account creation on the Datavenue. It cannot be created with the SDK. But, it can be used in the SDK, for example, to list the customer account information or create a Datasource Management Key (Master key).

- **Master key:** these are access keys to the different customer datasources. They can be created using the Datavenue portal or the SDK. The master keys are used to manage all the resources (prototype, templates, datasources, streams, values. they are also used to manage tha access keys (APIKeys)

- **API key:** it's a key used to access and manage datasources in particular.
- It is created using either the portal Datavenue or, using the SDK.

2.5.3. Relations between different keys and functions Datavenue

2.5.3.1. Primary master key

For security reasons, an account has one"Primary Master Key" which only be used to manage the "Master Key" . It is important that management (CRUD) of "Master Keys" is done independently to ensure that account manager can control access to its data, and repudiate a faulty MasterKey Therefore, a PMK can:

- Create a Master Key (associating them rights : GET / POST / PUT / DELETE)
- Change / regenerate a Master Key
- Delete Master Key
- List all master keys linked to an account
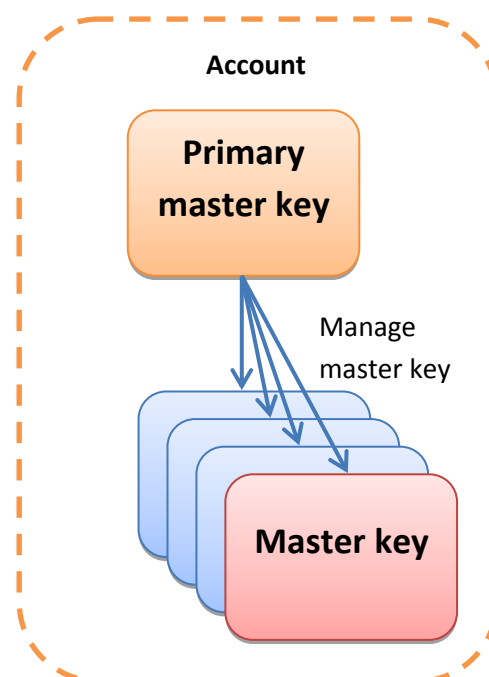
2.5.3.2. Master key

A master key allows an application to access operations on all datasources of an account. Depending on the rights that are associated with, a master key may be used on all data-sources belonging to the client account.

2.5.3.2.1. List right (GET)

If a master key has the rights to GET, it may do the following on all data sources:

- List all prototypes of the account and get the information associated with them
- Read the information of a prototype
- List the streams belonging to a prototype
- Read the information of a stream belonging to a prototype
- Read the collected values on a stream belonging to a prototype
- List and read the API key prototype
- Regenerate the prototype API Key
- List all templates of the account and obtain the information associated with them
- Read the information in a template
- List all data-sources account and get the information associated with them
- Read the information in a data-source

Version 1.0.1

- List the streams belonging to a data-source
- Read the information from a stream belonging to a data-source
- Read the collected values on a stream belonging to a data-source
- List and read the API keys data-source
- Regenerate Key data-source API

2.5.3.2.2. Create right (POST)

If a master key has the rights POST, it may do the following on all data-sources:

- Create a new prototype
- Create a stream in a prototype
- Add Value in a stream of a prototype
- Create an API key for a prototype
- Create a template
- Create a new data-source
- Create a stream in a data-source
- Add one or more values to a stream of a data-source
- Create an API key to a data-source

2.5.3.2.3. Update right (PUT)

If a master key has the rights PUT, it may do the following on all data-sources:

- Update a prototype
- Update a stream of a prototype
- Update the information of a template
- Update the information in a data-source
- Update the information of a stream in a data-source

2.5.3.2.4. Delete right (DELETE)

If a master key has the rights DELETE, it may do the following on all data-sources:

- Delete a prototype
- Delete a stream in a prototype
- Delete one or more values in a stream of a prototype
- Delete an API key for a prototype
- Delete a template
- Delete a data-source
- Delete a stream in a data-source
- Delete one or more values to a stream of a data-source
- Delete an API key to a data-source

2.5.3.3. API key

An API Key allows an application to access the operations of a single data-source. All API Key are unique and provide access to a single data-source, as opposed to those Master Keys can address all data-sources an account.
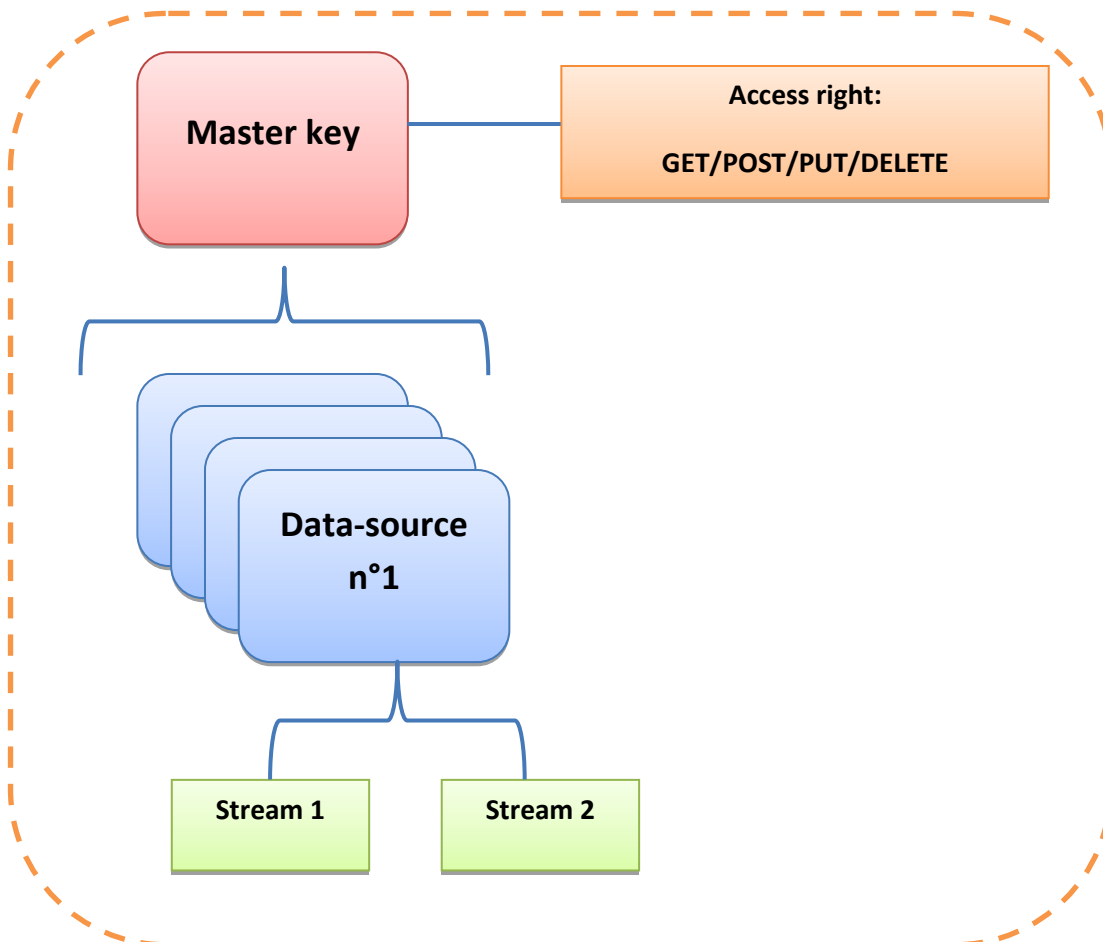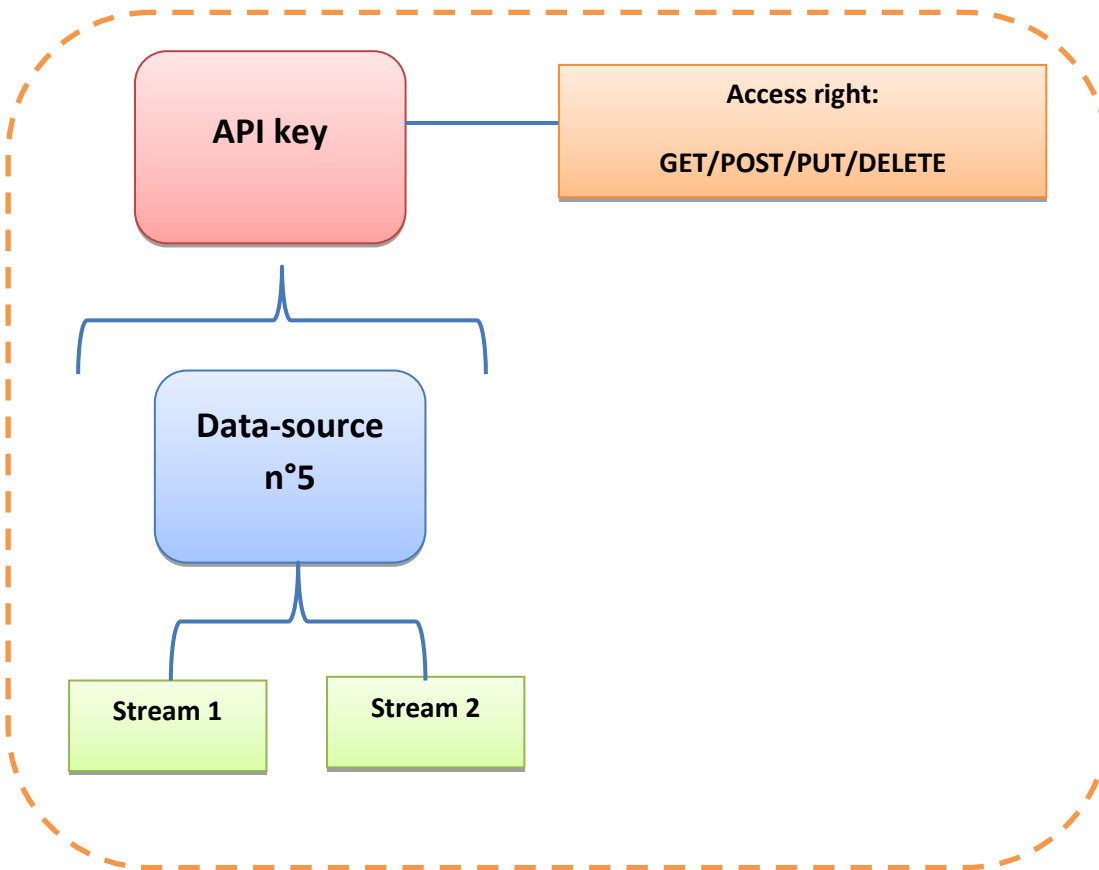
Figure 6: API key and data-source relationship

2.5.3.3.1. List right (GET)

If an API key has the rights to GET, it may do the following on all data-sources:

- Read the information in a data-source
- List the streams belonging to a data-source
- Read the information from a stream belonging to a data-source
- Read the collected values on a stream belonging to a data-source
- List and read the API keys data-source
- Regenerate Key data-source API

Version 1.0.1

2.5.3.3.2. Create right (POST)

If an API key has the rights to POST, it may do the following on all data-sources:

- Create a stream in a data-source
- Add one or more values to a stream of a data-source
- Create an API key to a data-source

2.5.3.3.3. Update right (PUT)

If an API key has the rights to PUT, it may do the following on all data-sources:

- Update the information in a data-source
- Update the information of a stream in a data-source

2.5.3.3.4. Delete right (DELETE)

If an API key has the rights to DELETE, it may do the following on all data-sources:

- Delete a data-source
- Delete a stream in a data-source
- Delete one or more values to a stream of a data-source
- Delete an API key to a data-source

2.5.3.4. Summary table of keys and associated functions

Entity		Create	List	Get	Update	Delete	Regenerate
Prototype		Master key	Master key	Master key	Master key	Master key	
	stream	Master key	Master key	Master key	Master key	Master key	
	value	Master key	Master key	Master key		Master key	
	API key	Master key	Master key	Master key	Master key	Master key	Master key
Template		Master key	Master key	Master key	Master key	Master key	
Data-source		Master key	Master key API key	Master key API key	Master key API key	Master key API key	
	stream	Master key API key	Master key API key	Master key API key	Master key API key	Master key API key	
	value	Master key API key	Master key API key	Master key API key		Master key API key	
	API key	Master key	Master key API key	Master key API key	Master key	Master key	Master key API key

orange™

3. SDK node js: What is it for ?

The Datavenue API uses a REST / JSON architecture over HTTPS.

The Node.js SDK includes:

- A library that runs in the client code, which provides all the technical part for REST / JSON calls. The developer of the client application can focus on data structures and javascript methods.

- A sample code in nodeJS for the implementation of the library.

- A user manual that presents the service.

Example: the developer of the client application writes the following code in NodeJS, using the library SDK NodeJS Datavenue.

```javascript
var datavenue = require('datavenue);
var datasource = new datavenue.Datasource.Datasource();
var bodyDatasource = {
     name : "NodeJS",
     description : "Datasource de test pour SDK nodejs"
};
var parameters = {
      xISSKey : MASTERKEY,
      body : bodyDatasource
};
datasource.createDatasource(parameters).then(
function(data) {
   console.log('Datasource "' +
      data.body.name + '" created.');
});
```

The NodeJS SDK will transform this javascript method to a REST request.

4. Installation

Simply use npm for install the SDK:

```
npm install datavenue
```

CHAP 2: BEFORE STARTING

1. Delivery presentation

The package is presented in the form of a zip file containing the following files:

- The user guide
- The NODE JS SDK
- A sample that use the NODEJS JAVA.

2. User account

Before you can use the SDK NODEJS Datavenue, you must have:

- an Orange Partner Authentication Key (you will get it at registration)
- A Primary Key Master or Master Key or API Key Datavenue.

You can get your Primary Master Key on the portal Datavenue when creating the user account.
The Master Key or API key can be created either using the portal Datavenue or using the SDK NODE JS.

3. Proxy and firewall

Before using the SDK, check with your network administrator that the firewall does not filter the access to URL Datavenue service.

The SDK has no proxy configuration. See sample proxy part, for more details.

CHAP 3: ERROR MANAGEMENT

For the NODEJS SDK version 1.0.2, there is no error management. Actually the SDK return for each function the Datavenue platform response like a "promise" (See here for more details: https://www.npmjs.com/package/q). This promise is an object structure and contains the Datavenue error:

Error code	Error type	Description
901	Access denied	
910	Resource not found	The account {account_id} is not found
911	Resource not found	The datasource {datasource_id} is not found
912	Resource not found	The stream {stream_id} is not found
913	Resource not found	The prototype {prototype_id} is not found
914	Resource not found	The template {template_id} is not found
915	Resource not found	The value {value_id} is not found
916	Resource not found	The key {key_id} is not found
920	Invalid input data	The required field {field} is required
921	Invalid input data	{field} value must be a {type}
922	Invalid input data	The field {field} is badly formatted
923	Invalid input data	{field} value is not supported
924	Invalid input data	{field} value exceeds xx characters
925	Invalid input data	The parameter {param} is badly formatted
926	Invalid input data	{param1} could not be higher to {param2}
927	Invalid input data	A value is required fot the parameter {param}
928	Invalid input data	The field {field} is empty
929	Invalid input data	The value size exceeds 5Mo
930	Naming conflict	The email {email} already exists
932	Naming conflict	The prototype name {prototype_name} already exists
933	Naming conflict	The stream name {stream_name} already exists
934	Naming conflict	The key name {key_name} already exists
950	Internal Error	The account {account_id} is corrupted
951	Internal Error	The datasource {datasource _id} is corrupted
952	Internal Error	The stream {stream _id} is corrupted
953	Internal Error	The prototype {prototype _id} is corrupted
954	Internal Error	The template {template _id} is corrupted
955	Internal Error	The value {value _id} is corrupted

Figure 7 : Errors list of Datavenue platform

Version 1.0.1

Entity	Create	List	Get	Update	Delete	Regenerate
Account			901, 910	901, 910, 921, 924, 928		
Master key	901, 920, 924, 928, 934	901, 910	901, 910, 916	901, 916, 920, 924, 928, 934	901, 910, 916	901, 910
Prototype	901, 920, 924, 928, 932	901, 925	901, 913	901, 913, 921, 924, 928	901, 913	
Template	901, 913, 924, 928	901, 925	901, 914	901, 914, 924, 928	901, 914	
Datasource	901, 911, 913, 928, 934	901, 925	901, 911, 912, 913	901, 911, 912, 913, 924, 928	901, 911, 912, 913, 923	
Stream	901, 911, 913, 924, 928, 934	901, 913, 925	901, 911, 912, 913	901, 911, 912, 913, 924, 928	901, 911, 912, 913, 923	
Value	901, 911, 912, 913, 921, 922, 928	901, 911, 912, 913, 926, 927	901, 911, 912, 913, 926, 927		901, 911, 912, 913, 915	
API key	901, 920, 928, 934	901, 910	901, 916	901, 916, 928, 934	901, 910, 916	901, 913, 916

Figure 8: Table errors codes and related functions

CHAP 4: NODEJS SAMPLE

1. Proxy

The sample uses the module global-tunnel for configure the proxy. For more detail see:
https://www.npmjs.com/package/global-tunnel

The global-tunnel configuration use environment variable (http_proxy).
Be cared to set this variable in your environment system.

2. What it does?

The NODEJS sample is a Web server.

When you run it, you get a Web page at localhost:8080.

Put your information on this page and the sample will execute some calls to Datavenue platform.
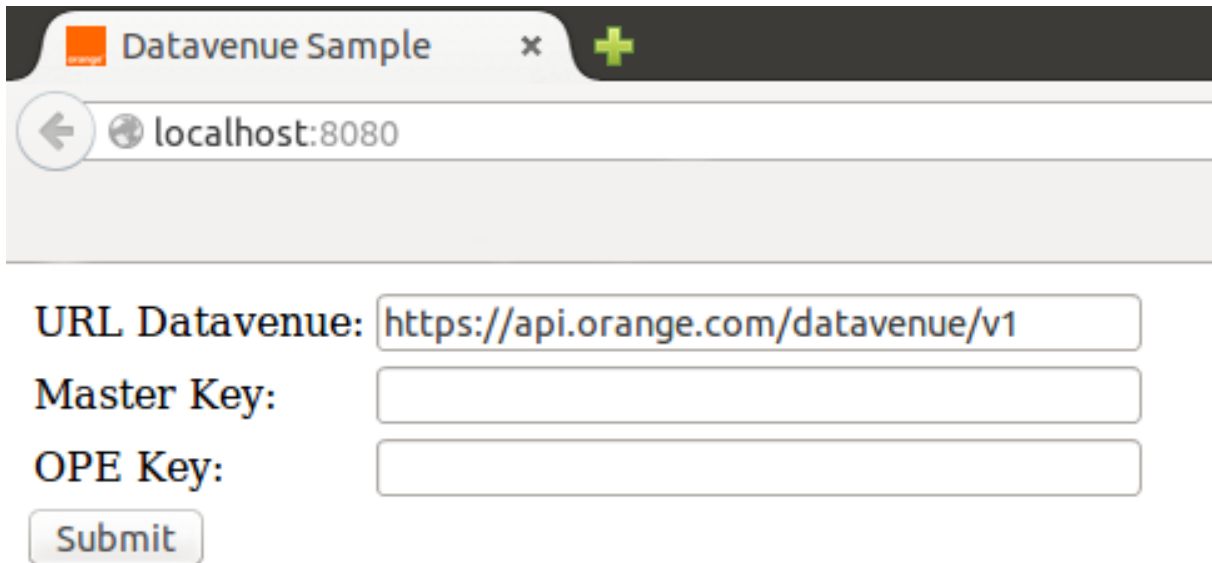
Execute calls:

- Create template
- Update template
- Get template
- Delete template

- Create datasource
- Update datasource
- Get datasource
- Create an API Key
- Delete datasource

- Create prototype
- Update prototype
- Get prototype
- Create an API Key
- Delete prototype

18

orange

3. How to run NODE JS sample

- Extract sample code.
- Run prompt line and go to sample folder.
- Run : node serveur.js (need to have install node)
  You should get: Starting web server at 127.0.0.1:8080
- Go to your web browser at 127.0.0.1:8080 (localhost:8080).
- You should get this web page:

- Put a master Key and your Orange Partner Key (you can get these keys on your account at https://datavenue.orange.com)
- Click on "Submit".
- If you check server log, you will see the calls done.