NetIQ® Identity Manager™

Driver for Identity Gateway Integration Module Implementation Guide

October 2019

Legal Notices

Contents

About this Book and the Library 5

About NetIQ Corporation 7

1 Understanding the Identity Gateway Integration Module Driver 9

How the Driver Works 9
Default Driver Configuration 9
Planning to Install and Configure the Driver 10

2 Installing and Configuring the IGIM Driver 11

Checklist for Installing and Configuring the Driver 11
Rights Needed by the Driver Object 11
Installing the Driver Files 12
- Configuring the Remote Loader for the Driver Instance 12

Creating and Configuring the Driver Object 14
- Importing the Current Driver Packages 14
- Creating the Driver Object 14
- Configuring the Driver Settings 16

Deploying the Driver Object 16
Starting the Driver 17
Activating the Driver 18

3 REST Services 19

Common Response Parameters 19
Actions 19
- Add a Consumer 20
- Get the Registered Consumers 22
- Get a Consumer’s Details 23
- Modify a Consumer 24
- Delete a Consumer 26
- Get Events 26
- Flush All Events 28
- Flush Events for a Consumer 28
- Flush Events on a Page for a Consumer 29
- Obtain the Driver Filter 30
- Get the Driver Attribute Filter for an Object Class 31
- Restart the Driver 31

About this Book and the Library

This guide explains how to install and configure the Identity Manager Driver for Identity Gateway Integration Module.

Intended Audience

This guide is intended for administrators, consultants, and network engineers who require a high-level introduction to Identity Manager business solutions, technologies, and tools.

Other Information in the Library

For more information about the library for Identity Manager, see the following resources:

- Identity Manager documentation website (https://www.netiq.com/documentation/identity-manager-48/)
- Identity Manager drivers documentation website (https://www.netiq.com/documentation/identity-manager-48-drivers/)

About NetIQ Corporation

We are a global, enterprise software company, with a focus on the three persistent challenges in your environment: Change, complexity and risk—and how we can help you control them.

Our Viewpoint

Adapting to change and managing complexity and risk are nothing new

In fact, of all the challenges you face, these are perhaps the most prominent variables that deny you the control you need to securely measure, monitor, and manage your physical, virtual, and cloud computing environments.

Enabling critical business services, better and faster

We believe that providing as much control as possible to IT organizations is the only way to enable timelier and cost effective delivery of services. Persistent pressures like change and complexity will only continue to increase as organizations continue to change and the technologies needed to manage them become inherently more complex.

Our Philosophy

Selling intelligent solutions, not just software

In order to provide reliable control, we first make sure we understand the real-world scenarios in which IT organizations like yours operate — day in and day out. That's the only way we can develop practical, intelligent IT solutions that successfully yield proven, measurable results. And that's so much more rewarding than simply selling software.

Driving your success is our passion

We place your success at the heart of how we do business. From product inception to deployment, we understand that you need IT solutions that work well and integrate seamlessly with your existing investments; you need ongoing support and training post-deployment; and you need someone that is truly easy to work with — for a change. Ultimately, when you succeed, we all succeed.

Our Solutions

- Identity & Access Governance
- Access Management
- Security Management
- Systems & Application Management
- Workload Management
- Service Management

Contacting Sales Support

For questions about products, pricing, and capabilities, contact your local partner. If you cannot contact your partner, contact our Sales Support team.

Worldwide:	www.netiq.com/about_netiq/officelocations.asp
United States and Canada:	1-888-323-6768
Email:	info@netiq.com
Web Site:	www.netiq.com

Contacting Technical Support

For specific product issues, contact our Technical Support team.

Worldwide:	www.netiq.com/support/contactinfo.asp
North and South America:	1-713-418-5555
Europe, Middle East, and Africa:	+353 (0) 91-782 677
Email:	support@netiq.com
Web Site:	www.netiq.com/support

Contacting Documentation Support

Our goal is to provide documentation that meets your needs. If you have suggestions for improvements, click **Add Comment** at the bottom of any page in the HTML versions of the documentation posted at www.netiq.com/documentation. You can also email Documentation-Feedback@netiq.com. We value your input and look forward to hearing from you.

Contacting the Online User Community

Qmunity, the NetIQ online community, is a collaborative network connecting you to your peers and NetIQ experts. By providing more immediate information, useful links to helpful resources, and access to NetIQ experts, Qmunity helps ensure you are mastering the knowledge you need to realize the full potential of IT investments upon which you rely. For more information, visit http://community.netiq.com.

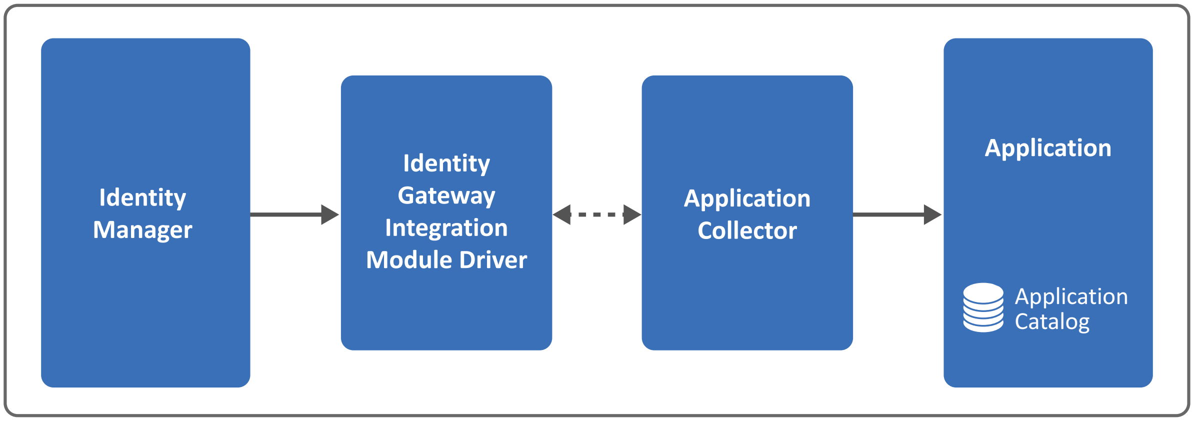
1 Understanding the Identity Gateway Integration Module Driver

The Identity Gateway Integration Module driver (IGIM) provides incremental change events for Identity Vault objects. For retrieving the events from the driver, application collectors use the REST APIs exposed by the driver. For example, NetIQ Identity Governance uses the Identity Manager Identity with Changes collector to obtain the changes.

How the Driver Works

The IGIM driver does not connect to a target application. The driver allows consumers and clients to register for change events for specific classes or attributes, or both. You can register a consumer through a REST API by specifying a unique name and an attribute filter for the consumer. For more information, see “Add a Consumer” on page 20.

The driver is initially configured with a skeletal filter, which contains only those attributes that are required by the driver to function. The filter is dynamically updated based on the consumer registration data. Therefore, you must not directly modify the filter. Multiple consumers can be registered with the driver.



When a change occurs for any classes or attributes specified in the driver filter, the driver caches the change and makes it available to the REST APIs. The application collector uses these REST APIs to obtain the changes.

All requests to the REST APIs must be authenticated. Ensure that you use the credentials of the user object that is a security equivalent of the driver object for authentication. The collector configuration must also use the same user.

Default Driver Configuration

The IGIM driver is shipped with packages. You can create the driver in NetIQ Designer for Identity Manager. If your driver requirements are different from the default configuration, modify the configuration according to your requirement.

NOTE: The driver supports only the Publisher channel. It does not support any Subscriber channel operations.

Planning to Install and Configure the Driver

The IGIM driver requires the following applications and files, at a minimum. When you installed Identity Manager, you might also have chosen to install the files for the IGIM driver.

- Identity Manager 4.7 at a minimum, particularly the following components:
  - Designer
  - Driver Set Package
    - `NOVLCOMSET` - Driver Set package for Common Settings
  - Identity Gateway Integration Module driver file
    - `IGDriverShim.jar` - Identity Gateway Integration Module driver shim
  - Identity Gateway Integration Module driver packages
    - `NETQIGIMBASE` - Identity Gateway Integration Module Base
    - `NETQIGIMDFLT` - Identity Gateway Integration Module Default

1 Installing and Configuring the IGIM Driver

This section contains information about installing and configuring the IGIM driver.

The installation and configuration process for the driver varies depending on whether you want to install the driver with the Identity Manager server or with the Remote Loader. Regardless of the method of driver installation, ensure that you have activated Identity Manager, which will automatically activate the driver. For more information, see "Activating the Driver" on page 8.

Checklist for Installing and Configuring the Driver

Use the following checklist to ensure that you complete all of the tasks required to set up and use the IGIM driver:

- Ensure that your environment meets the requirements for installing and configuring the IGIM driver. For more information, see "Planning to Install and Configure the Driver" on page 10.
- Install the driver files on the Identity Manager engine or on a Remote Loader server. For more information, see "Installing the Driver Files" on page 2.
- Ensure that you have the appropriate packages installed and imported for the driver. For more information, see "Importing the Current Driver Packages" on page 4.
- Configure the basic settings for the driver from Driver Parameters. For more information, see "Configuring the Driver Settings" on page 7.
- Assign appropriate rights to the driver object for reading and writing Identity Vault objects. For more information, see "Rights Needed by the Driver Object" on page 1.
- Deploy the driver to the Identity Vault. For more information, see "Deploying the Driver Object" on page 7.
- Start the driver. For more information, see "Starting the Driver" on page 8.

Rights Needed by the Driver Object

The driver must have appropriate rights to the Identity Vault objects that it reads or writes. You can ensure this by making the driver object security equivalent to another user object with those rights. Security Equivalence refers to an object being equivalent in rights to another object. This ensures that a client (for example, Identity Governance) accessing the driver's REST APIs has the necessary access to report the changes.

The user object (trustee) that is security equivalent to the driver object must have the following minimum permissions:

- For reading changes on the Identity Vault objects, the trustee must have the following rights (inherited to the child objects) on the base container:
    - [All Attribute Rights] - Compare, Read
    - [Entry Rights] - Browse

- For the driver operations, the trustee must have the following rights on the driver object:
  - [All Attribute Rights] - Compare, Read
  - [Entry Rights] - Browse
  - DirXML-DriverFilter - Compare, Read, and Write
  - DirXML-AccessSubmitCommand - Write
  - DirXML-AccessRun - Read
  - DirXML-AccessConfigure - Read

You must make the trustee with the above rights as a security equivalent of the driver as instructed in Step 7 in "Deploying the Driver Object" on page 7.

Installing the Driver Files

The IGIM driver is a Java-based driver and can be run on the Identity Manager engine or on a Remote Loader server. It works only with native and Java Remote Loader.

By default, the IGIM driver files are installed on the Identity Manager server at the same time as the Identity Manger engine. The installation program extends the Identity Vault's schema and installs the driver shim and the driver configuration file. It does not create the driver in the Identity Vault or upgrade an existing driver's configuration. For information about creating the driver, see "Creating the Driver Object" on page 5.

If you want to run the driver with the Remote loader, verify that the `IGDriverShim.jar` file is present in the `/opt/novell/eDirectory/lib/dirxml/classes` or `C:\novell\remoteloader\lib` directory for the Remote Loader. Otherwise, copy the file from the Identity Vault server to the appropriate directory for your operating system. You must configure the driver to use an SSL connection between the Remote Loader and the Identity Manager server. For more information, see Creating a Secure Connection to the Identity Manager Engine in the *NetIQ Identity Manager Driver Administration Guide*.

Before using the Remote Loader, you must configure the Remote Loader and the driver. For more information, see "Configuring the Remote Loader for the Driver Instance" on page 2.

Configuring the Remote Loader for the Driver Instance

To configure the driver to work with the Remote Loader, specify the following configuration parameters from the command line, in a configuration file (UNIX or Linux), or in the Remote Loader Console (Windows):

- -description (optional)
- -commandport
- connection parameters:
  - port (mandatory)
  - address
  - fromaddress
  - handshaketimeout
  - rootfile

    - keystore
    - storepass
    - localaddress
    - hostname
    - kmo
    - secureprotocol
    - enforceSuiteB
    - useMutualAuth
- trace file parameters (optional):
    - -trace
    - -tracefile
    - -tracefilemax
- -javaparam
- -class or -module

For more information about specifying values for these parameters, see Configuring the Remote Loader and Drivers in the *NetIQ Identity Manager Driver Administration Guide*.

NetIQ provides a sample configuration file `config8000.txt` to help you configure the Remote Loader and drivers for use with your application shim on Linux. By default, the sample file is located in the `/opt/novell/dirxml/doc` directory. You can save the file in the default path or choose a different location for the file. On Windows, specify the configuration details at the time of adding the driver instance to the Remote Loader in the Remote Loader Console.

Below is an example of a configuration file with some sample values.

```
-description "IGIM Driver"
-commandport 8000
-connection "port=8090"
-trace 3
-tracefile "/opt/netiq/igimdriver.log" (or
"C:\novell\remoteloader\64bit\IGIMDriver-Trace.log" on Windows)
-tracefilemax 100M
-class "com.netiq.idm.driver.igdriver.IGDriverShim"
```

Once the driver instance is running, you can use the command line on Linux or Remote Loader Console on Windows to instruct the Remote Loader to perform a function. For example, turn the trace on or off on Windows or stop the driver instance.

Configuring the Remote Loader for the Driver Instance on Linux

This section assumes that the `IGDriverShim.jar` file is present in the `/opt/novell/eDirectory/lib/dirxml/classes` directory for the Remote Loader.

1. Log in to the computer where you installed the Remote Loader.
2. In a text editor, create a new file or open the sample file `config8000.txt` from the `/opt/novell/dirxml/doc` directory.
3. Add the configuration parameters to the file.

4 Save the file.

5 Note the port number associated with the Remote Loader instance. You need this value when configuring the driver in Designer.

6 (Optional) For the Remote Loader to start the driver automatically when your system starts, save the file to the `/etc/opt/novell/dirxml/rdxml` directory.

Configuring the Remote Loader for the Driver Instance on Windows

This section assumes that the `IGDriverShim.jar` file is present in the `c:\novell\remoteloader\lib` directory for the Remote Loader.

1 Log in to the computer where you installed the Remote Loader.

2 Run `rlconsole.exe` located by default in `C:\novell\remoteloader\nnbit`.

3 Click **Add**, then specify the **Description**, **Driver**, and other parameters in the page that displays. The Remote Loader uses these parameters to configure the driver instance and stores them in a `.txt` file.

4 Save the file in the default location or specify a different location for the file. For example, `C:\novell\remoteloader\IGIMRemoteLoader-Config.txt`.

5 Note the port number associated with the Remote Loader instance. You need this value when configuring the driver in Designer.

6 (Optional) For the Remote Loader to start the driver automatically when your system starts, verify that **Establish a Remote Loader service for this driver instance** is selected.

To configure the driver as an application, deselect this setting. In this case, you need to start the driver manually.

7 Click **OK**.

Creating and Configuring the Driver Object

This section helps you create and configure the IGIM driver. You perform these tasks in your project in Designer.

- Importing the Current Driver Packages
- Creating the Driver Object
- Configuring the Driver Settings

Importing the Current Driver Packages

NetIQ regularly provides updates to the Identity Manager drivers. You must have the latest content for the IGIM driver. For more information about the packages, see "Planning to Install and Configure the Driver" on page 10.

1 Open Designer.

2 Select **Help > Check for Package Updates**.

3. Select the updated packages for the driver that you want to update. Or, click **Select All** to import all of the packages displayed.

   By default, only the base packages are displayed. Deselect **Show Base Packages Only** to display all packages.
4. Click **OK** to import the selected packages, then click **OK** in the successfully imported packages message.
5. When the update completes, restart Designer for the changes to take effect.

Creating the Driver Object

This section helps you configure the IGIM driver and establish its basic settings.

NOTE: The IGIM driver requires `NOVLCOMSET`, the driver set package for common settings. Ensure that you import this packages before configuring the driver. For more information about the required packages, see "Planning to Install and Configure the Driver" on page 10.

---

1. In the **Modeler** view of Designer, select **Developer**.
2. (Conditional) If you have more than one driver set in the Identity Vault, select the driver set in the **Modeler** view to which you want to add the driver.
3. In the **Palette** view, expand **Service**.
4. Drag **Identity Gateway Integration Module** to the **Modeler** view.

   This action opens the Driver Configuration Wizard.
5. For **Select Driver Base Configuration,** select **Identity Gateway Integration Module Base**, then click **Next**.
6. For **Optional Features**, select the following item:
   - `Identity Gateway Integration Module Default`
7. Click **Next**.
8. For **Driver Name**, specify a value. For example, `Identity Gateway Integration Module Driver`.
9. Click **Next**.
10. (Conditional) On the **Driver Parameters** page, specify the following details, and then click **Next**:
    - **Preferred Server:** Specify the selection mode for the preferred server.The default setting is **Auto**. This setting automatically selects a preferred server from the list of servers associated with the driver set. To select a specific server, change the setting to **Manual**.
    - **Default Page Size:** Specify the page size of the cached event pages. The default value is `100`.
    - **Flush Strategy:** Specify the method to remove the already read events. The options are **Auto** and **Manual**. The default setting is **Auto**. If you want the driver to purge the read events automatically, leave the setting as default. If the reading client must explicitly purge the older events, change the setting to **Manual**.
    - **Hostname:** Specify the network interface on which the driver accepts connections for the REST server. To listen on all interfaces, leave the field empty.

- **Port:** Specify the port number on which the driver accepts connections for the REST server.
- **Connection Protocol:** Specify the connection protocol to use. It is recommended to use a secured (**https**) connection. When you select **https**, the following parameters are displayed:
    - **Driver configuration mode:** Specify the configuration mode for the driver. Select **Local** to configure the driver to run with the Identity Manager engine. To configure the driver to run with the Remote Loader, select **Remote**.
    - **KMO (for secure connection):** If you select **Local** in **Driver configuration mode**, specify the key name of the Key Material Object (KMO) that contains the keys and certificate that the REST server uses for an SSL connection with the Identity Manager Identity with Changes collector. The default value is **SSL CertificateDNS**.
    - **Keystore file:** If you select **Remote** in **Driver configuration mode**, specify the name and path of the keystore file containing the trusted certificates that the REST server uses when the remote server is configured for server authentication.
    - **Keystore Password:** Specify the password for the private key for authenticating to the remote server.
- **Heartbeat Interval (in minutes):** Allows the driver to send a periodic status message on the Publisher channel when there has been no Publisher channel traffic for the given number of minutes. The default value is 1 minute.

11 (Conditional) Select *Yes* or *No* to determine if the driver will use the Remote Loader. If you select *No*, skip to Step 12. If you select *Yes*, use the following information to complete the Remote Loader configuration, then click **Next**:

- **Host Name:** Specify the hostname or IP address of the server where the driver's Remote Loader service is running.
- **Port:** Specify the port number where the Remote Loader is installed and running. The default port number is `8090`.
- **KMO:** Specify the key name of the Key Material Object (KMO) that contains the keys and certificate the Remote Loader uses for an SSL connection. This parameter is only used when you use SSL for connections between the Remote Loader and the Identity Manager engine.
- **Other Parameters:** Specify any other parameters required to connect to the Remote Loader. Any parameters specified must use a key-value pair format, as follows: `paraName1=paraValue1 paraName2=paraValue2`.
- **Remote Password:** Specify the Remote Loader's password as defined on the Remote Loader. The Identity Manager server (or Remote Loader) requires this password to authenticate to the Remote Loader.
- **Driver Password:** Specify the driver object password that is defined in the Remote Loader service. The Remote Loader requires this password to authenticate to the Identity Manager server.

12 Review the summary of tasks that will be completed to create the driver, then click **Finish**.

After you have installed the driver, you can change the configuration for your environment.

If you do not need to configure the driver, continue with "Deploying the Driver Object" on page 7.

Configuring the Driver Settings

There are many settings that can help you customize and optimize the driver. The settings are defined in Driver Parameters located on the Driver Configuration page. These settings must be configured properly for the driver to start and function correctly.

To access the Driver Properties page:

1. Open your project.
2. In the Modeler view, right-click the driver icon or the driver line, then select **Properties**.
3. Make the changes you want, then continue with "Deploying the Driver Object" on page 7.

Deploying the Driver Object

After you create, configure, or modify the driver, you must deploy the IGIM driver to the Identity Vault.

1. In Designer, open your project.
2. In the Modeler view, right-click the driver icon or the driver line, then select **Live > Deploy**.
3. If you are authenticated to the Identity Vault, skip to Step 5; otherwise, specify the following information:

   Host: Specify the IP address or DNS name of the server hosting the Identity Vault.

   Username: Specify the DN of the user object used to authenticate to the Identity Vault.

   Password: Specify the user's password.
4. Click **OK**.
5. Read through the deployment summary, then click **Deploy**.
6. Read the success message, then click **OK**.
7. Click **Define Security Equivalence** to assign rights to the driver as specified.

   NOTE: You must use the credentials of the user object that is a security equivalent of the driver object. The collector configuration must also use the same user.
8. Click **Exclude Administrative Roles** to exclude users that should not be synchronized.

   You should exclude any administrative User objects (for example, Admin and DriversUser) from synchronization.

   8a Click **Add**, then browse to and select the user object you want to exclude.

   8b Click **OK**.

   8c Repeat Step 8a and Step 8b for each object you want to exclude.

   8d Click **OK**.
9. Click **OK**.

Starting the Driver

When a driver is created, it is stopped by default. To make the driver work, you must start the driver. Identity Manager is an event-driven system, so after the driver is started, it won't do anything until an event occurs. You can use Identity Console or `dxevent` commands to start the driver.

To start the driver using Designer:

1. In Designer, open your project.
2. In the Modeler view, right-click the driver icon or the driver line, then select **Live > Start Driver**.

To start the driver using Identity Console:

1. In Identity Console, click **IDM Administration**.
2. Browse to and select the driver set object that contains the driver you want to start.
3. Click the upper right corner of the driver, then click **Start driver**.

For instructions about starting and stopping the Remote Loader and driver instances on Linux and Windows, see Starting and Stopping the Remote Loader in the *NetIQ Identity Manager Driver Administration Guide*.

Activating the Driver

The Identity Manager driver for IGIM is automatically activated as part of Identity Manager activation. If you create the driver in a driver set where you have already activated the Identity Manager server and service drivers, the driver inherits the activation from the driver set.

If you create the driver in a driver set that has not been previously activated, the driver will run in the evaluation mode for 90 days. You must activate the driver during the evaluation period; otherwise, the driver will be disabled. If you try to run the driver, `dstrace` displays an error message indicating that you need to reactivate the driver to use it. For information on activation, refer to Activating Identity Manager in the *NetIQ Identity Manager Overview and Planning Guide*.

3 REST Services

The IGIM driver exposes REST APIs that enable you to perform the following operations for clients and consumers:

- Create, read, update, and delete consumers
- Obtain or delete events from the Identity Vault for a particular consumer
- Restart the driver
- View the driver filter

The APIs use HTTP methods that return results in JSON format. You can configure `HTTP` or `HTTPS` access to enable the REST APIs in the driver parameters. `HTTP` connections are not secure because they exchange credentials and data in clear text. You are highly recommended to use `HTTPS`.

The APIs are protected by using Basic Authentication, and all requests require a base64-encoded username and password included in the Authorization header. All requests to the APIs must be authenticated by using the credentials of the user object that is a security equivalent of the driver object. The collector configuration must also use the same user.

You can access the APIs after the driver is started. To invoke the APIs, use the following base URL in a browser:

```
https://<hostname>:<port>/idv/driver
```

The APIs support GET, POST, PUT, and DELETE requests.

By default, the number of events displayed per page is `100`. You can change this setting in driver parameters or by using the `pageSize` query parameter in the APIs.

Common Response Parameters

The following list contains the parameters whose values are returned in response JSON by all actions pertaining to obtaining events. Any action-specific parameters are listed in the topic for that action.

- `size` - Specifies the number of records returned. Type: `String`
- `hasMore` - Returns `true` if there are additional events. Otherwise, it returns `false`. Type: `Boolean`
- `pageId` - Specifies the ID of the page containing the events. If you configure the client to manually remove the read events, use this parameter to delete the page containing the events. Type: `String`

Actions

The following actions are supported:

- "Add a Consumer" on page 20
- "Get the Registered Consumers" on page 22
- "Get a Consumer's Details" on page 23

- "Modify a Consumer" on page 24
- "Delete a Consumer" on page 26
- "Get Events" on page 26
- "Flush All Events" on page 28
- "Flush Events for a Consumer" on page 28
- "Flush Events on a Page for a Consumer" on page 29
- "Obtain the Driver Filter" on page 30
- "Get the Driver Attribute Filter for an Object Class" on page 31
- "Restart the Driver" on page 31

Add a Consumer

Request Type

POST

API

```
/idv/driver/consumer
```

Sample URL

```
https://<hostname>:<port>/idv/driver/consumer
```

Description

Adds a consumer.

A consumer represents a view in the application. For example, Identity Governance. You can add multiple consumers for the same set of object classes or attributes. The driver filter is constructed from the classes or attributes registered for each consumer. Direct update to the filter is not available.

To eliminate the need for issuing multiple REST calls for flushing events and restarting the driver, specify the following parameters in the request payload:

- `purgeParam`: Purges the cached events of all or specific consumers.
- `restartParam:` Restarts the driver automatically after the duration specified for the parameter. This is a mandatory parameter because the changes will not be applied to the consumer until the driver is restarted.

Sample1 Request

```
POST https://192.168.0.1:7708/idv/driver/consumer
Authorization: Basic Y249YWRtaW4sb3U9c2Esbz1zeXN0ZW06bm92ZWxs
Content-Type: application/json
{
"consumer": {
"consumerName": "Acme",
"filter": [{
"className": "inetorgperson",
"attributes": [
"cn",
"sn",
"givenname"
]
}]
}
}
```

Sample1 Response

```
200 OK
Content-Type: application/json
Date: Wed, 10 Oct 2018 08:26:44 GMT
Server: Jetty(9.4.z-SNAPSHOT)
Transfer-Encoding: chunked
{
"consumerName": "Acme",
"consumerId": "2119f87e-899a-4e91-97e0-c7c4e28daf09",
"filter": [{
"className": "inetorgperson",
"attributes": ["cn", "sn", "givenname"]
}]
}
```

Sample2 Request

```
POST https://192.168.0.1:7708/idv/driver/consumer
Authorization: Basic Y249YWRtaW4sb3U9c2Esbz1zeXN0ZW06bm92ZWxs
Content-Type: application/json
{
   "consumer":{
      "consumerName":"LooneyCorporation",
      "filter":[
         {
            "className":"inetorgperson",
            "attributes":[
               "cn",
               "fullName",
               "title"
            ]
         }
      ]
   },
   "restartParam":{
"initialDelay":5000
},
   "purgeParam":{"purgeConsumers":["2119f87e-899a-4e91-97e0-c7c4e28daf09"]
   }
}
```

NOTE

- The `purgeConsumers` parameter accepts the set of consumer IDs for which all the cached events are purged. To purge all cached events for all consumers, specify the `purgeParam` parameter in the request payload in the following syntax:

  ```
  "purgeParam":{ "purgeAll":true}
  ```

- The `restartParam` parameter uses the value of `initialDelay` in milliseconds after that the driver is restarted.

Sample2 Response

```
200 OK
Content-Type: application/json
Server: Jetty(9.4.z-SNAPSHOT)
Transfer-Encoding: chunked
{
"consumerName": "LooneyCorporation",
"consumerId": "6c0f26da-c83e-4c1f-a90d-7f8de2ff3129",
"filter": [
        {
            "className": "inetorgperson",
            "attributes": [
                "cn",
                "fullName",
                "title"
            ]
        }
    ]
}
```

Authentication

Yes

Error Codes

- 200 - OK
- 204 - No response, OK
- 400 - Bad Request, ERROR
- 401 - Authentication Failure, ERROR
- 404 - Not Found, ERROR
- 500 - Driver Error, ERROR

Get the Registered Consumers

Request Type

GET

API

```
/idv/driver/consumer
```

Sample URL

```
https://<hostname>:<port>/idv/driver/consumer
```

Description

Obtains the details of registered consumers.

Sample Request

```
GET https://192.168.0.1:7708/idv/driver/consumer

Authorization: Basic Y249YWRtaW4sb3U9c2Esbz1zeXN0ZW06bm92ZWxs
```

Sample Response

```
200 OK
Content-Type: application/json
Date: Wed, 10 Oct 2018 08:30:17 GMT
Server: Jetty(9.4.z-SNAPSHOT)
Transfer-Encoding: chunked
{
"consumers": [{
"consumerName": "Acme",
"consumerId": "2119f87e-899a-4e91-97e0-c7c4e28daf09",
"filter": [{
"className": "inetorgperson",
"attributes": ["cn", "sn", "givenname"]
}]
}, {
"consumerName": "LooneyCorporation",
"consumerId": "6c0f26da-c83e-4c1f-a90d-7f8de2ff3129",
"filter": [{
"className": "inetorgperson",
"attributes": ["cn", "fullName", "title"]
}]
}]
}
```

Authentication

Yes

Error Codes

- 200 - OK
- 204 - No response, OK
- 400 - Bad Request, ERROR
- 401 - Authentication Failure, ERROR
- 404 - Not Found, ERROR
- 500 - Driver Error, ERROR

Get a Consumer’s Details

Request Type

GET

API

```
/idv/driver/consumer/{consumerId}
```

Sample URL

```
https://<hostname>:<port>/idv/driver/consumer/{consumerId}
```

Description

Obtains the details of a consumer.

Sample Request

```
GET https://192.168.0.1:7708/idv/driver/consumer/2119f87e-899a-4e91-97e0-
c7c4e28daf09
Authorization: Basic Y249YWRtaW4sb3U9c2Esbz1zeXN0ZW06bm92ZWxs
```

Sample Response

```
200 OK
Content-Type: application/json
Date: Wed, 10 Oct 2018 08:30:57 GMT
Server: Jetty(9.4.z-SNAPSHOT)
Transfer-Encoding: chunked
{
"consumerName": "Acme",
"consumerId": "2119f87e-899a-4e91-97e0-c7c4e28daf09",
"filter": [{
"className": "inetorgperson",
"attributes": ["cn", "sn", "givenname"]
}]
}
```

Authentication

Yes

Error Codes

- 200 - OK
- 204 - No response, OK
- 400 - Bad Request, ERROR
- 401 - Authentication Failure, ERROR
- 404 - Not Found, ERROR
- 500 - Driver Error, ERROR

Modify a Consumer

Request Type

PUT

API

```
/idv/driver/consumer
```

Sample URL

```
https://<hostname>:<port>/idv/driver/consumer
```

Description

Updates a consumer.

To eliminate the need for issuing multiple REST calls for flushing events and restarting the driver, specify the following parameters in the request payload:

- `purgeParam`: Purges the cached events of all or specific consumers.
- `restartParam:` Restarts the driver automatically after the duration specified for the parameter. This is a mandatory parameter because the changes will not be applied to the consumers until the driver is restarted.

Sample Request

```
PUT https://192.168.0.1:7708/idv/driver/consumer
Authorization: Basic Y249YWRtaW4sb3U9c2Esbz1zeXN0ZW06bm92ZWxs
Content-Type: application/json
{
   "consumer":{
      "consumerName":"Acme",
      "filter":[
         {"className":"inetorgperson",
            "attributes":["givenname"]
         }
      ]
   },
   "restartParam":{
"initialDelay":5000
   },
   "purgeParam":{"purgeConsumers":["2119f87e-899a-4e91-97e0-c7c4e28daf09”]
   }
}
```

Sample Response

```
200 OK
Content-Type: application/json
Server: Jetty(9.4.z-SNAPSHOT)
Transfer-Encoding: chunked
{
"consumerName": "Acme",
"consumerId": "2119f87e-899a-4e91-97e0-c7c4e28daf09",
"filter": [
        {
            "className": "inetorgperson",
            "attributes": ["givenname"]
        }
    ]
}
```

Authentication

Yes

Error Codes

- 200 - OK
- 204 - No response, OK
- 400 - Bad Request, ERROR
- 401 - Authentication Failure, ERROR
- 404 - Not Found, ERROR
- 500 - Driver Error, ERROR

Delete a Consumer

Request Type

DELETE

API

```
/idv/driver/consumer/{consumerId}
```

Sample URL

```
https://<hostname>:<port>/idv/driver/consumer/{consumerId}
```

Description

Deletes a consumer.

Sample Request

```
DELETE https://192.168.0.1:7708/idv/driver/consumer/2119f87e-899a-4e91-97e0-
c7c4e28daf09
Authorization: Basic Y249YWRtaW4sb3U9c2Esbz1zeXN0ZW06bm92ZWxs
```

Sample Response

```
204 No Content
Content-Type: application/json
Date: Wed, 10 Oct 2018 08:54:11 GMT
Server: Jetty(9.4.z-SNAPSHOT)
```

Authentication

Yes

Error Codes

- 200 - OK
- 204 - No response, OK
- 400 - Bad Request, ERROR
- 401 - Authentication Failure, ERROR
- 404 - Not Found, ERROR
- 500 - Driver Error, ERROR

Get Events

Request Type

GET

API

```
/idv/events/{consumerId}
```

Query Parameters

`pageSize`

- `Default`: 25
- `Min`: 1
- `Max`: 500

Sample URL

```
https://<hostname>:<port>/idv/events/{consumerId}
```

Description

Obtains a page of cached events for the specified consumer. If you are registering a consumer, it returns a `consumerId` for the consumer.

Sample Request

```
GET https://192.168.0.1:7708/idv/events/6c0f26da-c83e-4c1f-a90d-7f8de2ff3129

Authorization: Basic Y249YWRtaW4sb3U9c2Esbz1zeXN0ZW06bm92ZWxs
```

Sample Response

```
200 OK
Content-Type: application/json
Date: Thu, 11 Oct 2018 08:11:22 GMT
Server: Jetty(9.4.z-SNAPSHOT)
Transfer-Encoding: chunked
{
"events": [{
"eventType": "modify",
"objectClass": "inetorgperson",
"srcDn": "CN=ab4,OU=engg,OU=users,O=data",
"association": "uv8QJqFRUkSxObr\/ECahUQ==",
"attributes": {
"cn": {
"removeAllValues": true,
"addValue": ["ab4"]
},
"fullName": {
"removeAllValues": true,
"addValue": ["ab4 ab4"]
},
"title": {
"removeAllValues": true
}
}
}, {
"eventType": "move",
"objectClass": "inetorgperson",
"srcDn": "CN=ab4,OU=engg,OU=users,O=data",
"cachedTime": "20171012042831.471Z",
"oldSrcDn": "CN=ab4,OU=users,O=data",
"association": "uv8QJqFRUkSxObr\/ECahUQ==",
"parent": {
"srcDn": "OU=engg,OU=users,O=data"
}
}],
"size": 2,
"hasMore": false,
"pageId": "a763bf7d-6b3b-4f5f-8881-ecad91483f50"
}
```

In `Manual` flush mode, the cached events are cleaned up by the Flush API. The API returns an empty response in absence of change events.

Authentication

Yes

Error Codes

- 200 - OK
- 204 - No response, OK
- 400 - Bad Request, ERROR
- 401 - Authentication Failure, ERROR
- 404 - Not Found, ERROR
- 500 - Driver Error, ERROR

Flush All Events

Request Type

DELETE

API

```
/idv/events
```

Sample URL

```
https://<hostname>:<port>/idv/events
```

Description

Deletes all cached events.

Sample Request

```
DELETE http://192.168.0.1:7708/idv/events
Authorization: Basic Y249YWRtaW4sb3U9c2Esbz1zeXN0ZW06bm92ZWxs
```

Sample2 Response

```
204 No Content
Content-Type: application/json
Date: Wed, 12 Sep 2018 08:55:50 GMT
Server: Jetty(9.4.z-SNAPSHOT)
```

Authentication

Yes

Error Codes

- 204 - No response, OK
- 401 - Authentication Failure, ERROR
- 500 - Driver Error, ERROR

Flush Events for a Consumer

Request Type

DELETE

API

```
/idv/events/{consumerId}
```

Sample URL

```
https://<hostname>:<port>/idv/events/{consumerId}
```

Description

Deletes all cached events for a consumer.

Sample Request

```
DELETE http://192.168.0.1:7708/idv/events/6c0f26dac83e-4c1f-a90d-7f8de2ff3129
Authorization: Basic Y249YWRtaW4sb3U9c2Esbz1zeXN0ZW06bm92ZWxs
```

Sample Response

```
204 No Content
Content-Type: application/json
Date: Wed, 10 Oct 2018 09:28:12 GMT
Server: Jetty(9.4.z-SNAPSHOT)
```

Authentication

Yes

Error Codes

- 204 - No response, OK
- 401 - Authentication Failure, ERROR
- 500 - Driver Error, ERROR

Flush Events on a Page for a Consumer

Request Type

DELETE

API

```
/idv/events/{consumerId}/{pageId}
```

Sample URL

```
https://<hostname>:<port>/idv/events/{consumerId}/{pageId}
```

Description

Deletes a cached event page for a consumer.

Sample Request

```
DELETE http://192.168.0.1:7708/idv/events/6c0f26dac83e-4c1f-a90d-7f8de2ff3129/
bf4dcbf8-31a8-4701-a9e4-4fc2ad7d40d4
Authorization: Basic Y249YWRtaW4sb3U9c2Esbz1zeXN0ZW06bm92ZWxs
```

Sample Response

```
204 No Content
Content-Type: application/json
Date: Wed, 12 Sep 2018 08:54:39 GMT
Server: Jetty(9.4.z-SNAPSHOT)
```

Authentication

Yes

Error Codes

- 204 - No response, OK
- 401 - Authentication Failure, ERROR
- 404 - Not Found, ERROR
- 500 - Driver Error, ERROR

Obtain the Driver Filter

Request Type

GET

API

```
/idv/driver/filter
```

Sample URL

```
https://<hostname>:<port>/idv/driver/filter
```

Description

Obtains details of the driver filter.

Sample Request

```
GET http://192.168.0.1:7708/idv/driver/filter
Authorization: Basic Y249YWRtaW4sb3U9c2Esbz1zeXN0ZW06bm92ZWxs
```

Sample Response

```
{
    "objectClasses": {
        "className": "inetorgperson",
        "attributes": [
"cn","fullName","title"
        ]
    }
}
```

Authentication

Yes

Error Codes

- 200 - OK
- 204 - No response, OK
- 401 - Authentication Failure, ERROR
- 404 - Not Found, ERROR
- 500 - Driver Error, ERROR

Get the Driver Attribute Filter for an Object Class

Request Type

GET

API

```
/idv/driver/filter/{objectclass}
```

Sample URL

```
https://<hostname>:<port>/idv/driver/filter/{objectclass}
```

Description

Obtains details of attribute filter of an object class.

Sample Request

```
GET
http://192.168.0.1:7708/idv/driver/filter/inetorgperson
Authorization: Basic Y249YWRtaW4sb3U9c2Esbz1zeXN0ZW06bm92ZWxs
```

Sample Response

```
{
    "className": "inetorgperson",
    "attributes": [
"cn","fullName","title"
        ]
}
```

Authentication

Yes

Error Codes

- 200 - OK
- 204 - No response, OK
- 401 - Authentication Failure, ERROR
- 404 - Not Found, ERROR
- 500 - Driver Error, ERROR

Restart the Driver

Request Type

PUT

API

```
/idv/driver/restart
```

Sample URL

```
https://<hostname>:<port>/idv/driver/restart
```

Description

Restarts the driver.

Sample Request

```
PUT http://192.168.0.1:7708/idv/driver/restart
Authorization: Basic Y249YWRtaW4sb3U9c2Esbz1zeXN0ZW06bm92ZWxs
Content-Type: application/json
{
"initialDelay":10000
}
```

Sample Response

```
200 OK
Content-Type: application/json
Date: Mon, 10 Sep 2018 08:22:54 GMT
Server: Jetty(9.4.z-SNAPSHOT)
Transfer-Encoding: chunked
{
"submitted": true
}
```

Authentication

Yes

Error Codes

- 200 - OK
- 401 - Authentication Failure, ERROR
- 500 - Driver Error, ERROR