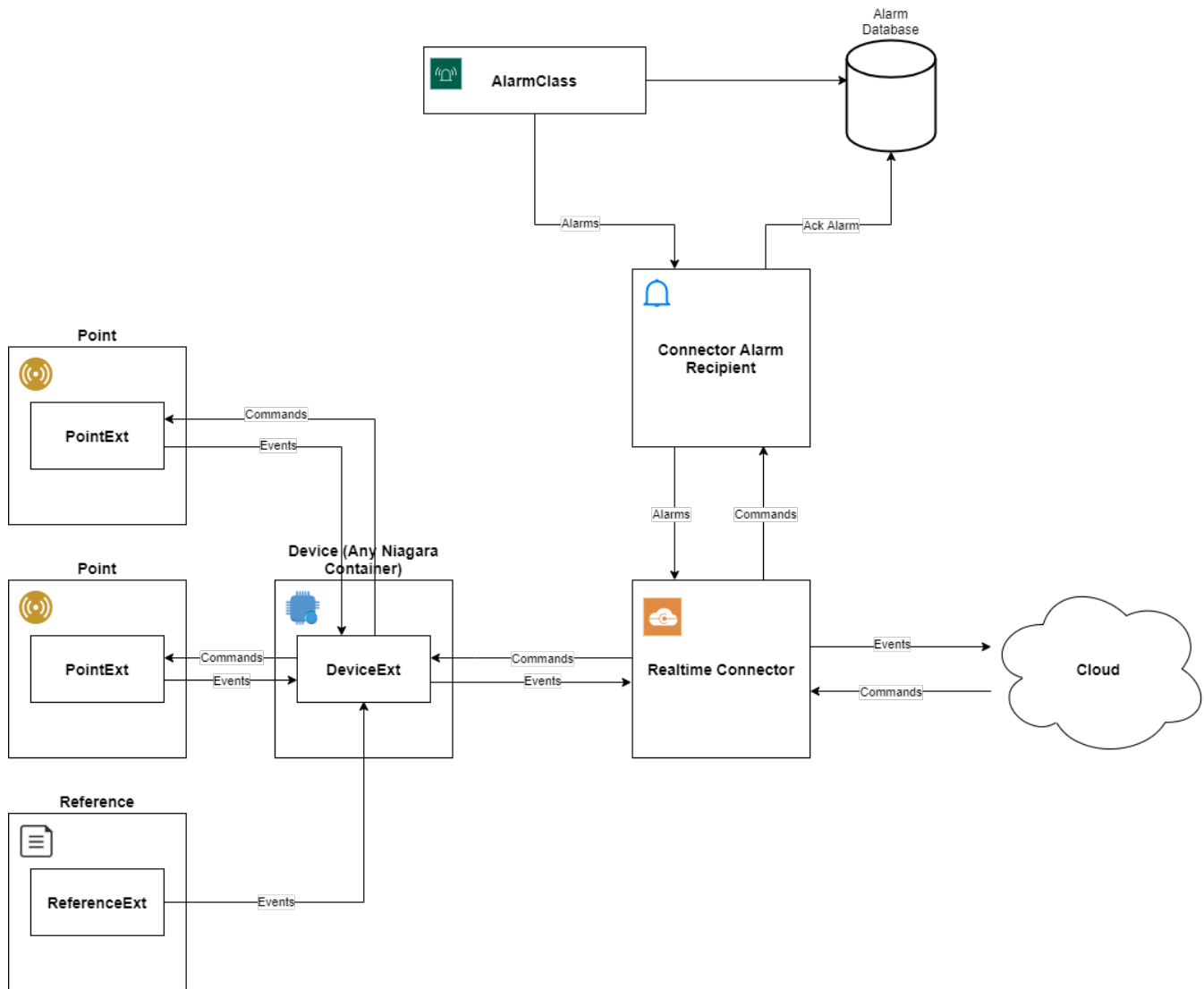


Architecture

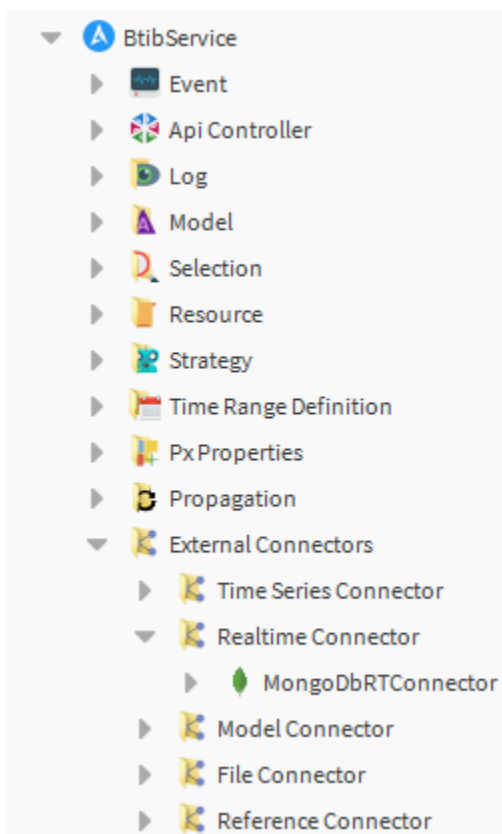
The base architecture is common to all the real-time connectors. It is build upon several components:

- **RealtimeConnector:** The component responsible for managing the connection to the remote service. Located in the BtibService.
- **DeviceExt:** An extension added to any Niagara container to extend its capability to connect to the remote service, and it uses the connector to access it.
- **PointExt:** An extension added to any control point in Niagara to extend its capability to connect to the remote service, and it uses the device to access it.
- **ReferenceExt:** An extension added to any reference component to extend its capability to connect to the remote service, and it uses the device to access it.
- **ScheduleExt:** An extension added to any Weekly Schedule to extend its capability to connect to the remote service, and it uses the device to access it.



Connector component

A real-time connector is a component to install in the Btib Service in the appropriate folder. Example with Mongo:



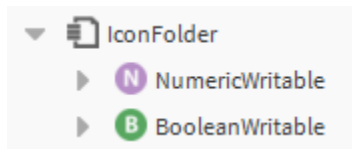
A connector hosts the credentials and defines the general parameters (that you can explore later)

MongoDbRTConnector (Mongo Db Connector)	
Log Ext	System Log Ext
Status	{ok}
Fault Cause	
Enabled	<input type="radio"/> false
Last Attempt	04-Mar-2022 01:08 AM CET
Last Success	04-Mar-2022 01:08 AM CET
Last Failure	20-Aug-2021 12:02 AM CEST
Auto Provision	<input checked="" type="radio"/> true
Use Data Retention	<input type="radio"/> false
Data Retention Duration	+00072h 00m 00s
Data Send Retry Duration	+00000h 01m 00s
Messages Thread Pool Size	5
Advanced Config	Advanced Config
Number Of Worker Threads	1
Number Of Connections Per Host	5
Connection String	••••••••

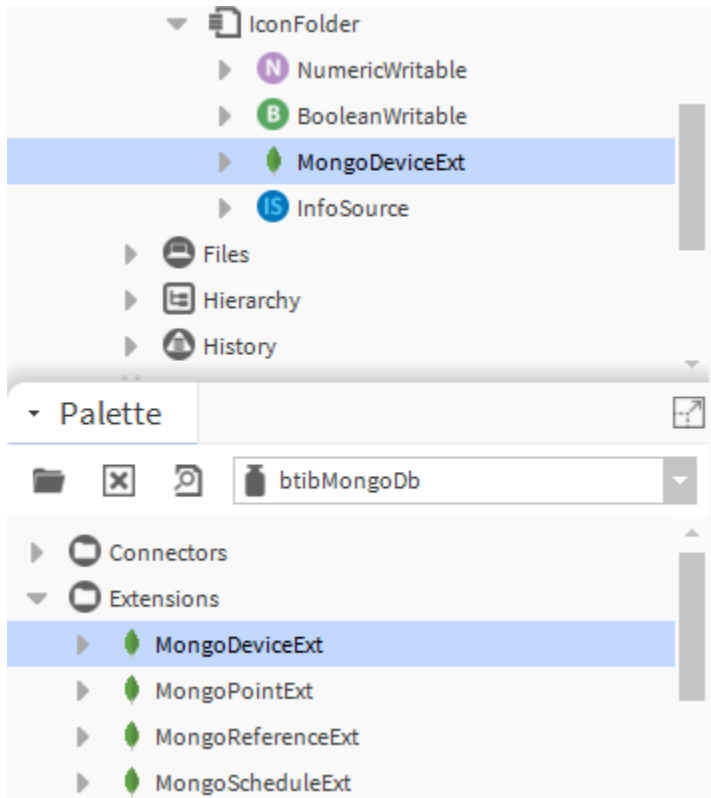
Container / Datapoint

Data points are not transferred directly to the database or cloud platform. They are grouped by a container which can be a device, a Folder, any component in Niagara.







If we take a very simple example with a folder containing two points:



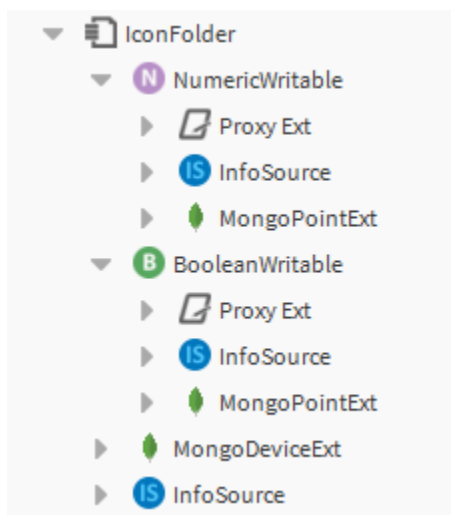
One extension, called a "DeviceExt" must be added from the palette into the data points container. (An InfoSource will popup automatically to provide a unique id to the container)



The DeviceExt is used to define which connector it should be sending data to (You can use several connector of the same type in the station)

 MongoDeviceExt (Mongo Device Ext)	
 Status	{fault}
 Fault Cause	Cannot register device IconFolder: Conne
 Enabled	<input checked="" type="radio"/> true
 Connector	 MongoDbRTConnector

Then, we add a PointExt to each data point (an InfoSource is also added automatically to generate a unique id to the point).










The PointExt defines the behaviour for the point changes: when it should send messages to the cloud, there are several options.

One of the most important property is the **DeviceQuery**. This is where we indicate where is the container (the PointFolder in our case). It can accept absolute Ord, relative Ord, [SFormat...](#)

In our example, we want to reach the grand parent of the PointExt (the IconFolder).

MongoPointExt (Mongo Point Ext)

▶ Advanced Config	Advanced Config
Status	<input data-bbox="548 961 782 993" type="text" value="{ok}"/>
Fault Cause	<input data-bbox="548 999 1010 1031" type="text" value=""/>
Enabled	<input checked="" type="radio"/> true 
Device Query	<input data-bbox="548 1087 1237 1119" type="text" value="slot:../.."/>    
Trigger On Value Change Only	<input checked="" type="radio"/> true 
Can Write	<input checked="" type="radio"/> true 

Now every time a changes occurs at the point level, either its status or its value, a message will be sent to the database/platform.

A third party can also change data point value.