Apama Connectivity For Cumulocity IoT

Version 10.1

November 2017

APAMA

This document applies to Apama Connectivity For Cumulocity IoT Version 10.1 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: PAU-101-20171127

Table of Contents

About this Guide....5
Documentation roadmap.... 5
Online Information.... 5
Contacting customer support....6

Release Notes.... 7
What's new in 10.1 Release.... 8
What's new in 10.0 Release.... 10
What's new in 9.12 Release.... 10

Installing Apama Connectivity For Cumulocity IoT.... 11
Installation.... 12
Migration from earlier versions to 10.1....12
Configuration Parameters....17
Monitor Injection Order....18

Working with Apama Connectivity For Cumulocity IoT....19
Introduction.... 20
Creating an Application Key....20
Developing with Software AG Designer....20
Manual Deployment....20

Interacting with Cumulocity IoT....23
Application Startup....24
Device Events....24
Measurement Events....24
Events and Alarms.... 25
Sending control events to Cumulocity IoT.... 25
Sample EPL....26

About this Guide

This guide describes how to configure the Apama Connectivity For Cumulocity IoT.

Documentation roadmap

Apama Connectivity For Cumulocity IoT provides documentation in the following formats:

- HTML (viewable in a web browser)
- PDF (available from the documentation website)

Online Information

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at http://documentation.softwareag.com. The site requires Empower credentials. If you do not have Empower credentials, you must use the TECHcommunity website.

Software AG Empower Product Support Website

You can find product information on the Software AG Empower Product Support website at https://empower.softwareag.com.

To submit feature/enhancement requests, get information about product availability, and download products, go to Products.

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the Knowledge Center.

Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at http://techcommunity.softwareag.com. You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

Contacting customer support

If you have an account, you may open Apama Support Incidents online via the eService section of Empower at https://empower.softwareag.com/. If you do not yet have an account, send an email to empower@softwareag.com with your name, company, and company email address and request an account.

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at https://empower.softwareag.com/public_directory.asp and give us a call.

1 Release Notes

- What's new in 10.1 Release 8
- What's new in 10.0 Release 10
- What's new in 9.12 Release 10

Release Notes describes the changes introduced with the current Apama Connectivity For Cumulocity IoT release.

What's new in 10.1 Release

- Product name changed to be Apama Connectivity For Cumulocity IoT.

  The previous product name Apama Connectivity with Device Integration Platform has been renamed to Apama Connectivity For Cumulocity IoT. This also means that, the filenames and configurations in the product have been updated. The changes in the filenames and configurations are listed below. For more information, see "Migration from earlier versions to 10.1" on page 12.

  - EPL files

Old name	New name
DIP_EventDefinitions.mon	C8Y_EventDefinitions.mon
DIP_ServiceMonitor.mon	C8Y_ServiceMonitor.mon

  - Transport name and transport class

Old name	New name
DIPTransport.jar	CumulocityTransport.jar
com.apama.dip.Transport	com.apama.cumulocity.Transport

  - Events. The package name has been renamed from `com.apama.dip` to `com.apama.cumulocity`. For example:

Old name	New name
com.apama.dip.Device	com.apama.cumulocity.Device
com.apama.dip.Event	com.apama.cumulocity.Event

  - Bundle files

Old name	New name
DIP Event Definitions.bnd	C8Y Event Definitions.bnd

Old name	New name
DIP Transport.bnd	C8Y Transport.bnd

- Connectivity configuration files

Old name	New name
DeviceIntegrationPlatform.yaml	Cumulocity.yaml
DeviceIntegrationPlatform-credentials.properties	Cumulocity-credentials.properties

The `chainName` is renamed from `DeviceintegrationPlatform` to `CumulocityIoT`. That is, any requests that are sent to Cumulocity should now be sent to `CumulocityIoT` channel instead of `DeviceintegrationPlatform` channel. For example, `send DeviceOperation(...) to 'CumulocityIoT'`.

The properties in the Cumulocity-credentials.properties file are also renamed.

Old name	New name
DIP_TENANT	C8Y_TENANT
DIP_USERNAME	C8Y_USERNAME
DIP_PASSWORD	C8Y_PASSWORD
DIP_APPKEY	C8Y_APPKEY

- Removed support for Integration with Industry Analytics Kit.

  The **Apama Connectivity with Device Integration Platform to Transformer Bridge** bundle (DIP Transformers Bridge.bnd) that is used to convert Cumulocity data to/from Industry Analytics data is no longer part of the product. It is available as a separate solution component. You can download this solution component from Software AG TECHcommunity website at http://techcommunity.softwareag.com.

  The following service monitors are no longer available with the installation and must removed from the service monitor injection order:

  - DIP_DisplayOperation.mon
  - DIP_DisplayOperationService.mon
  - DIP_IoTTransformersBridge.mon

What's new in 10.0 Release

- The Designer variable *IOT_ANALYTICS_HOME* for defining the installation directory of the IoT Analytics kit has been replaced with *INDUSTRY_IOT_KIT_HOME* .

What's new in 9.12 Release

This is the first release of Apama Connectivity with Device Integration Platform.

2 Installing Apama Connectivity For Cumulocity IoT

- Installation 12
- Migration from earlier versions to 10.1 12
- Configuration Parameters 17
- Monitor Injection Order 18

Installation

- Download the `ApamaConnectivityForCumulocityIoT_version_ALL.zip` from Software AG Empower Product Support website. See "Online Information" on page 5.
- Extract the Apama Connectivity For Cumulocity IoT zip file to ${APAMA_HOME}/connectivity. The installation is successful, if you can find the `ApamaConnectivityForCumulocityIoT-version.txt` file at ${APAMA_HOME}/connectivity.
- All the monitors, libraries, documents, bundles, samples are placed in the respective folders.
  - `bundles` folder includes
    - The `Cumulocity` folder which includes `Cumulocity.yaml` and `Cumulocity-credentials.properties`
    - `C8Y Event Definitions.bnd`
    - `C8Y Transport.bnd`
  - `doc` folder includes
    - `Cumulocity` folder which includes ApamaDoc
  - `lib` folder includes
    - `CumulocityTransport.jar`
    - `Cumulocity` folder which includes all dependant third party libraries
  - `monitors` folder includes
    - `C8Y_EventDefinitions.mon`
    - `C8Y_ServiceMonitor.mon`
  - `Samples` folder includes `Cumulocity` folder which includes sample projects
  - `ApamaConnectivityForCumulocityIoT-version.txt`

Migration from earlier versions to 10.1

If you are migrating from a previous version of Apama Connectivity For Cumulocity IoT (previously, Apama Connectivity with Device Integration Platform), you must update the filenames and configurations to work with the existing applications. Follow these steps to migrate from previous versions:

1. Backup the properties DIP_TENANT, DIP_USERNAME, DIP_PASSWORD, DIP_APPKEY in the the file DeviceIntegrationPlatform-credentials.properties.

2. Remove any dependency on the `Apama Connectivity With Device Integration Platform` bundle along with any bundle instance files. This is no longer available and must be removed completely from the project.
3. Add the new Connectivity adapter bundle Apama Connectivity For Cumulocity IoT to your project.
4. Copy the values of the properties DIP_TENANT, DIP_USERNAME, DIP_PASSWORD, DIP_APPKEY to C8Y_TENANT, C8Y_USERNAME, C8Y_PASSWORD, C8Y_APPKEY respectively in the Cumulocity-credentials.properties file.
5. Rename instances of the `com.apama.dip` package to `com.apama.cumulocity` package. This would impact the following events:

Old name	New name
com.apama.dip.Error	com.apama.cumulocity.Error
com.apama.dip.DeviceOperation	com.apama.cumulocity.DeviceOperation
com.apama.dip. DeviceOperationStringArray	com.apama.cumulocity.DeviceOperation
com.apama.dip.DeviceOperationJson	com.apama.cumulocity.DeviceOperation
com.apama.dip.RequestAllDevices	com.apama.cumulocity.RequestAllDevices
com.apama.dip. RequestAllDevicesComplete	com.apama.cumulocity. RequestAllDevicesComplete
com.apama.dip. SubscribeMeasurements	com.apama.cumulocity. SubscribeMeasurements
com.apama.dip. UnsubscribeMeasurements	com.apama.cumulocity. UnsubscribeMeasurements
com.apama.dip.MeasurementValue	com.apama.cumulocity.MeasurementValue
com.apama.dip.Measurement	com.apama.cumulocity.Measurement
com.apama.dip.Event	com.apama.cumulocity.Event
com.apama.dip.Device	com.apama.cumulocity.Device

Old name	New name
com.apama.dip.FindAlarm	com.apama.cumulocity.FindAlarm
com.apama.dip.FindAlarmResponse	com.apama.cumulocity.FindAlarmResponse
com.apama.dip. FindAlarmResponseAck	com.apama.cumulocity. FindAlarmResponseAck
com.apama.dip.FindManagedObject	com.apama.cumulocity.FindManagedObject
com.apama.dip. FindManagedObjectResponse	com.apama.cumulocity. FindManagedObjectResponse
com.apama.dip.SendEmail	com.apama.cumulocity.SendEmail
com.apama.dip.SendSMS	com.apama.cumulocity.SendSMS
com.apama.dip. SMSResourceReference	com.apama.cumulocity. SMSResourceReference
com.apama.dip.SMSResponse	com.apama.cumulocity.SMSResponse
com.apama.dip.SendSpeech	com.apama.cumulocity.SendSpeech

6. Rename the channel name `DeviceIntegrationPlatform` to `CumulocityIoT`.

With the adoption of correlator `any` type support to Apama Connectivity For Cumulocity IoT, the following event definitions have been updated:

Event

The application code must be updated accordingly for extracting values from `params` dictionary. For more information on `any` type, see Apama documentation. In the following event definition, the code phrase `dictionary <string, string> params` has been updated to `dictionary <string, any> params`.

```
@ExtraFieldsDict("params")
event Event {
    /** Only used for internal purposes. Should not be used
        by application developer. */
    constant string CHANNEL :="CumulocityIoT";
    /** Type of the Event or Alarm. */
    string eventType;
    /** assetId of the source of the event. */
    string assetId;
    /** Timestamp of the event. */
    float timestamp;
    /** Text or message of the Event or Alarm */
    string text;
```

```
    /** Any extra parameters available on the Event or Alarm. */
    dictionary<string, any> params;
}
```

Device

The application code must be updated accordingly for extracting values from `params` dictionary. For more information on `any` type, see Apama documentation. In the following event definition, the code phrase `dictionary <string, string> params` has been updated to `dictionary <string, any> params`.

```
@ExtraFieldsDict("params")
event Device {
    /** The channel to which measurements are sent from the transport.*/
    constant string CHANNEL:= "cumulocity.devices";
    /** The unique identifier for this specific Device. */
    string assetId;
    /** The name of this Device. <b>Note:</b> This does not
        have to be a unique value. */
    string name;
    /** The device integration platform type of the device */
    string type;
    /** The description for this Asset. */
    string description;
    /** List of supported operations for this device. */
    sequence<string> supportedOperations;
    /** List of supported measurements for this device */
    sequence<string> supportedMeasurements;
    /** Child assetIds of child devices */
    sequence<string> childAssetIds;
    /** If knonwn, contains lat, lng, altitude and accuracy */
    dictionary<string, decimal> position;
    /** A set of stringified extra configuration information
        for this Device. */
    dictionary<string, any> params;
}
```

Measurement or MeasurementValue

The application code must be updated accordingly for extracting values from `extraParams/params` dictionary. For more information on `any` type, see Apama documentation. In the following event definition, the code phrase `dictionary <string, string> extraParams` has been updated to `dictionary <string, any> extraParams`.

```
@ExtraFieldsDict("extraParams")
event MeasurementValue {
    /** Value from the sensor */
    decimal value;
    /** Units the reading is in, e.g. mm, lux */
    string unit;
    /** Type of the measurement - COUNTER, BOOLEAN, GUAGE or RATE */
    string type;
    /** Quantity */
    string quantity;
    /** State of the measurement - ORIGINAL, INTERPOLATED, VALIDATED */
    string state;
    /** Any per-value extra parameters */
    dictionary<string, any> extraParams;
}
```

In the following event definition, the code phrase `dictionary <string, string> params` has been updated to `dictionary <string, any> params`.

```
@ExtraFieldsDict("params")
event Measurement {
    /** The channel to which measurements are sent from the transport.
     */
    constant string CHANNEL:= "cumulocity.measurements";
    /** The channel to send a measurements to create a new measurement
     */
    constant string CREATE_CHANNEL:= "CumulocityIoT";
    /** The type of the measurement. */
    string measurementType;
    /** The unique identifier of the source of the measurement.
     *  This should correspond to an Device sent in previously. */
    string assetId;
    /** The timestamp the measurement was taken, represented as the number of
     * seconds since the epoch (1st January 1970). */
    float timestamp;
    /** Value name to MeasurementValue map of measurements. */
    dictionary<string, MeasurementValue> measurements;
    /** This stringified dictionary is available to hold any other data
     *  associated with the measurement. */
    dictionary<string, any> params;
}
```

Device operation

The previous versions of `DeviceOperationStringArray` and `DeviceOperationJson` are now supported using the base `DeviceOperation` event object. See "Sending device operations" on page 16 section.

```
@ExtraFieldsDict("params")
event DeviceOperation {
    /** Channel to send the events to. */
    constant string CHANNEL := "CumulocityIoT";
    /** assetId of the Device to send the operation to. */
    string assetId;
    /** Operation-specific data. */
    dictionary<string, any> operations;
    /** Any extra parameters available on the Device Operation. */
    dictionary<string, any> params;
}
```

Sending device operations

You can send device operations using the `DeviceOperation` event.

```
DeviceOperation do := new DeviceOperation;
do.assetId := "<DEVICE_ID>"; // This needs to be populated with
 the appropriate device/asset Id
//Display a sample message if the device supports a display
//This makes supporting the usecases similar to the earlier versions
  of both DeviceOperation and DeviceOperationJson far more simpler
do.operations.add("c8y_Meassage", {"text":"Hello Cumulocity device"});
//Set the states of multiple Relay Devices attached to a device
//This makes supporting the usecases similar to the earlier versions of
  both DeviceOperationStringArray far more simpler
do.operations.add("c8y_RelayArray", ["CLOSED", "OPEN", "OPEN", "CLOSED"]);
send do to DeviceOperation.CHANNEL;
```

Examples

- **DeviceOperation**

  Previous: `DeviceOperation("12345","c8y_Meassage",{"text":"Hello Cumulocity device"})`

  New: `DeviceOperation("12345", {"c8y_Meassage":<any>{<any>"text":<any>"Hello Cumulocity device"}},new dictionary<string, any>)`

- **DeviceOperationStringArray**

  Previous: `DeviceOperationStringArray("12345","c8y_RelayArray", ["CLOSED","OPEN","CLOSED","OPEN"])`

  New: `DeviceOperation("12345", {"c8y_RelayArray":<any>["CLOSED","OPEN","CLOSED","OPEN"]},new dictionary<string, any>)`

- **DeviceOperationJson**

  Previous: `DeviceOperationJson("12345",{"c8y_Meassage":"{\"text\": \"Hello cumulocity\"}","c8y_Relay":"{\"relayState\":\"CLOSED\"}"})`

  New: `DeviceOperation("12345", {"c8y_Meassage":<any>{<any>"text":<any>"Hello cumulocity"},"c8y_Relay":<any>{<any>"relayState":<any>"CLOSED"}},new dictionary<string, any>)`

Configuration Parameters

Provide the following parameters in a yaml configuration file. Refer to `Cumulocity.yaml`.

- `requestAllDevices`. Requests for all assets at startup. Default value is true.
- `subscribeToAllMeasurements`. Subscribes for measurements of all devices during startup. Default value is true.
- `subscribeToDevices`. Subscribes for all device related updates. Default value is true.
- `tenant`. Unique name given to the application tenant.
- `username`. Name of the user.
- `password`. Password of the user.
- `appKey`. Unique key for the application defined on the Cumulocity IoT instance. For more information, see "Creating an Application Key" on page 20.
- `url`. Optionally provide an URL if connecting to an on-premise installation of Cumulocity IoT.

Monitor Injection Order

To support the Apama Connectivity For Cumulocity IoT, you must inject the following monitor files into the correlator before injecting your application monitors:

- `connectivity/monitors/C8Y_EventDefinitions.mon`
- `connectivity/monitors/C8Y_ServiceMonitor.mon`

As with other Connectivity plugins, the application should call `com.softwareag.connectivity.ConnectivityPlugins.onApplicationInitialized()`. For more information, see "Working with Connectivity Plug-ins" section in Apama documentation.

3 Working with Apama Connectivity For Cumulocity IoT

- Introduction 20
- Creating an Application Key 20
- Developing with Software AG Designer 20
- Manual Deployment 20

Introduction

The Apama Connectivity For Cumulocity IoT allows you to receive information from devices, measurements, events and alarms from Cumulocity IoT and send operations to Cumulocity IoT (and thus to devices).

Creating an Application Key

To create an application key

1. Create an account with Cumulocity IoT. For more information, see Cumulocity IoT User's Guide.
2. Log in to Cumulocity IoT account, and go to the Administration
3. Select **Applications > Own Applications**. Click **Create Application**.
4. Provide a valid tenant name, user name, password and application key.
5. Select application type as **External application**.
6. Click **Save**.

Developing with Software AG Designer

To add Apama Connectivity For Cumulocity IoT to your project

1. Add **Apama Connectivity for Cumulocity IoT** bundle.
2. In the **Project Explorer** view, select the project and expand **Connectivity and Adapters -> Apama Connectivity for Cumulocity IoT**.
3. Edit `Cumulocity-credentials.properties` file. You must replace the ${C8Y_...} parts with the username, password, tenant, and application key.
4. Start the application. See "Application Startup" on page 24.

Manual Deployment

1. Customize the `C8Y_*` values in the `Cumulocity-credentials.properties` file with the correct tenant.
   - C8Y_USERNAME=<username>
   - C8Y_TENANT=<tenant>
   - C8Y_PASSWORD=<password>

- C8Y_APPKEY=<application key created above>

2. Add `--config Cumulocity-credentials.properties --config Cumulocity.yaml` to the correlator command line.
3. Inject the following monitors in the specified order:
   - `connectivity/monitors/C8Y_EventDefinitions.mon`
   - `connectivity/monitors/C8Y_ServiceMonitor.mon`
4. Start the application. See "Application Startup" on page 24.

4 Interacting with Cumulocity IoT

- Application Startup 24
- Device Events 24
- Measurement Events 24
- Events and Alarms 25
- Sending control events to Cumulocity IoT 25
- Sample EPL 26

Refer to the included ApamaDoc installed at Apama/connectivity/doc/Cumulocity.

Application Startup

As with other Connectivity plugins, the application should call `com.softwareag.connectivity.ConnectivityPlugins.onApplicationInitialized()`. For more information, see "Working with Connectivity Plug-ins" section in Apama documentation.

Device Events

The `com.apama.cumulocity.Device` events are sent to the `com.apama.cumulocity.Device.CHANNEL` that is `cumulocity.devices` when `onApplicationInitialized()` is called (unless `requestAllDevices` is set to false in the `Cumulocity.yaml` file). After all assets are sent, the `com.apama.cumulocity.RequestAllDevicesComplete(-1)` event is sent.

Example of a device event:

```
com.apama.cumulocity.Device("43026768","Mbed Test Device",
  "com_ublox_C027_REV-A","",["c8y_Relay","c8y_Configuration",
  "c8y_Message"],[],[],{"alt":610.2,"lat":17.426479,"lng":78.33123},
  {,"attrs.c8y_Availability":any(dictionary<any,any>,
  {any(string,"status"):any(string,"UNAVAILABLE")}),
  "attrs.c8y_Hardware":any(dictionary<any,any>,
  {any(string,"model"):any(string,"Ublox C027"),
  any(string,"revision"):any(string,"1"),
  any(string,"serialNumber"):any(string,"352648069564516")}),
  "attrs.c8y_IsDevice":any(dictionary<any,any>,{}),
  "attrs.c8y_Mobile":any(dictionary<any,any>,
  {any(string,"cellId"):any(string,"14D80CD"),
  any(string,"iccid"):any(string,"899149059000016774658"),
  any(string,"imei"):any(string,"352648069564516")}),
  "attrs.c8y_MotionTracking":any(dictionary<any,any>,
  {any(string,"active"):any(boolean,true),
  any(string,"interval"):any(integer,0)}),
  "attrs.c8y_RequiredAvailability":any(dictionary<any,any>,
  {any(string,"responseInterval"):any(integer,20)}),
  "attrs.com_cumulocity_model_Agent":any(dictionary<any,any>,{}),
  "owner":any(string,"device_352648069564516")})
```

Any device added after `applicationInitialized` is called is sent to the default channel (unless `subscribeToDevices` is set to false in the `Cumulocity.yaml` file).

Measurement Events

The `com.apama.cumulocity.Measurement` events are sent to `com.apama.cumulocity.Measurement.CHANNEL` that is `cumulocity.measurements` when `onApplicationInitialized()` is called (unless `subscribeToAllMeasures` is set to false in the `Cumulocity.yaml` file). For example, `c8y_LightMeasurement`

or `c8y_DistanceMeasurement`. These events may be sent before all assets are sent. Measurement events contain the assetID of the source of the measurement, the type of measurement, timestamp, and a dictionary of values which contain the numeric value, units and optional type, quantity and state.

Examples of measurement events:

```
com.apama.cumulocity.Measurement("c8y_LightMeasurement","12346081",1464359004.89,
{"e":com.apama.cumulocity.MeasurementValue(108.1,"lux","","","",{})},{})
com.apama.cumulocity.Measurement("c8y_DistanceMeasurement","12346082",1464359005.396,
{"distance":com.apama.cumulocity.MeasurementValue(344,"mm","","","",{})},{})
```

Events and Alarms

The `com.apama.cumulocity.Event` event is sent on any event or alarm from a device. This event is sent to `com.apama.cumulocity.Measurement.CHANNEL` that is `cumulocity.measurements` channel. This event contains the assetID of the source, a timestamp (same form as `currentTime`), message text, and optional parameters.

Examples of events and alarms:

```
com.apama.cumulocity.Event("c8y_EntranceEvent","12346082",1464364483.396,
  "Entrance event triggered.",{"distance":any(float,317)})
com.apama.cumulocity.Event("c8y_UnavailabilityAlarm","12669915",1464365565.661,
  "No communication with device since
  2016-05-27T18:11:23.886+02:00",{"count":any(integer,1),
  "severity":any(string,"MAJOR")})
```

Note: See the Apamadoc included with Apama Connectivity For Cumulocity IoT.

Sending control events to Cumulocity IoT

The Apama Connectivity For Cumulocity IoT listens on the `CumulocityIoT` channel for most requests. For all events sent to the transport, a `CHANNEL` constant is defined on the event type with the correct channel to send the event to.

See the `com.apama.cumulocity.RequestAllDevices` event for refreshing the list of devices. Note that the `com.apama.cumulocity.RequestAllDevices` event is sent to the channel defined by the CHANNEL constant on `RequestAllDevices` event. The device events are sent to the channel specified in the `RequestAllDevices` event and followed by a `com.apama.cumulocity.RequestAllDevicesComplete` event with the `requestId` sent in the `RequestAllDevices` event.

The `com.apama.cumulocity.SubscribeMeasurements` and `com.apama.cumulocity.UnsubscribeMeasurements` events control subscriptions to measurements from assets. Specify the assetId as either an assetId from a device event or as `"*"` for a wildcard (use the `WILDCARD` constant). Subscriptions are reference counted by the transport, so send as many Unsubscribe events as Subscribe events have been sent to completely unsubscribe. Unless the `subscribeToMeasurements` configuration property is false, the transport automatically subscribes as if

`SubscribeMeasurements("*")` had been sent at `applicationInitialized` time. Subscriptions to measurements also subscribe to events and alarms.

Send a `com.apama.cumulocity.DeviceOperation` event to send an operation to a device. For example, to set a device's display and modify a set of relays, send:

```
com.apama.cumuclocity.DeviceOperation("12345",{"c8y_Meassage":<any>
  {<any>"text":<any>"Hello Cumulocity device"}},
  new dictionary<string, any>)
com.apama.cumuclocity.DeviceOperation("12345",{"c8y_RelayArray":<any>
  ["CLOSED","OPEN","CLOSED","OPEN"]},
  new dictionary<string, any>)
com.apama.cumuclocity.DeviceOperation("12345",{"c8y_Meassage":<any>
  {<any>"text":<any>"Hello cumulocity"},
  "c8y_Relay":<any>{<any>"relayState":<any>"CLOSED"}},
  new dictionary<string, any>)
```

Sample EPL

Below sample EPL describes how to subscribe and receive device measurements, device events and device information.

```
package com.apama.cumulocity.sample;
monitor SampleApplication {
 com.apama.cumulocity.Measurement m;
 com.apama.cumulocity.Event e;
 com.apama.cumulocity.Device d;
 action onload() {
  // Subscribe to receive all the devices from Cumulocity IoT
  monitor.subscribe(com.apama.cumulocity.Device.CHANNEL);
  // Consume all the devices from Cumulocity IoT
  on all com.apama.cumulocity.Device() as d {
   log d.toString() at INFO;
  }
  // Subscribe to receive all the measurement published from
  // Cumulocity IoT
  monitor.subscribe(com.apama.cumulocity.Measurement.CHANNEL);
  // Consume all the measurements from Cumulocity IoT
  on all com.apama.cumulocity.Measurement() as m {
   log  m.toString() at INFO;
  }
  // Consume all the events from Cumulocity IoT
  on all com.apama.cumulocity.Event() as e {
   log e.toString() at INFO;
  }
 }
}
```