

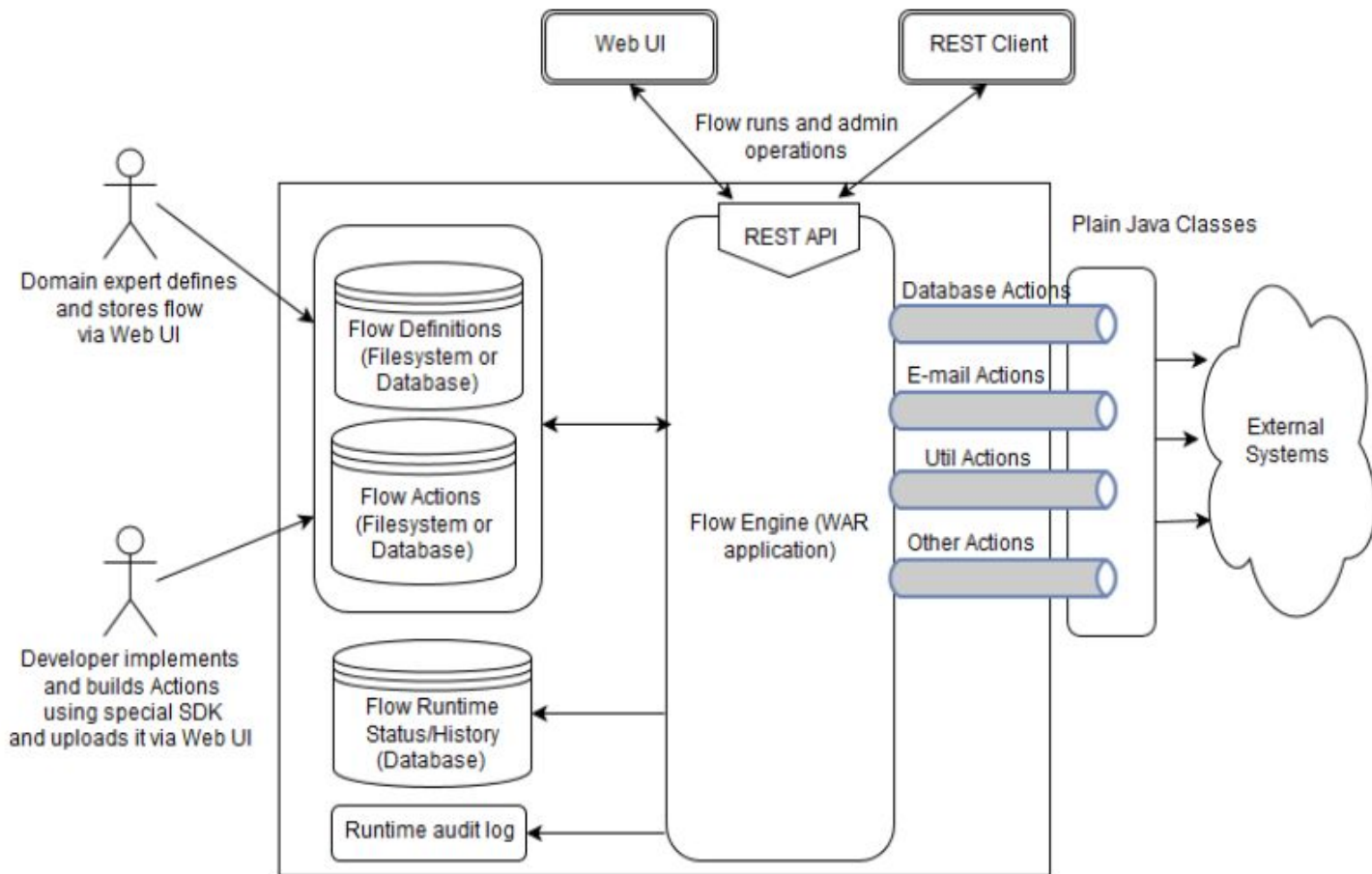
# Business Flows Automation

Highlights of current project state

# Key Items

- The flows automation framework includes runtime engine, Web UI and scripts parsing/compilation unit
- Allows domain experts creating human-readable scripts that define automation flows
- Enables a simple development/deploy of custom actions that are building blocks of automation flows
- Provides graphical view of automation flow scripts
- Supports on-the-fly flow updates. No application or server restart is needed after updating flow or its building blocks.
- Exposes REST API for flows runtime and administration operations (besides Web UI).
- The architecture assumes simple vertical scalability
- Can be used, for example, by Telecom operators for describing/implementing of some of their automation flows. The examples are Service provisioning in OSS (some parts of the whole flow), rules for Value-Added Services.

# High-Level Architecture



# Flow Syntax

Scripts

- Custom Group 003
- Custom Group 01
- Custom Group 02
- Default Group
- Demo Flows
  - Process MSISDN
  - Process MSISDN 2
  - Process MSISDN Invoker

Process MSISDN x MSISDN x Write EDR x

Source Flow Chart

```
1 DeclareInputVar("msisdn", "String");
2
3 DeclareLocalVar("msisdnRec", "MSISDN");
4
5 Action("MSISDN::Get", "msisdn").SetLocalVar("msisdnRec");
6
7 if (Equal("msisdnRec", "null", "MSISDN missed")) {
8     Action("MSISDN::Create", "msisdn").SetLocalVar("msisdnRec");
9 }
10 else {
11     Action("MSISDN::Update Count", "msisdn");
12     SetLocalVar("msisdnRec.flowCounter", "msisdnRec.flowCounter + 1");
13 }
14
15 Action("Util::Send Email",
16     "'BFA Notification: MSISDN Processed'",
17     "'MSISDN ' + msisdnRec.value + ' was processed!'");
18
19 Action("Util::Write EDR", "msisdnRec.value", "'some.email555@gmail.com'");
20
21
```

Input parameters

Local variables

Custom actions

Expressions

# Sub-Flows

The screenshot displays a software interface with two main panels. On the left is the 'Scripts' panel, which contains a tree view of folders and scripts. The folders are 'Custom Group 003', 'Custom Group 01', 'Custom Group 02', 'Default Group', and 'Demo Flows'. Under 'Demo Flows', there are three scripts: 'Process MSISDN', 'Process MSISDN 2', and 'Process MSISDN Invoker', which is currently selected. On the right is the code editor, which has a tab bar at the top with four tabs: 'Process MSISDN', 'MSISDN', 'Write EDR', and 'Process MSISDN Invoker'. The 'Process MSISDN Invoker' tab is active. Below the tab bar is a toolbar with icons for saving, running, and switching between 'Source' (selected) and 'Flow Chart' views. The code editor shows a single line of code: `SubFlow("Demo Flows::Process MSISDN", "'+1 866 555 555 555'");`. The second argument, `"'+1 866 555 555 555'"`, is highlighted with a red underline. A red arrow points from a box labeled 'Input parameter' to this highlighted string.

Scripts

- Custom Group 003
- Custom Group 01
- Custom Group 02
- Default Group
- Demo Flows
  - Process MSISDN
  - Process MSISDN 2
  - Process MSISDN Invoker

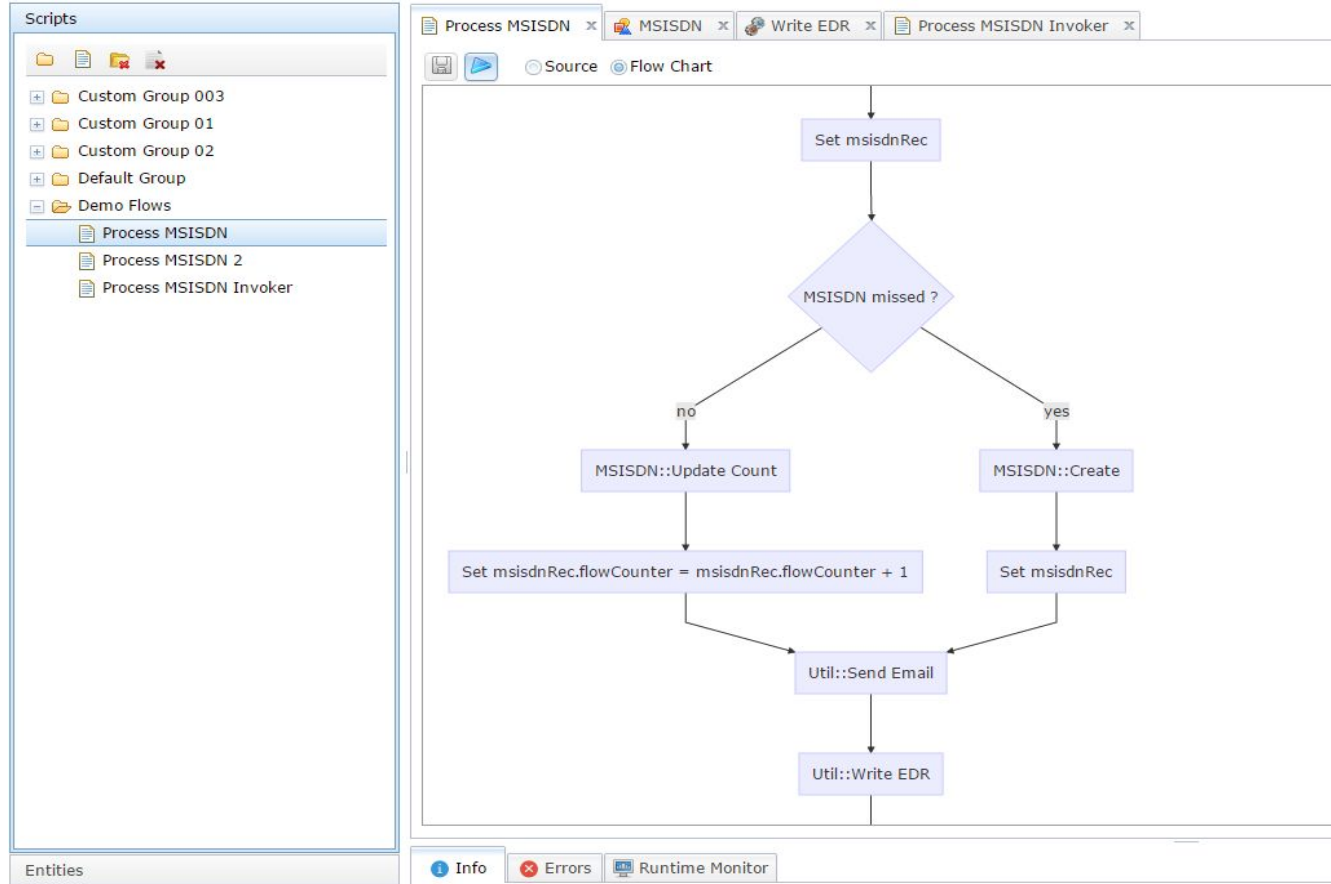
Process MSISDN x MSISDN x Write EDR x Process MSISDN Invoker x

Source Flow Chart

```
1 SubFlow("Demo Flows::Process MSISDN", "'+1 866 555 555 555'");
2
```

Input parameter

# Flow Chart View



# Custom Data Types

The screenshot displays a software development environment with two main windows. The top window shows the 'Entities' list on the left and a table of field definitions for the 'MSISDN' entity. The bottom window shows a script editor with a code snippet that uses the 'MSISDN' custom data type.

**Entities List (Left Panel):**

- Default Group
  - AccessType
  - Contract
  - DefaultConfiguration
  - MSISDN**
  - PrepaidSubscriber
  - ReserveAmountError
  - SubscriberTariffCode
  - Subscriber
- E1

**Field Definitions Table (Top Right):**

Field Name	Field Type
id	Number
value	String
creationDate	String
flowCounter	Number

**Script Editor (Bottom Right):**

Process MSISDN x MSISDN x Write EDR x Process MSISDN Invoker x

Source Flow Chart

```
1 DeclareInputVar("msisdn", "String");
2
3 DeclareLocalVar("msisdnRec", "MSISDN");
4
5 Action("MSISDN::Get", "msisdn").SetLocalVar("msisdnRec");
6
7 if (Equal("msisdnRec", "null", "MSISDN missed")) {
8     Action("MSISDN::Create", "msisdn").SetLocalVar("msisdnRec");
9 }
10 else {
11     Action("MSISDN::Update Count", "msisdn");
12     SetLocalVar("msisdnRec.flowCounter", "msisdnRec.flowCounter + 1");
13 }
14
15 Action("Util::Send Email",
16     "'BFA Notification: MSISDN Processed'",
17     "'MSISDN ' + msisdnRec.value + ' was processed!'");
18
19 Action("Util::Write EDR", "msisdnRec.value", "'some.email555@gmail.com'");
20
21
```

**Usage of custom data types:**

Red arrows point from the text box to the following code elements:

- Line 3: `DeclareLocalVar("msisdnRec", "MSISDN");`
- Line 12: `SetLocalVar("msisdnRec.flowCounter", "msisdnRec.flowCounter + 1");`
- Line 17: `"'MSISDN ' + msisdnRec.value + ' was processed!'"`
- Line 19: `"msisdnRec.value"`

# Custom Actions with Local Dependencies

The screenshot displays a software development environment with a project structure on the left and a configuration panel on the right.

**Left Panel (Project Structure):**

- Scripts
- Entities
- Actions
  - [Folder Icon] [Globe Icon] [Folder Icon] [X Icon]
  - [+] [Folder Icon] Billing
  - [-] [Folder Icon] Database
    - [Globe Icon] Close Connection
  - [+] [Folder Icon] Default Group
  - [-] [Folder Icon] MSISDN
    - [Globe Icon] Create
    - [Globe Icon] Get
    - [Globe Icon] Update Count
  - [+] [Folder Icon] Std
  - [+] [Folder Icon] Subscriber
  - [+] [Folder Icon] SubscriberTariffCode
  - [-] [Folder Icon] Util
    - [Globe Icon] Format MSISDN
    - [Globe Icon] Send Email
    - [Globe Icon] Write EDR (Selected)

**Right Panel (Configuration):**

Process MSISDN x MSISDN x Create x Write EDR x Process

[Refresh Icon]

Implementation class: `com.bfa.demo.action.util.WriteEDR`

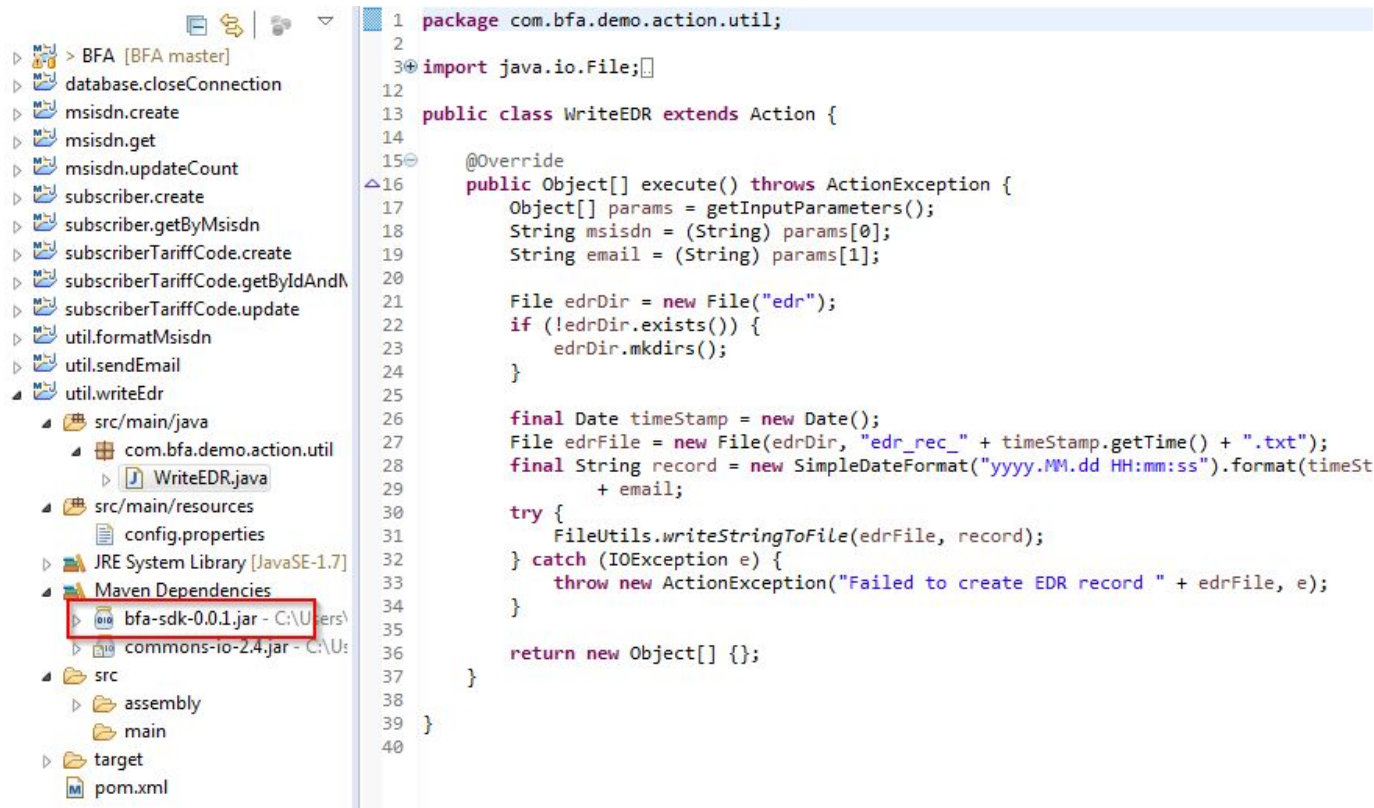
**Dependencies**

`commons-io-2.4.jar`



# Actions SDK

Action template project can be created with SDK generator



The screenshot displays an IDE interface. On the left, the project structure is visible, showing a folder named 'BFA [BFA master]' containing various sub-projects and a 'Maven Dependencies' folder. The 'bfa-sdk-0.0.1.jar' file is highlighted in the 'Maven Dependencies' folder. On the right, the code editor shows the contents of 'WriteEDR.java'.

```
1 package com.bfa.demo.action.util;
2
3 import java.io.File;
4
5 public class WriteEDR extends Action {
6
7     @Override
8     public Object[] execute() throws ActionException {
9         Object[] params = getInputParameters();
10        String msisdn = (String) params[0];
11        String email = (String) params[1];
12
13        File edrDir = new File("edr");
14        if (!edrDir.exists()) {
15            edrDir.mkdirs();
16        }
17
18        final Date timeStamp = new Date();
19        File edrFile = new File(edrDir, "edr_rec_" + timeStamp.getTime() + ".txt");
20        final String record = new SimpleDateFormat("yyyy.MM.dd HH:mm:ss").format(timeSt
21            + email;
22
23        try {
24            FileUtils.writeStringToFile(edrFile, record);
25        } catch (IOException e) {
26            throw new ActionException("Failed to create EDR record " + edrFile, e);
27        }
28
29        return new Object[] { };
30    }
31}
```

# Actions and Custom Data Types

Actions can use custom data types defined in flows

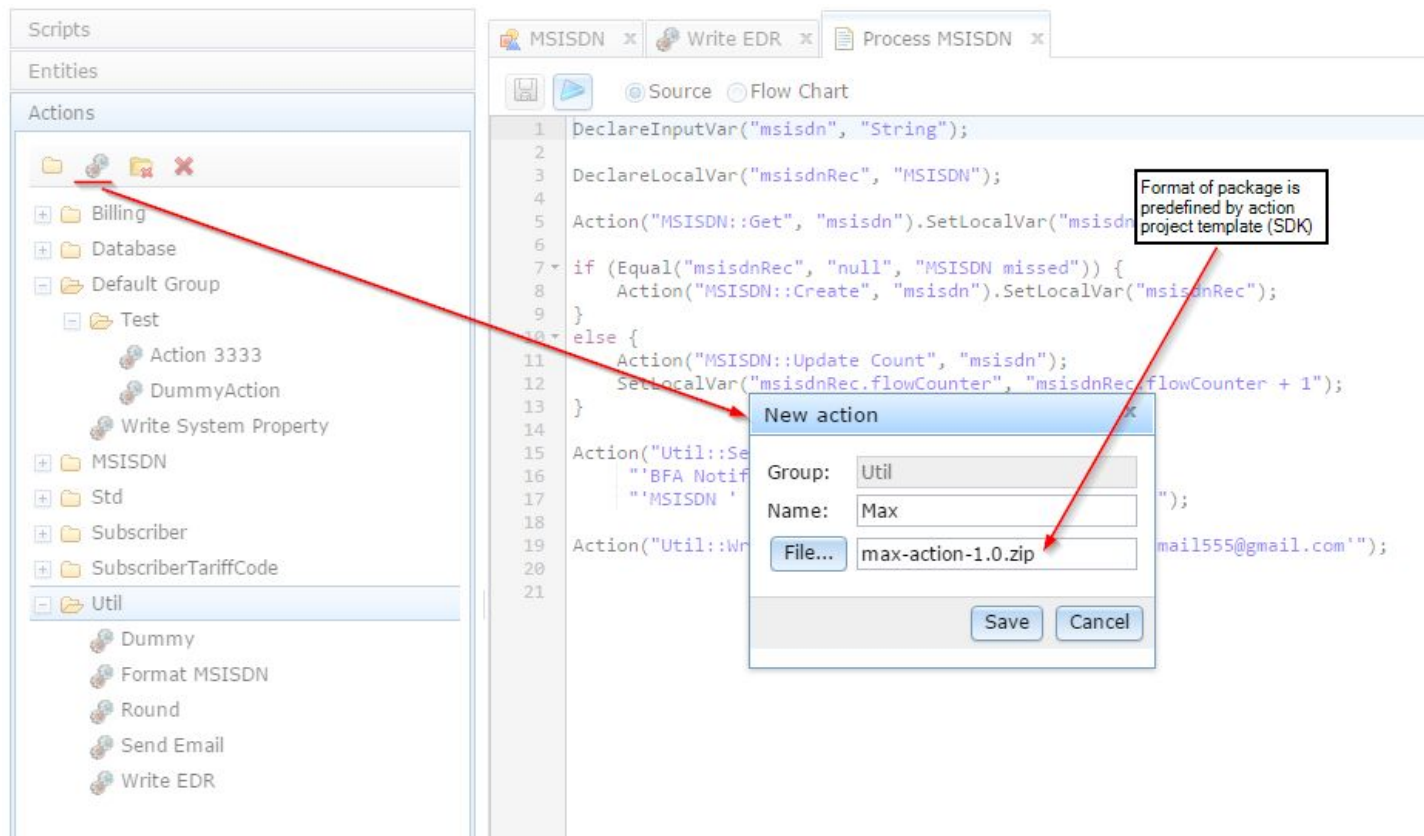
The screenshot illustrates the process of using a custom data type in a Java action. It is divided into three main sections:

- Left Panel (Project Explorer):** Shows the project structure. Under 'Maven Dependencies', the file `bfa-entities-0.0.1.jar` is highlighted with a red box. A red arrow points from this file to the `import` statement in the code.
- Center Panel (Code Editor):** Displays the `DaoHelper` class. The `import java.sql.CallableStatement;` line is highlighted with a blue box. A red arrow points from this line to the `MSISDN` parameter in the `create` method signature. Another red arrow points from the `MSISDN` parameter to the `new MSISDN()` instantiation in the method body. A text box labeled 'Custom data type exported from repository' points to the `import` statement.
- Right Panel (Scripts/Entities):** Shows the 'Entities' section of the IDE. A 'Download Library button' (represented by a download icon) is highlighted with a red box and a red arrow. Below it, the 'Default Group' contains several entities, including `MSISDN`. Below this panel, a preview of the `MSISDN` entity is shown as a table.

Field Name	Field Type
id	Number
value	String
creationDate	String
flowCounter	Number

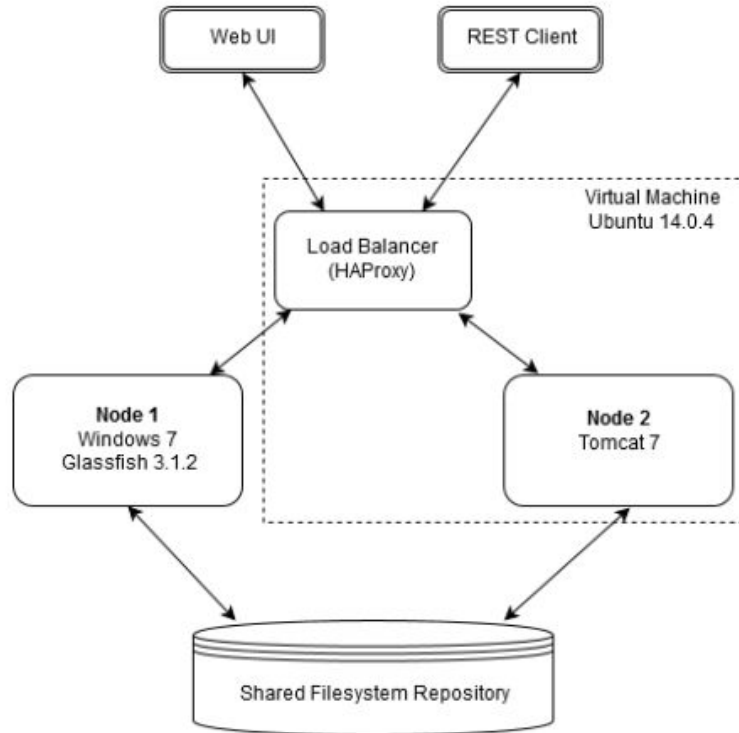
# Actions Deployment

Action package is built by SDK means (generated Maven pom.xml) and can be easily deployed



# Cluster Environment

Flow Engine can be used in clustered environment. The below configuration was tested.



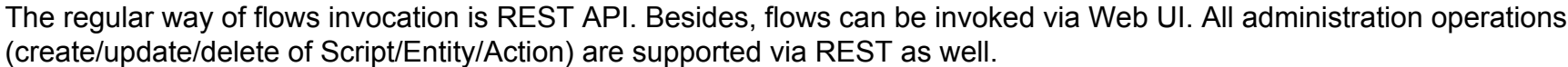
# On-the-Fly Updates

Script, custom data type or action can be updated on-the-fly. The changes are applied to all nodes in cluster. No server or application restart is needed

The screenshot displays a software interface for managing scripts. On the left, a 'Scripts' sidebar lists several groups: 'Custom Group 003', 'Custom Group 01', 'Custom Group 02', 'Default Group', and 'Demo Flows'. Under 'Demo Flows', three scripts are listed: 'Process MSISDN' (selected), 'Process MSISDN 2', and 'Process MSISDN Invoker'. The main area shows the 'Process MSISDN' script in 'Source' view. The script code includes variable declarations and conditional logic for handling 'msisdn' data. An 'Update script' dialog box is overlaid on the script, showing the 'Group' as 'Demo Flows' and the 'Name' as 'Process MSISDN'. The dialog has 'Update' and 'Cancel' buttons.

```
1 DeclareInputVar("msisdn", "String");
2
3 DeclareLocalVar("msisdnRec", "MSISDN");
4
5 Action("MSISDN::Get", "msisdn").SetLocalVar("msisdnRec");
6
7 if (Equal("msisdnRec", "null", "MSISDN missed")) {
8     Action("MSISDN::Create", "msisdn").SetLocalVar("msisdnRec");
9 }
10 else {
11     Action("MSISDN::", "msisdn").SetLocalVar("msisdnRec");
12     SetLocalVar("msisdnRec", "msisdnRec + msisdn + '\n' + msisdnCounter + 1");
13 }
14
15 Action("Util::Send E", "msisdnRec").SetLocalVar("msisdnRec");
16     "'BFA Notificat", "msisdnRec");
17     "'MSISDN ' + msisdnRec";
18
19 Action("Util::Write", "msisdnRec").SetLocalVar("msisdnRec");
20
21
```

The regular way of flows invocation is REST API. Besides, flows can be invoked via Web UI. All administration operations (create/update/delete of Script/Entity/Action) are supported via REST as well.



## Future Plans

- Script syntax enhancements
- DBMS based repository. Now, the repository is filesystem based.
- Flow runtime monitoring/logging/tracing. The basic functionality has been already implemented.
- Web UI enhancements (search, navigation etc.)
- Actions packaging/deployment enhancements (Action Group support)
- Security