Example with Elephant

Step 1: Create the connector

The connector establishes the connection with Elephant. We will put the authentication methods in this component and eventually some common methods.

The properties depends on the connector. There are usually credentials (username and password) to authenticate to the API.

```
package fr.btib.elephant.connector;

import com.google.gson.JsonObject;
import com.google.gson.JsonParser;
import fr.btib.connector.BExternalConnector;
import fr.btib.connector.element.connector.BExternalConnectorFolder;
import fr.btib.connector.model.BIModelConnector;
import fr.btib.core.BtibLogger;
import fr.btib.core.service.BBtibService;
import fr.btib.core.tool.CompTool;
import fr.btib.core.tool.HttpTool;

import javax.baja.nre.annotations.NiagaraAction;
import javax.baja.nre.annotations.NiagaraProperty;
import javax.baja.nre.annotations.NiagaraType;
import javax.baja.sys.*;
import java.net.URL;
import java.util.*;

@NiagaraType
@NiagaraProperty(
    name = "pingUrl",
    type = "String",
    defaultValue = ""
)
@NiagaraProperty(
    name = "tokenUrl",
    type = "String",
    defaultValue = ""
)
@NiagaraProperty(
    name = "clientId",
    type = "String",
    defaultValue = ""
)
@NiagaraProperty(
    name = "clientSecret",
    type = "String",
    defaultValue = ""
)
@NiagaraProperty(
    name = "username",
    type = "String",
    defaultValue = ""
)
@NiagaraProperty(
    name = "password",
    type = "String",
    defaultValue = ""
)
@NiagaraAction(
    name = "ping"
)
public class BElephantConnector extends BExternalConnector implements BIModelConnector
{
    /*+ ------------ BEGIN BAJA AUTO GENERATED CODE ------------ +*/
    /*@ $fr.btib.elephant.connector.BElephantConnector(1561721217)1.0$ @*/
```

```
/* Generated Wed Jan 29 10:47:25 CET 2020 by Slot-o-Matic (c) Tridium, Inc. 2012 */

////////////////////////////////////////////////////////////////
// Property "pingUrl"
////////////////////////////////////////////////////////////////

/**
 * Slot for the {@code pingUrl} property.
 *
 * @see #getPingUrl
 * @see #setPingUrl
 */
public static final Property pingUrl = newProperty(0, "", null);

/**
 * Get the {@code pingUrl} property.
 *
 * @see #pingUrl
 */
public String getPingUrl()
{
    return this.getString(pingUrl);
}

/**
 * Set the {@code pingUrl} property.
 *
 * @see #pingUrl
 */
public void setPingUrl(String v)
{
    this.setString(pingUrl, v, null);
}

////////////////////////////////////////////////////////////////
// Property "tokenUrl"
////////////////////////////////////////////////////////////////

/**
 * Slot for the {@code tokenUrl} property.
 *
 * @see #getTokenUrl
 * @see #setTokenUrl
 */
public static final Property tokenUrl = newProperty(0, "", null);

/**
 * Get the {@code tokenUrl} property.
 *
 * @see #tokenUrl
 */
public String getTokenUrl()
{
    return this.getString(tokenUrl);
}

/**
 * Set the {@code tokenUrl} property.
 *
 * @see #tokenUrl
 */
public void setTokenUrl(String v)
{
    this.setString(tokenUrl, v, null);
}

////////////////////////////////////////////////////////////////
// Property "clientId"
////////////////////////////////////////////////////////////////

/**
```

```
     * Slot for the {@code clientId} property.
     *
     * @see #getClientId
     * @see #setClientId
     */
    public static final Property clientId = newProperty(0, "", null);

    /**
     * Get the {@code clientId} property.
     *
     * @see #clientId
     */
    public String getClientId()
    {
        return this.getString(clientId);
    }

    /**
     * Set the {@code clientId} property.
     *
     * @see #clientId
     */
    public void setClientId(String v)
    {
        this.setString(clientId, v, null);
    }

    //////////////////////////////////////////////////////////////
    // Property "clientSecret"
    //////////////////////////////////////////////////////////////

    /**
     * Slot for the {@code clientSecret} property.
     *
     * @see #getClientSecret
     * @see #setClientSecret
     */
    public static final Property clientSecret = newProperty(0, "", null);

    /**
     * Get the {@code clientSecret} property.
     *
     * @see #clientSecret
     */
    public String getClientSecret()
    {
        return this.getString(clientSecret);
    }

    /**
     * Set the {@code clientSecret} property.
     *
     * @see #clientSecret
     */
    public void setClientSecret(String v)
    {
        this.setString(clientSecret, v, null);
    }

    //////////////////////////////////////////////////////////////
    // Property "username"
    //////////////////////////////////////////////////////////////

    /**
     * Slot for the {@code username} property.
     *
     * @see #getUsername
     * @see #setUsername
     */
    public static final Property username = newProperty(0, "", null);
```

```
/**
 * Get the {@code username} property.
 *
 * @see #username
 */
public String getUsername()
{
    return this.getString(username);
}

/**
 * Set the {@code username} property.
 *
 * @see #username
 */
public void setUsername(String v)
{
    this.setString(username, v, null);
}

////////////////////////////////////////////////////////////////
// Property "password"
////////////////////////////////////////////////////////////////

/**
 * Slot for the {@code password} property.
 *
 * @see #getPassword
 * @see #setPassword
 */
public static final Property password = newProperty(0, "", null);

/**
 * Get the {@code password} property.
 *
 * @see #password
 */
public String getPassword()
{
    return this.getString(password);
}

/**
 * Set the {@code password} property.
 *
 * @see #password
 */
public void setPassword(String v)
{
    this.setString(password, v, null);
}

////////////////////////////////////////////////////////////////
// Action "ping"
////////////////////////////////////////////////////////////////

/**
 * Slot for the {@code ping} action.
 *
 * @see #ping()
 */
public static final Action ping = newAction(0, null);

/**
 * Invoke the {@code ping} action.
 *
 * @see #ping
 */
public void ping()
{
    this.invoke(ping, null, null);
```

```
}

////////////////////////////////////////////////////////////////
// Type
////////////////////////////////////////////////////////////////

@Override
public Type getType()
{
    return TYPE;
}

public static final Type TYPE = Sys.loadType(BElephantConnector.class);

/*+ ------------ END BAJA AUTO GENERATED CODE -------------- +*/

public static final String CLIENT_ID = "client_id";
public static final String CLIENT_SECRET = "client_secret";
public static final String GRANT_TYPE = "grant_type";
public static final String USERNAME = "username";
public static final String PASSWORD = "password";
public static final String ACCESS_TOKEN = "access_token";
public static final String EXPIRES_IN = "expires_in";

private static final BtibLogger LOG = BtibLogger.getLogger(TYPE);

private Timer timer;
private String token;

////////////////////////////////////////////////////////////////
// Actions Implementation
////////////////////////////////////////////////////////////////

/**
 * Ping base url
 */
public void doPing()
{
    try
    {
        this.setLastAttempt(BAbsTime.now());

        // Check enabled
        if (!this.getEnabled())
        {
            throw new Exception(this.getDisplayName(null) + " is disabled");
        }

        // Ping ping url
        HttpTool.makeGet(new URL(this.getPingUrl()));

        CompTool.setOk(this);
        this.setLastSuccess(BAbsTime.now());
    }
    catch (Exception e)
    {
        CompTool.setFault(this, "Cannot ping: " + e.getMessage(), e, LOG);
        this.setLastFailure(BAbsTime.now());
    }
}

////////////////////////////////////////////////////////////////
// BILogger
////////////////////////////////////////////////////////////////

@Override
public BtibLogger getBtibLogger()
{
    return LOG;
}
```

```
////////////////////////////////////////////////////////////////
// Static Methods
////////////////////////////////////////////////////////////////

/**
 * Get the Elephant connector
 *
 * @param base
 * @return
 */
public static Optional<BElephantConnector> get(BObject base)
{
    Optional<BBtibService> opt = BBtibService.get(base);
    if (opt.isPresent())
    {
        BExternalConnectorFolder folder = CompTool.getFirstChild(opt.get(), BExternalConnectorFolder.class);
        if (folder != null)
        {
            return Optional.ofNullable(CompTool.getFirstChild(folder, BElephantConnector.class));
        }
    }
    return Optional.empty();
}

////////////////////////////////////////////////////////////////
// Token
////////////////////////////////////////////////////////////////

/**
 * Get token
 */
public synchronized String getToken()
{
    String response = null;

    try
    {
        this.setLastAttempt(BAbsTime.now());

        // Check enabled
        if (!this.getEnabled())
        {
            throw new Exception(this.getDisplayName(null) + " is disabled");
        }

        // Token not expired
        if (this.token != null)
        {
            return this.token;
        }

        // Build body
        String body = CLIENT_ID + "=" + this.getClientId() + "&"
            + CLIENT_SECRET + "=" + this.getClientSecret() + "&"
            + GRANT_TYPE + "=" + PASSWORD + "&"
            + USERNAME + "=" + this.getUsername() + "&"
            + PASSWORD + "=" + this.getPassword();

        // Build headers
        Map<String, String> headers = new LinkedHashMap<>();
        headers.put("Content-Type", "application/x-www-form-urlencoded");

        // Make POST
        response = HttpTool.makePost(new URL(this.getTokenUrl()), body, null, headers, "\n");

        // Parse response (only access_token and respires_in are used)
        JsonObject oResponse = (JsonObject) new JsonParser().parse(response);

        // Store access token
        this.token = oResponse.get(ACCESS_TOKEN).getAsString();
```

```
        // Start a timer to make token expire
        this.startTimer(oResponse.get(EXPIRES_IN).getAsInt());

        CompTool.setOk(this);
        this.setLastSuccess(BAbsTime.now());
    }
    catch (Exception e)
    {
        this.token = null;

        CompTool.setFault(this, "Cannot get token: " + e.getMessage(), e, LOG);
        if (response != null)
        {
            LOG.severe(this, response);
        }
        this.setLastFailure(BAbsTime.now());
    }

    return this.token;
}

////////////////////////////////////////////////////////////////
// Timer
////////////////////////////////////////////////////////////////

/**
 * Start a timer to make the token expire
 *
 * @param delay
 */
private void startTimer(int delay)
{
    this.cancelTimer();

    this.timer = new Timer();
    this.timer.schedule(new TimerTask()
    {
        @Override
        public void run()
        {
            BElephantConnector.this.token = null;
        }
    }, delay);
}

/**
 * Cancel the timer
 */
private void cancelTimer()
{
    if (this.timer != null)
    {
        this.timer.cancel();
        this.timer = null;
    }
}

////////////////////////////////////////////////////////////////
// Niagara Callbacks
////////////////////////////////////////////////////////////////

@Override
public void stopped() throws Exception
{
    super.stopped();
    this.cancelTimer();
}

@Override
public void changed(Property prop, Context cx)
{
```

```
        if (!this.isRunning())
        {
            return;
        }

        if (prop == enabled)
        {
            if (this.getEnabled())
            {
                this.ping();
            }
            else
            {
                this.cancelTimer();
                this.token = null;
            }
        }
    }

    ////////////////////////////////////////////////////////////////
    // Getters / Setters
    ////////////////////////////////////////////////////////////////

    @Override
    public BIcon getIcon()
    {
        return BIcon.make("module://elephant/resource/icon/x16/elephant.png");
    }
}
```

Step 2: Create the importer

```
package fr.btib.elephant.model;

import com.google.gson.JsonArray;
import com.google.gson.JsonElement;
import com.google.gson.JsonObject;
import com.google.gson.JsonParser;
import com.google.gson.stream.JsonReader;
import fr.btib.connector.model.BModelImporter;
import fr.btib.core.BtibLogger;
import fr.btib.core.tool.HttpTool;
import fr.btib.elephant.connector.BElephantConnector;
import fr.btib.structure.model.component.item.BDefinition;
import fr.btib.structure.tagdictionary.BBtibTagDictionary;

import javax.baja.nre.annotations.Facet;
import javax.baja.nre.annotations.NiagaraProperty;
import javax.baja.nre.annotations.NiagaraType;
import javax.baja.sys.*;
import java.io.StringReader;
import java.net.URL;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.Optional;

@NiagaraType
@NiagaraProperty(
    name = "definitions",
    type = "String",
    defaultValue = "building=|floor=|zone=",
    facets = @Facet(name = "BFacets.FIELD_EDITOR", value = "\"btibStructure:DefinitionsFE\""),
    override = true
)
```

```
@NiagaraProperty(
    name = "apiUrl",
    type = "String",
    defaultValue = ""
)
public class BElephantImporter extends BModelImporter
{
    /*+ ------------ BEGIN BAJA AUTO GENERATED CODE ------------ +*/
    /*@ $fr.btib.elephant.model.BElephantImporter(3759224333)1.0$ @*/
    /* Generated Thu Jan 30 18:28:19 CET 2020 by Slot-o-Matic (c) Tridium, Inc. 2012 */

    ////////////////////////////////////////////////////////////////
    // Property "definitions"
    ////////////////////////////////////////////////////////////////

    /**
     * Slot for the {@code definitions} property.
     *
     * @see #getDefinitions
     * @see #setDefinitions
     */
    public static final Property definitions = newProperty(0, "building=|floor=|zone=", BFacets.make(BFacets.
FIELD_EDITOR, "btibStructure:DefinitionsFE"));

    ////////////////////////////////////////////////////////////////
    // Property "apiUrl"
    ////////////////////////////////////////////////////////////////

    /**
     * Slot for the {@code apiUrl} property.
     *
     * @see #getApiUrl
     * @see #setApiUrl
     */
    public static final Property apiUrl = newProperty(0, "", null);

    /**
     * Get the {@code apiUrl} property.
     *
     * @see #apiUrl
     */
    public String getApiUrl()
    {
        return this.getString(apiUrl);
    }

    /**
     * Set the {@code apiUrl} property.
     *
     * @see #apiUrl
     */
    public void setApiUrl(String v)
    {
        this.setString(apiUrl, v, null);
    }

    ////////////////////////////////////////////////////////////////
    // Type
    ////////////////////////////////////////////////////////////////

    @Override
    public Type getType()
    {
        return TYPE;
    }

    public static final Type TYPE = Sys.loadType(BElephantImporter.class);

    /*+ ------------ END BAJA AUTO GENERATED CODE -------------- +*/

    public static final String BUILDING_URL = "/building";
```

```
    public static final String FLOOR_URL = "/floor";
    public static final String ZONE_URL = "/zone";

    public static final String ID = "id";

    public static final String DISPLAY_NAME = "displayName";
    public static final String ADDRESS = "address";
    public static final String GEO_POINT = "geoPoint";
    public static final String LON = "lon";
    public static final String LAT = "lat";

    public static final String BUILDING = "building";
    public static final String FLOOR = "floor";
    public static final String ZONE = "zone";

    public static final BtibLogger LOG = BtibLogger.getLogger(TYPE);

    ////////////////////////////////////////////////////////////////
    // BModelImporter
    ////////////////////////////////////////////////////////////////

    @Override
    protected void processImport() throws Exception
    {
        // Decode definitions
        Map<String, BDefinition> map = this.decodeDefinitions();
        String[] definitions = new String[]{BUILDING, FLOOR, ZONE};
        for (String definition : definitions)
        {
            if (!map.containsKey(definition))
            {
                throw new Exception("Cannot find definition for " + definition);
            }
        }

        // Get connector
        Optional<BElephantConnector> opt = BElephantConnector.get(this);
        if (!opt.isPresent())
        {
            throw new Exception("Cannot find Elephant connector");
        }
        BElephantConnector connector = opt.get();

        // Get url and token
        String baseUrl = this.getApiUrl();
        String token = connector.getToken();
        if (token == null)
        {
            throw new Exception("Cannot get token from Elephant connector. See connector for more information");
        }

        Map<String, String> headers = new LinkedHashMap<>();
        headers.put("Authorization", "Bearer " + token);

        // Json parser and variables
        JsonParser parser = new JsonParser();

        // Pull model
        String sites = HttpTool.makeGet(new URL(baseUrl + SITE_URL), headers, "\n");

        String buildings = HttpTool.makeGet(new URL(baseUrl + BUILDING_URL), headers, "\n");
        JsonArray aBuildings = (JsonArray) this.parse(parser, buildings);
        aBuildings.forEach(oBuilding -> this.processBuilding(map.get(BUILDING), (JsonObject) oBuilding));

        String floors = HttpTool.makeGet(new URL(baseUrl + FLOOR_URL), headers, "\n");
        JsonArray aFloors = (JsonArray) this.parse(parser, floors);
        aFloors.forEach(oFloor -> this.processFloor(map.get(FLOOR), (JsonObject) oFloor));

        String zones = HttpTool.makeGet(new URL(baseUrl + ZONE_URL + "?" + LOCATION_GUID + "=" + siteId),
headers, "\n");
        JsonArray aZones = (JsonArray) this.parse(parser, zones);
```

```
    aZones.forEach(oZone -> this.processZone(map.get(ZONE), (JsonObject) oZone));
}

////////////////////////////////////////////////////////////
// Model
////////////////////////////////////////////////////////////

/**
 * Process building
 *
 * @param definition
 * @param oBuilding
 */
private void processBuilding(BDefinition definition, JsonObject oBuilding)
{
    String id = oBuilding.get(ID).getAsString();
    String displayName = oBuilding.get(DISPLAY_NAME).getAsString();
    Map<String, BValue> tags = new LinkedHashMap<>();

    this.createNode(definition, id, displayName, tags);

    JsonObject oSite = oBuilding.get(SITE).getAsJsonObject();
    this.addRelation(id, oSite.get(ID).getAsString(), BBtibTagDictionary.IS_IN, null);
}

/**
 * Process floor
 *
 * @param definition
 * @param oLevel
 */
private void processFloor(BDefinition definition, JsonObject oFloor)
{
    String id = oFloor.get(GUID).getAsString();
    String displayName = oFloor.get(DISPLAY_NAME).getAsString();
    Map<String, BValue> tags = new LinkedHashMap<>();

    this.createNode(definition, id, displayName, tags);

    JsonObject oBuilding = oFloor.get(BUILDING).getAsJsonObject();
    this.addRelation(id, oBuilding.get(ID).getAsString(), BBtibTagDictionary.IS_IN, null);
}

/**
 * Process zone
 *
 * @param definition
 * @param oZone
 */
public void processZone(BDefinition definition, JsonObject oZone)
{
    String id = oZone.get(ID).getAsString();
    String displayName = oZone.get(DISPLAY_NAME).getAsString();
    Map<String, BValue> tags = new LinkedHashMap<>();

    this.createNode(definition, id, displayName, tags);

    JsonObject oBuilding = oZone.get(BUILDING).getAsJsonObject();
    this.addRelation(id, oBuilding.get(ID).getAsString(), BBtibTagDictionary.IS_IN, null);
}

////////////////////////////////////////////////////////////
// Utils
////////////////////////////////////////////////////////////

/**
 * Parse with lenient
 *
 * @param parser
 * @param toParse
 * @return
```

```
     */
    private JsonElement parse(JsonParser parser, String toParse)
    {
        JsonReader reader = new JsonReader(new StringReader(toParse));
        reader.setLenient(true);
        return parser.parse(reader);
    }

    ////////////////////////////////////////////////////////////////
    // BILogger
    ////////////////////////////////////////////////////////////////

    @Override
    public BtibLogger getBtibLogger()
    {
        return LOG;
    }

    ////////////////////////////////////////////////////////////////
    // Getters / Setters
    ////////////////////////////////////////////////////////////////

    @Override
    public BIcon getIcon()
    {
        return BIcon.make("module://elephant/resource/icon/x16/elephant.png");
    }
}
```