DTSS-Configuration Procedures

- Get Started With DTSS
  - Synchronization Overview
    - DTSS Pub-Sub Support
      - DTSS JMS Tenant Synchronization Service
        - Implementing the Service Process
  - Tenant Registrations
    - Data Tenant Registration
    - DTSS System Configuration
    - Unregister a Single or a Set of Data Tenant(s) in DTSS (delete)
    - Get all Data Tenants
    - Get Specified Data Tenant
    - Validate Data Tenant
    - Customer Tenant Registration
    - Unregister a Single or a Set of Customer Tenant(s)
    - Get All Customer Tenants
    - Get Specified Customer Tenant
    - Validate Customer Tenant
  - Tenant Subscription
    - Create Subscription (Between Registered DTs and CTs)
      - Create Subscription (Subscribe Customer Tenant to Data Tenant)
      - Tenant Subscription Events Configuration
      - Tenant Subscription Synchronization Configuration
      - Tenant Subscription Transform Configuration
        - Golden Record Configuration
      - Tenant Subscription Removal of startDate/endDate in Relations
      - Tenant Subscription Tasks Configuration
        - OnFinish Configuration
        - ConsistencyReport Configuration
      - Validate Tenant Subscription
        - Tenant Subscription Verification
          - Data Tenant Configuration Verification
          - Customer Tenant Subscription Verification
          - Subscription Verification
          - Get invalid subscriptions
          - Email Notification
      - Get/Filter Tenant Subscription
      - Delete Tenant Subscription
      - Change groupContributors Configuration Parameter
  - Entity Subscription
    - Import Entity from Data Tenant to Customer Tenant
    - Merge Entity
    - UnMerge Entity
    - Import Relations From Data Tenant to Customer Tenant
      - Import Single Relation
      - Import Multiple Relations
      - Mark Entities as notMatch
      - Delete notMatch
      - Get entity notMatch
  - Potential Matches Search in DT
  - Potential Matches Search in CT
  - Match Explanation
  - Entities Search by DT Potential Matches
  - Facet Search by DT Potential Matches
  - Entities Search in Data Tenants
- Contracts
  - Configuration

Get Started With DTSS

Refer to these topics (and related sub-topics) in the order given for the information to initiate DTSS.

- Tenant Registrations
- Tenant Subscription
- Entity Subscription
- DTSS Tasks
- Admin Features

Synchronization Overview

DTSS mechanism of automatic tenant synchronization can be done in the two ways:

- Real-Time: After any modification (event) of the Data/Customer tenant DTSS will perform a synchronization of the single entity or relation.
- Pull Task: DTSS allows to create a Pull Task that will perform the tenants synchronization for all the modifications from certain time interval.

For the events currently supported and how to activate or deactivate certain ones please refer to Events Configuration.

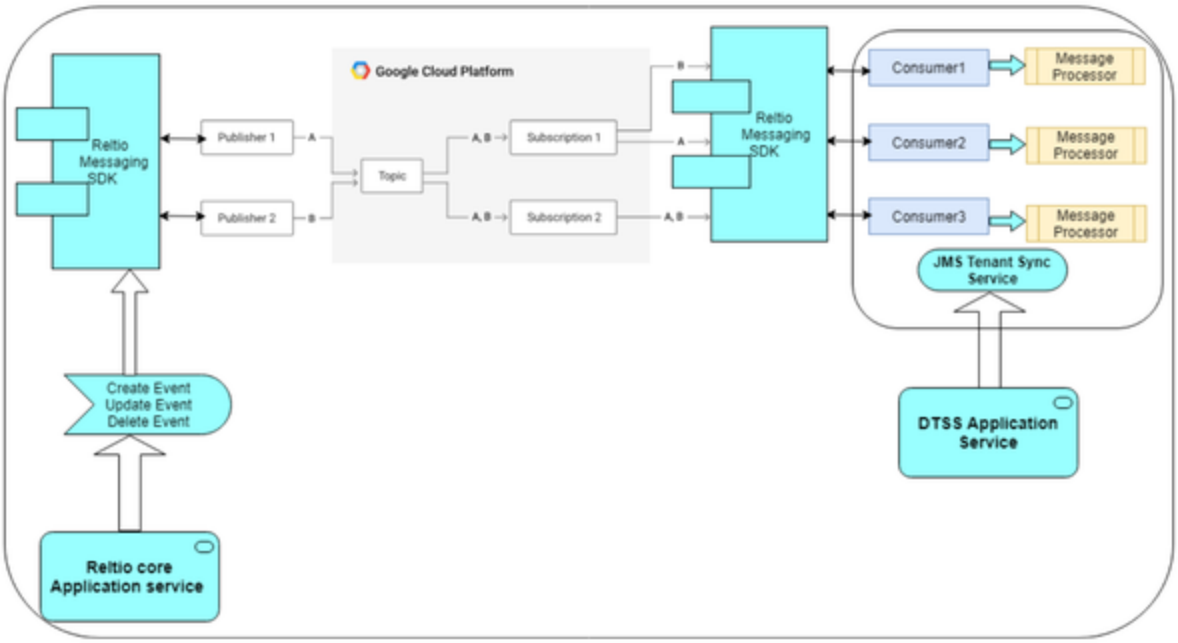
DTSS allows to set up synchronization precisely by specifying rules. Please refer to Synchronization Configuration.

DTSS Pub-Sub Support

DTSS provides the ability to synchronize tenants through the JMS process through the support of the Google cloud Pub/Sub message processing client in real-time mode. DTSS Pub/Sub works in pull mode and only reads the message from GCP and then processes it. For each unique queue name, DTSS creates a topic, and for each unique queue name and tenant id, DTSS Pub/Sub creates a subscription.

DTSS JMS Tenant Synchronization Service

The Pub/Sub mechanism/feature is added to the current DTSS JMS Tenant Synchronization Service.



IMPLEMENTING THE SERVICE PROCESS

1. Create a Message Consumer class which implements the JMSDtssConsumer Interface.
2. Implement all required methods such as “`addHandler`”, “`removeHandler`”, “`stopAll`”, “`reInit`”, “`isHandledType`”, and “`checkConnectivity`”.
3. The Consumer processes the message in a batch, and the maximum batch size is provided to the Consumer through the constructor. The maximum batch size is configurable and is read from the properties file. For the block DTSS uses in the properties file, some file fields properties files are defined, such as `static.gcp.user`, `static.gcp.pass`, and `gcp.project.name`, where the actual values to use are mentioned.
4. Each consumer is created for each different combination of “`gcp project name`” + “`username`” + “`password`”. Example of expected values: "`customer-facing:`" + "`user@customer-facing.iam.gserviceaccount.com`" + "`encoded_password`".

Notes: DTSS only consumes the Pub/Sub messages. Publishing the messages to Pub/Sub is not part of DTSS; it’s handled by the Reltio Core API (server).

> #1: Message serialization/deserialization is handled by the Reltio-messaging SDK. In DTSS Pub/Sub the only messages read are those which are received by the Reltio-messaging SDK from GCP. Handling Message serialization/deserialization is not in the scope for DTSS Pub/Sub.
>
> #2: Subscription message consumption is handled by the Reltio-messaging SDK. Then, the consumed message sends to DTSS handler to process it.
>
> #3: Currently, DTSS does not support StackDriver.
>
> #4: DTSS Pub/Sub consumer receives a ReltioMessage object, which does not have a serial version.

Tenant Registrations

Data Tenant Registration

To create or update a set of Data Tenants, use the `{DTSSURL}/tenants/dataTenants` endpoint.

HTTP METHOD	POST
URL	`{DTSSURL}/tenants/dataTenants`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

BODY	**Data Tenant Registration Body**
	
	`[`
	`  {`
	`  "tenant_id": {`
	`  "id": "{tenant_id}"`
	`  },`
	`  "oauth_end_point_url": "{oauth_end_point_url}",`
	`  "api_url": "{reltio_api_url}",`
	`  "label": "{label}",`
	`  "name": "{name}",`
	`  "support_email": "{email}"`
	`  }`
	`]`

Parameters

Name	Required	Description
oauth_token	Yes	Authorization token of User.
`tenant_id`	Yes	id of data tenant to be registered (**Now this id must be unique in DTSS**).
`oauth_end_point_url`	Yes	OAuth token enpoint url. (Example: http://auth-stg.dev.reltio.com/oauth/token) **Must be registered as an OAuth Instance**
`api_url`	Yes	API url of Data Tenant (Example: http://sndbx.reltio.com/reltio/api).
`label`	Yes	Readable name of DT; should start with the prefix `"DT_"`.
`name`	Yes	Short Name of DT (It is used for data tenant crosswalk in CT); should start with the prefix `"DT:"`.
support_email	No	Email used for mailing in case of validation errors.

Important! To register a Data Tenant: if there is no registered OAuth Instance for the URL that you need (you receive a corresponding error message), please contact administrators to create it.

DTSS System Configuration

HTTP METHOD	`GET`
URL	`{DTSSURL}/admin/systemConfig`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

Parameters

Name	Required	Description
oauth_token	Yes	Authorization token of user.

Unregister a Single or a Set of Data Tenant(s) in DTSS (delete)

HTTP METHOD	`DELETE`
URL	`{DTSSURL}/tenants/dataTenants`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

BODY	**Data Tenant Deregistration Body**
	`[`
	`  "tenant_id1",`
	`  "tenant_id2",`
	`  ...`
	`  "tenant_idN"`
	`]`

Parameters

Name	Required	Description
`oauth_token`	Yes	Authorization token of user.
`tenant_ids`	Yes	ids of Data Tenants to unregister.

Important! Unregistering a Data Tenant doesn't remove it completely.

Get all Data Tenants

HTTP METHOD	`GET`
URL	`{DTSSURL}/tenants/dataTenants?useCache={boolean}`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

Parameters

Name	Required	Description
`oauth_token`	Yes	Authorization token of user.
`useCache`	No	Forces to use cache.

Get Specified Data Tenant

HTTP METHOD	`GET`
URL	`{DTSSURL}/tenants/dataTenants/{dataTenantId}`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

Parameters

Name	Required	Description

`oauth_token`	Yes	Authorization token of user.
`dataTenantId`	Yes	Identifier of Data Tenant.

Validate Data Tenant

Validate the configuration of a Data Tenant using a validation endpoint.

Customer Tenant Registration

HTTP METHOD	`POST`
URL	`{DTSSURL}/tenants/customerTenants`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`
BODY	**Customer Tenant Registration Body** `[ { "tenant_id": { "id": "{tenant_id}" }, "oauth_end_point_url": "{oauth_end_point_url}", "api_url": "{reltio_api_url}", "support_email": "{email}" } ]`

Parameters

Name	Required	Description
`oauth_token`	Yes	Authorization token of user.
`tenant_id`	Yes	id of customer tenant to register (**this id must be unique in DTSS**).
`oauth_end_point_url`	Yes	OAuth token enpoint url. (Example: http://auth-stg.dev.reltio.com/oauth/token) **Must be registered as an OAuth Instance**.
`api_url`	Yes	API url of Data Tenant (Example: http://sndbx.reltio.com/reltio/api).
`support_email`	No	Email to use for mailing in case of validation errors.

Important! To register customer tenant: if there is no registered OAuth Instance for the URL you need (you receive a corresponding error message), please contact administrators to create it.

Unregister a Single or a Set of Customer Tenant(s)

HTTP METHOD	`DELETE`
URL	`{DTSSURL}/tenants/customerTenants`

HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

BODY	**Customer Tenant Deregistration Body**
	`[`
	`  "tenant_id1",`
	`  "tenant_id2",`
	`  ...`
	`  "tenant_idN"`
	`]`

Parameters

Name	Required	Description
`oauth_token`	Yes	Authorization token of user.
`tenant_ids`	Yes	ids of Customer Tenants to unregister.

Important! Unregistering a customer tenant doesn't remove it completely.

Get All Customer Tenants

HTTP METHOD	`GET`
URL	`{DTSSURL}/tenants/customerTenants?useCache={boolean}`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

Parameters

Name	Required	Description
`oauth_token`	Yes	Authorization token of user.
`useCache`	No	Forces to use cache.

Get Specified Customer Tenant

HTTP METHOD	`GET`
URL	`{DTSSURL}/tenants/customerTenants/{customerTenantId}`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

Parameters

Name	Required	Description
`oauth_token`	Yes	Authorization token of user.
`customerTenantId`	Yes	Identifier of Customer Tenant.

Validate Customer Tenant

Validate configuration of Customer Tenant using a validation endpoint.

Tenant Subscription

Tenant subscription is a logic that allows you to request and obtain data from the Data Tenant to the Customer Tenant.

Create Subscription (Between Registered DTs and CTs)

Tenant subscription is a logic that allows you to request and get data from Data Tenants to Customer Tenants.

Create Subscription (Subscribe Customer Tenant to Data Tenant)

HTTP METHOD	`POST`
URL	`{DTSSURL}/subscriptions`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`
BODY	**Create Tenant Subscription Body** `[{ "dataTenant": { "id": "{dataTenantId}" }, "customerTenant": { "id": "{customerTenantId}" }, "bringGoldenRecord": "{bringGoldenRecord}", "bringInternalSources": "{bringInternalSources}", "mappings": "{mappings}", "security": "{security}", "eventsConfiguration": "{eventsConfiguration}", "importRelationsConfig": "{importConnectionConfiguration}", "groupContributors": "{groupContributors}", "supportEmail": "{email}", "fullImportLogCF": "{cassandraColumnFamily}", "fullImportLogESIndex": "{elasticsearchIndex}" }]`

Parameters

Name	Required	Description
`oauth_token`	Yes	Authorization token of user.

`dataTenantId`	Yes	Source Data Tenant id.
`customerTenantId`	Yes	Target Customer Tenant id.
`bringGoldenRecord`	Yes	This parameter means that DTSS will load the `ov=true` slice of data entity attributes. Possible values: `true` or `false`.
`bringInternalSources`	No	JSON Array of Crosswalk Source Types that are loaded from the Data Tenant to the Customer Tenant. Value `["*"]` means that all Crosswalk Source Types are loaded from the Data Tenant to the Customer Tenant. Example: `bringInternalSources` : `["configuration/sources/Veeva", "configuration/sources/AMA"]`.
`hiddenDt`	No	If set to `true` in the DTSS subscription configuration, the Data Tenant will not be visible in the UI (left-hand side panel of the search results UI in the Customer Tenant), and any potential matches from the hidden Data Tenant will not be shown.
`mappings`	No	If tenant subscription transformation mappings are not defined, all attributes of the entity satisfying `bringInternalSources` and `bringGoldenRecord` are imported to the customer tenant.
`security`	No	Security configuration of Tenant Subscription.
`eventsConfiguration`	No, but it will be created automatically	Configuration of event processing.
`synchronizationConfig`	No, but it will be created automatically	Configuration of entities and relations synchronization.
`validation`	No, but it will be created automatically	The attribute “`validation`” is added to the subscription structure to provide an option for lower validation severity from FAILED to WARN. "validation": { "strictValidation": true/false, "strictDataTenantValidation": true/false, "strictCustomerTenantValidation": true/false, "strictMappingCheck": true/false } These properties are added to the DTSS properties file to assign the default values of newly added attributes: • `validation.check.subscription.mapping.strict.default` This property performs the subscription mapping strict validation check (default value is `false`). • `validation.check.tenant.data.strict.default` This property performs the data tenant strict validation check (default value is `false`). • `validation.check.tenant.customer.strict.default` This property performs the customer tenant strict validation check and (default value is `false`). • `validation.check.subscription.strict.default` This property performs the subscription strict validation check (default value is `false`).

<table>
<tr><td><code>endpointsConfig</code></td><td>No, but it will be created automatically</td><td>Advanced behavior for endpoints.

"endpointsConfig": { "useSynchronizationConfig": true|false, "ruleNameInMatchResult": true|false }

<code>useSynchronizationConfig</code>

Configuration for import entities and relationships. This provides a configurable option that allows you to define what should be downloaded when you press the Import entities button in the UI. If the option is <code>true</code>, then import entities and their associated relationships occurs. If the option is <code>false</code>, then import entities only occurs.

The DTSS properties field <code>"endpoints.entities.import.useSynchronization Config"</code> provides the default value <code>true</code> | <code>false</code>. If this properties field is not provided, then the application considers the default value as <code>false</code>.

The DTSS subscription field can be <code>subscription.endpointsConfig.useSynchroni zationConfig</code> and is located in the subscription configuration:

<code>ruleNameInMatchResult</code>

Whether PotentialmatchessearchinDT and PotentialmatchessearchinCT should return Match Rule names instead of Match Rule URIs in the result.</td></tr>
<tr><td><code>importRelationsConfig</code></td><td>No, but it will be created automatically</td><td>Configuration for import connections functionality.
"importRelationsConfig": { "defaultStrategy": "ALL|IMPORT_IF_ENTITIES_SUBSCRIBED|IMPORT_IF_START_ENTITY_SUBSCRIBED|IMPORT_IF_END_ENTITY_SUBSCRIBED", "strategyPerRelationType": [{ "relationTypes": [{"<type>"}], "strategy": "ALL|IMPORT_IF_ENTITIES_SUBSCRIBED|IMPORT_IF_START_ENTITY_SUBSCRIBED|IMPORT_IF_END_ENTITY_SUBSCRIBED" }] }

<code>defaultStrategy</code> and <code>strategy</code> copying rules description:

<table><tr><th>Value</th><th>Description</th></tr><tr><td><code>ALL</code></td><td>All of the relations.</td></tr><tr><td><code>IMPORT_IF_ENTITIES_SUBSCRIBED</code></td><td>Only if CT is subscribed on DT relation's both <code>startEntity</code> and <code>endEntity</code> entities.</td></tr><tr><td><code>IMPORT_IF_START_ENTITY_SUBSCRIBED</code></td><td>Only if CT is subscribed on at least DT relation's <code>startEntity</code> entity.</td></tr><tr><td><code>IMPORT_IF_END_ENTITY_SUBSCRIBED</code></td><td>Only if CT is subscribed on at least DT relation's <code>endEntity</code> entity.</td></tr></table></td></tr>
<tr><td><code>groupContributors</code></td><td>No</td><td>This configuration parameter is used to define if DTSS should group all crosswalks to one contributor in customer tenant. Possible values: <code>true</code> or <code>false</code>. Default value is <code>true</code>.</td></tr>
<tr><td><code>fullImportLogCF</code></td><td>No</td><td>Used by the Full Import Log (default: <code>"FullImportLogCommonCF"</code>).

Note: By default, DTSS stores all the records across all the subscriptions in the same Cassandra CF. This does not affect Full Import Log functionality and was done in order to keep fewer CFs.</td></tr>
<tr><td><code>fullImportLogESIndex</code></td><td>No</td><td>Used by the Full Import Log (default: <code>"dtss-full-import-log-index-default"</code>).

Note: By default DTSS indexes all the records cross all the subscriptions into the same ES Index. This does not affect Full Import Log functionality and was done to keep fewer indices.</td></tr>
<tr><td><code>supportEmail</code></td><td>No</td><td>Email to use for mailing in case of validation errors.</td></tr>
</table>

`validationResult`	No	If validation is requested, the field contains one of these validation results: • `NOT_CHECKED` • `SUCCESS` • `INFO` • `WARN` • `FAILED`
`partialOverride` `ForReference`	No, but it will be auto-matically created	The default value is `false`. If you make the value `true`, then all attributes are overridden. However, for references, any attributes not involved with `referencedAttributeURIs` are not touched.

Tenant Subscription Example

Example of Tenant Subscription

```
{
  "dataTenant": {
    "id": "DTData"
  },
  "customerTenant": {
    "id": "TestData"
  },
  "bringGoldenRecord": false,
  "bringInternalSources": [
    "configuration/sources/Veeva",
    "configuration/sources/AMA"
  ],
  "id": "DTData_TestData",
  "mappings": [
    {
      "copyFromDT": "configuration/entityTypes/HCP",
      "copyToCT": "configuration/entityTypes/HCP",
      "attributes": [
        {
          "copyFromDT":
"configuration/entityTypes/HCP/attributes/FirstName",
          "copyToCT": [
            "configuration/entityTypes/HCP/attributes/FirstName"
          ]
        },
        {
          "copyFromDT":
"configuration/entityTypes/HCP/attributes/LastName",
          "copyToCT": [
            "configuration/entityTypes/HCP/attributes/LastName"
          ]
        },
        {
          "copyFromDT":
```

```
"configuration/entityTypes/HCP/attributes/MiddleName",
          "copyToCT": [
            "configuration/entityTypes/HCP/attributes/MiddleName"
          ]
        },
        {
          "copyFromDT":
"configuration/entityTypes/HCP/attributes/Prefix",
          "copyToCT": [
            "configuration/entityTypes/HCP/attributes/Prefix"
          ]
        },
        {
          "copyFromDT":
"configuration/entityTypes/HCP/attributes/SuffixName",
          "copyToCT": [
            "configuration/entityTypes/HCP/attributes/SuffixName"
          ]
        },
        {
          "copyFromDT":
"configuration/entityTypes/HCP/attributes/Phone",
          "copyToCT": [
            "configuration/entityTypes/HCP/attributes/Phone"
          ],
          "attributes": [
            {
              "copyFromDT":
"configuration/entityTypes/HCP/attributes/Phone/attributes/Active",
              "copyToCT": [

"configuration/entityTypes/HCP/attributes/Phone/attributes/Active"
              ]
            },
            {
              "copyFromDT":
"configuration/entityTypes/HCP/attributes/Phone/attributes/AreaCode",
              "copyToCT": [

"configuration/entityTypes/HCP/attributes/Phone/attributes/AreaCode"
              ]
            },
            {
              "copyFromDT":
"configuration/entityTypes/HCP/attributes/Phone/attributes/CountryCode",
              "copyToCT": [

"configuration/entityTypes/HCP/attributes/Phone/attributes/CountryCode"
              ]
            },
```

```
            {
              "copyFromDT":
"configuration/entityTypes/HCP/attributes/Phone/attributes/DigitCount",
              "copyToCT": [

"configuration/entityTypes/HCP/attributes/Phone/attributes/DigitCount"
              ]
            },
            {
              "copyFromDT":
"configuration/entityTypes/HCP/attributes/Phone/attributes/FormatMask",
              "copyToCT": [

"configuration/entityTypes/HCP/attributes/Phone/attributes/FormatMask"
              ]
            },
            {
              "copyFromDT":
"configuration/entityTypes/HCP/attributes/Phone/attributes/GeoArea",
              "copyToCT": [

"configuration/entityTypes/HCP/attributes/Phone/attributes/GeoArea"
              ]
            },
            {
              "copyFromDT":
"configuration/entityTypes/HCP/attributes/Phone/attributes/GeoCountry",
              "copyToCT": [

"configuration/entityTypes/HCP/attributes/Phone/attributes/GeoCountry"
              ]
            },
            {
              "copyFromDT":
"configuration/entityTypes/HCP/attributes/Phone/attributes/LineType",
              "copyToCT": [

"configuration/entityTypes/HCP/attributes/Phone/attributes/LineType"
              ]
            },
            {
              "copyFromDT":
"configuration/entityTypes/HCP/attributes/Phone/attributes/LocalNumber",
              "copyToCT": [

"configuration/entityTypes/HCP/attributes/Phone/attributes/LocalNumber"
              ]
            },
            {
              "copyFromDT":
```

```
"configuration/entityTypes/HCP/attributes/Phone/attributes/Number",
              "copyToCT": [

"configuration/entityTypes/HCP/attributes/Phone/attributes/Number"
              ]
            },
            {
              "copyFromDT":
"configuration/entityTypes/HCP/attributes/Phone/attributes/Type",
              "copyToCT": [

"configuration/entityTypes/HCP/attributes/Phone/attributes/Type"
              ],
              "transformValue": [
                {
                  "from": "Business",
                  "to": "Work"
                }
              ]
            },
            {
              "copyFromDT":
"configuration/entityTypes/HCP/attributes/Phone/attributes/ValidationSta
tus",
              "copyToCT": [

"configuration/entityTypes/HCP/attributes/Phone/attributes/ValidationSta
tus"
              ]
            }
          ]
        },
        {
          "copyFromDT":
"configuration/entityTypes/HCP/attributes/ProfDesignation",
          "copyToCT": [
            "configuration/entityTypes/HCP/attributes/AccountType"
          ]
        },
        {
          "copyFromDT":
"configuration/entityTypes/HCP/attributes/Specialities",
          "copyToCT": [
            "configuration/entityTypes/HCP/attributes/Specialities"
          ],
          "attributes": [
            {
              "copyFromDT":
"configuration/entityTypes/HCP/attributes/Specialities/attributes/Rank",
              "copyToCT": [
```

```
"configuration/entityTypes/HCP/attributes/Specialities/attributes/Rank"
              ]
            },
            {
              "copyFromDT":
"configuration/entityTypes/HCP/attributes/Specialities/attributes/Specia
lty",
              "copyToCT": [

"configuration/entityTypes/HCP/attributes/Specialities/attributes/Specia
lty"
              ]
            },
            {
              "copyFromDT":
"configuration/entityTypes/HCP/attributes/Specialities/attributes/Specia
ltyType",
              "copyToCT": [

"configuration/entityTypes/HCP/attributes/Specialities/attributes/Specia
ltyType"
              ]
            }
          ]
        },
        {
          "copyFromDT":
"configuration/entityTypes/HCP/attributes/License",
          "copyToCT": [
            "configuration/entityTypes/HCP/attributes/License"
          ],
          "attributes": [
            {
              "copyFromDT":
"configuration/entityTypes/HCP/attributes/License/attributes/ExpirationD
ate",
              "copyToCT": [

"configuration/entityTypes/HCP/attributes/License/attributes/ExpirationD
ate"
              ]
            },
            {
              "copyFromDT":
"configuration/entityTypes/HCP/attributes/License/attributes/Number",
              "copyToCT": [

"configuration/entityTypes/HCP/attributes/License/attributes/Number"
              ]
```

```
            },
            {
              "copyFromDT":
"configuration/entityTypes/HCP/attributes/License/attributes/State",
              "copyToCT": [

"configuration/entityTypes/HCP/attributes/License/attributes/State"
              ]
            },
            {
              "copyFromDT":
"configuration/entityTypes/HCP/attributes/License/attributes/Status",
              "copyToCT": [

"configuration/entityTypes/HCP/attributes/License/attributes/Status"
              ]
            },
            {
              "copyFromDT":
"configuration/entityTypes/HCP/attributes/License/attributes/WorkType",
              "copyToCT": [

"configuration/entityTypes/HCP/attributes/License/attributes/WorkType"
              ]
            },
            {
              "copyFromDT":
"configuration/entityTypes/HCP/attributes/License/attributes/Qualifier",
              "copyToCT": [

"configuration/entityTypes/HCP/attributes/License/attributes/Qualifier"
              ]
            },
            {
              "copyFromDT":
"configuration/entityTypes/HCP/attributes/License/attributes/SubQualifie
r",
              "copyToCT": [

"configuration/entityTypes/HCP/attributes/License/attributes/SubQualifie
r"
              ]
            }
          ]
        },
        {
          "copyFromDT":
"configuration/entityTypes/HCP/attributes/Identifiers",
          "copyToCT": [
            "configuration/entityTypes/HCP/attributes/Identifiers"
```

```
          ],
          "attributes": [
            {
              "copyFromDT":
"configuration/entityTypes/HCP/attributes/Identifiers/attributes/ID",
              "copyToCT": [

"configuration/entityTypes/HCP/attributes/Identifiers/attributes/ID"
              ]
            },
            {
              "copyFromDT":
"configuration/entityTypes/HCP/attributes/Identifiers/attributes/Type",
              "copyToCT": [

"configuration/entityTypes/HCP/attributes/Identifiers/attributes/Type"
              ]
            },
            {
              "copyFromDT":
"configuration/entityTypes/HCP/attributes/Identifiers/attributes/Rank",
              "copyToCT": [

"configuration/entityTypes/HCP/attributes/Identifiers/attributes/Rank"
              ]
            },
            {
              "copyFromDT":
"configuration/entityTypes/HCP/attributes/Identifiers/attributes/Status"
,
              "copyToCT": [

"configuration/entityTypes/HCP/attributes/Identifiers/attributes/Status"
              ]
            }
          ]
        },
        {
          "copyFromDT":
"configuration/entityTypes/HCP/attributes/Credentials",
          "copyToCT": [
            "configuration/entityTypes/HCP/attributes/Credentials"
          ]
        },
        {
          "copyFromDT":
"configuration/entityTypes/HCP/attributes/Address",
          "copyToCT": [
            "configuration/entityTypes/HCP/attributes/Address"
          ]
```

```
        }
      ]
    },
    {
      "copyFromDT": "configuration/entityTypes/Location",
      "copyToCT": "configuration/entityTypes/Location",
      "attributes": [
        {
          "copyFromDT":
"configuration/entityTypes/Location/attributes/AddressLine1",
          "copyToCT": [
            "configuration/entityTypes/Location/attributes/AddressLine1"
          ]
        },
        {
          "copyFromDT":
"configuration/entityTypes/Location/attributes/AddressLine2",
          "copyToCT": [
            "configuration/entityTypes/Location/attributes/AddressLine2"
          ]
        },
        {
          "copyFromDT":
"configuration/entityTypes/Location/attributes/City",
          "copyToCT": [
            "configuration/entityTypes/Location/attributes/City"
          ]
        },
        {
          "copyFromDT":
"configuration/entityTypes/Location/attributes/StateProvince",
          "copyToCT": [

"configuration/entityTypes/Location/attributes/StateProvince"
          ]
        },
        {
          "copyFromDT":
"configuration/entityTypes/Location/attributes/SubAdministrativeArea",
          "copyToCT": [

"configuration/entityTypes/Location/attributes/SubAdministrativeArea"
          ]
        },
        {
          "copyFromDT":
"configuration/entityTypes/Location/attributes/Country",
          "copyToCT": [
            "configuration/entityTypes/Location/attributes/Country"
          ]
```

```
        },
        {
          "copyFromDT":
"configuration/entityTypes/Location/attributes/Zip",
          "copyToCT": [
            "configuration/entityTypes/Location/attributes/Zip"
          ],
          "attributes": [
            {
              "copyFromDT":
"configuration/entityTypes/Location/attributes/Zip/attributes/Zip5",
              "copyToCT": [

"configuration/entityTypes/Location/attributes/Zip/attributes/Zip5"
              ]
            },
            {
              "copyFromDT":
"configuration/entityTypes/Location/attributes/Zip/attributes/Zip4",
              "copyToCT": [

"configuration/entityTypes/Location/attributes/Zip/attributes/Zip4"
              ]
            }
          ]
        }
      ]
    },
    {
      "copyFromDT": "configuration/relationTypes/HasAddress",
      "copyToCT": "configuration/relationTypes/HasAddress",
      "filter": [
        {
          "attribute":
"configuration/relationTypes/HasAddress/attributes/AddressRank",
          "values": [
            "1",
            "2"
          ]
        }
      ],
      "attributes": [
        {
          "copyFromDT":
"configuration/relationTypes/HasAddress/attributes/AddressRank",
          "copyToCT": [

"configuration/relationTypes/HasAddress/attributes/AddressRank"
          ],
          "transformValue": [
```

```
            {
              "from": "1",
              "to": "0"
            },
            {
              "from": "2",
              "to": "0"
            }
          ]
        },
        {
          "copyFromDT":
"configuration/relationTypes/HasAddress/attributes/AddressType",
          "copyToCT": [

"configuration/relationTypes/HasAddress/attributes/AddressType"
          ]
        },
        {
          "copyFromDT":
"configuration/relationTypes/HasAddress/attributes/Phone",
          "copyToCT": [
            "configuration/relationTypes/HasAddress/attributes/Phone"
          ],
          "attributes": [
            {
              "copyFromDT":
"configuration/relationTypes/HasAddress/attributes/Phone/attributes/Acti
ve",
              "copyToCT": [

"configuration/relationTypes/HasAddress/attributes/Phone/attributes/Acti
ve"
              ]
            },
            {
              "copyFromDT":
"configuration/relationTypes/HasAddress/attributes/Phone/attributes/Type
",
              "copyToCT": [

"configuration/relationTypes/HasAddress/attributes/Phone/attributes/Type
"
              ],
              "transformValue": [
                {
                  "from": "Business",
                  "to": "Work"
                }
              ]
```

```
            },
            {
              "copyFromDT":
"configuration/relationTypes/HasAddress/attributes/Phone/attributes/Rank
",
              "copyToCT": [

"configuration/relationTypes/HasAddress/attributes/Phone/attributes/Rank
"
              ]
            },
            {
              "copyFromDT":
"configuration/relationTypes/HasAddress/attributes/Phone/attributes/Numb
er",
              "copyToCT": [

"configuration/relationTypes/HasAddress/attributes/Phone/attributes/Numb
er"
              ]
            }
          ]
        },
        {
          "copyFromDT":
"configuration/relationTypes/HasAddress/attributes/DEA",
          "copyToCT": [
            "configuration/relationTypes/HasAddress/attributes/DEA"
          ],
          "attributes": [
            {
              "copyFromDT":
"configuration/relationTypes/HasAddress/attributes/DEA/attributes/Number
",
              "copyToCT": [

"configuration/relationTypes/HasAddress/attributes/DEA/attributes/Number
"
              ]
            },
            {
              "copyFromDT":
"configuration/relationTypes/HasAddress/attributes/DEA/attributes/Expira
tionDate",
              "copyToCT": [

"configuration/relationTypes/HasAddress/attributes/DEA/attributes/Expira
tionDate"
              ]
            },
```

```
            {
              "copyFromDT":
"configuration/relationTypes/HasAddress/attributes/DEA/attributes/Status
",
              "copyToCT": [

"configuration/relationTypes/HasAddress/attributes/DEA/attributes/Status
"
              ]
            }
          ]
        }
      ]
    }
  ],
  "synchronizationConfig": {
    "dtSyncType": "REALTIME",
    "ctSyncType": "REALTIME",
    "entities": [
      {
        "action": "AUTOSUBSCRIBE",
        "types": [
          "configuration/entityTypes/HCP"
        ],
        "matchRules": [
          "configuration/entityTypes/HCP/matchGroups/PersonByMEAuto",
          "configuration/entityTypes/HCP/matchGroups/PersonByMESuspect2"
        ],
  "filter": "<filter>"
      }
    ],
    "relationTypes": [
      {
        "types": [
          "configuration/relationTypes/AffiliatedWith"
        ],
        "action": "COPY"
      }
    ]
  },
  "importRelationsConfig": {
    "defaultStrategy": "ALL",
    "strategyPerRelationType": []
  },
```

```
  "groupContributors": true,
  "supportEmail": "example@mail.com"
}
```

Tenant Subscription Events Configuration

This function defines some behavior of events processing.

Events Configuration Definition Json Object

```
{
  "crosswalkDeleted": {
    "defaultActionInCT": "IGNORE|SET_END_DATE|DELETE",
    "source": {
      "{sourceURI}": "INGORE|SET_END_DATE|DELETE",
      ...
    }
  },
  "entityDeleted": {
    "actionInCT": "IGNORE|ACTIVATE"
  },
  "entityModified": {
    "actionInCT": "IGNORE|ACTIVATE"
  },
  "entityUnmerged": {
    "actionInCT": "IGNORE|ACTIVATE"
  },
  "entityMerged": {
    "actionInCT": "IGNORE|ACTIVATE"
  },
  "entityCreated": {
    "actionInCT": "IGNORE|ACTIVATE",
    "entityTypes": ["entityTypeUri1", "entityTypeUri2"]
  }
  "entityMatch": {
    "action" : "IGNORE|ACTIVATE"
  },
  "relationCreated": {
    "actionInCT": "IGNORE|ACTIVATE"
  },
  "relationModified": {
    "actionInCT": "IGNORE|ACTIVATE"
  },
  "relationDeleted": {
    "actionInCT": "IGNORE|ACTIVATE"
  }
}
```

Parameters

Name	Required	Default Value	Description
`crosswalkDeleted`	No	`{"defaultActionInCT" : "IGNORE"}`	This event can happen when the entity or relation is deleted or updated in DT. It includes two optional sections: • `"entityCrosswalkDeleted"` (or `"crosswalkDeleted"` deprecated value) for entities • `"relationCrosswalkDeleted"` for relations `"defaultActionInCT"` - Default reaction to this event. It can be: • `IGNORE` - do nothing. • `SET_END_DATE` - end-date crosswalk. • `DELETE` - delete crosswalk. You can configure reaction to this event per source. For this, you can use the `"source"` parameter. Example: **Example of EntityCrosswalkDeleted Event** `{ "entityCrosswalkDeleted": { "defaultActionInCT": "DELETE", "source": { "configuration/sources/FB": "SET_END_DATE", "configuration/sources/TWEETER": "DELETE", "configuration/sources/LNKD": "IGNORE" } } }`
`entityDeleted`	No	`{"actionInCT" : "IGNORE"}`	This event can occur if an entity is deleted from: `"actionInCT"` - Default reaction to this event. It can be: • `IGNORE` - do nothing. • `ACTIVATE` - process this event. All crosswalks which have been imported to customer tenant are processed in accordance with the `entityCrosswalkDeleted` configuration.
`entityModified`	No	`{ "actionInCT" : "ACTIVATE" }`	This event can occur if an entity is updated. `"actionInCT"` - Default reaction to this event. It can be: • `IGNORE` - do nothing. • `ACTIVATE` - process this event. If the customer tenant entities have been subscribed on the updated entity they are updated; otherwise, entities are imported or autosubscribed, if this is possible. Allowed entity types and match rules are to be taken from `SynchronizationConfig` if one is present.

`entityUnmerged`	No	`{"actionInCT" : "IGNORE"}`	This event can happen if an entity is unmerged in data tenant. `"actionInCT"` - Default reaction to this event. It can be: • `IGNORE` - do nothing. • `ACTIVATE` - process this event. All crosswalks which exist in the customer tenant are marked as `"UNMERGED"` (You can see it in the `externalInfo` field of the crosswalk). This event is processed in accordance with the `entityCrosswalkDeleted` configuration.
`entityMerged`	No	`{"actionInCT" : "IGNORE"}`	This event can occur if some entities merge. However, this event is a part of the `entityModified` event. `"actionInCT"` - Default reaction to this event. It can be: • `IGNORE` - do nothing. However, all subscribed entities are updated in accordance with the `entityModified` event configuration. • `ACTIVATE` - process this event. Entities in the customer tenant which have been subscribed on merged entities in the data tenant are merged in the customer tenant if they haven't been marked as not a match. This event can be `"ACTIVATE"` only if the `entityModified` event has been configured.
`entityCreated`	No	`{"actionInCT" : "IGNORE",}`	This event can occur if a new entity has been created. `"actionInCT"` - Default reaction to this event. It can be: • `IGNORE` - do nothing. • `ACTIVATE` - process this event. If the customer tenant entities have been subscribed on the updated entity they are updated; otherwise, entities are imported or `autosubscribed` if that is possible. Allowed entity types and match rules are to be taken from `SynchronizationConfig` if this is present. If `SynchronizationConfig` is not present, then the default entity types from the `entityCreated` configuration are taken. **[DEPRECATED:** Use **Tenant Subscription Synchronization Configuration (**DT entity types to process). Example of the `EntityCreated` configuration: **Example of EntityCreated Config** `{ "entityCreated": { "actionInCT": "ACTIVATE" } }`
`entityMatch`	No	`{"action" : "ACTIVATE"}`	This is the configuration parameter for all events above. Since 2017.3, `entityMatch` is always `"ACTIVATE"`. • `ACTIVATE` - process all events, update match tables, and recalculate the number of potential matches for entities.

`relationCreated`	No	`{"actionInCT" : "ACTIVATE"}`	This event can occur if a new relation has been created. `"actionInCT"` - Default reaction to this event. It can be: • `IGNORE` - do nothing. • `ACTIVATE` - process this event. **Example of RelationCreated Config** `{ "relationCreated": { "actionInCT": "ACTIVATE" } }`
`relationModified`	No	`{"actionInCT" : "ACTIVATE"}`	This event can occur if a relation has been updated in the DT. `"actionInCT"` - Default reaction to this event. It can be: • `IGNORE` - do nothing. • `ACTIVATE` - process this event. DTSS updates the corresponding relation in the CT.
`relationDeleted`	No	`{"actionInCT" : "ACTIVATE"}`	This event can occur if a relation has been removed from DT. `"actionInCT"` - Default reaction to this event. It can be: • `IGNORE` - do nothing. • `ACTIVATE` - process this event. DTSS removes the corresponding relation from the CT it has imported.

If `dtssQueue` is configured in the streaming configuration for this tenant, then all events are processed in real time (based on the configuration provided above). Configure streaming and `dtssQueue` for the tenant. An option is available to delay events processing events and process all events using a special request. To process delayed events, create a pull event task.

Note: Delayed events are processed based on the event configuration for the subscription. Use the Synchronization Configuration procedure for advanced events processing.

Tenant Subscription Synchronization Configuration

This function defines some behavior of synchronization processing.

Synchronization Config Definition Json Object

```
{
  "dtSyncType": "PULL",
  "ctSyncType": "REALTIME",
  "entities": [
    {
      "types": [
        "configuration/entityTypes/HCP"
      ],
      "action": "AUTOSUBSCRIBE",
      "matchRules": [
        "<matchRule1>",
```

```
        "<matchRule2>"
      ],
      "filter": "<filter1>",
      "connections": [
         {
             "onCreateOnly": false,
             "types": [
                "configuration/relationTypes/Contractor"
             ],
             "strategy":
"ALL|IMPORT_IF_ENTITIES_SUBSCRIBED|IMPORT_IF_START_ENTITY_SUBSCRIBED|IMP
ORT_IF_END_ENTITY_SUBSCRIBED"
         }
      ],
      "thresholds": {
        "default": 2,
        "extendSubscription": "false",
        "custom": [
          {
            "matchRules": [

"configuration/entityTypes/HCP/matchGroups/PersonByMESuspect2"
            ],
            "threshold": 2,
            "extendSubscription": "false"
          }
        ]
      }
    },
    {
      "types": [
        "configuration/entityTypes/HCP"
      ],
      "action": "MANUAL_MATCH",
      "matchRules": [
        "<matchRule3>"
      ],
      "filter": "<filter>"
    },
    {
      "types": [
        "configuration/entityTypes/HCO"
      ],
      "action": "COPY"
    }
  ],
  "relations": [
    {
      "types": [
        "configuration/relationTypes/AffiliatedWith"
```

```
        ],
        "action": "COPY"
```

```
        }
    ]
}
```

Parameters

Name	Re-quired	Description
`dtSync Type`	No	Describes the way synchronization should be performed in the Data Tenant. It can be: • `PULL` - enable only Pull Task synchronization. • `REALTIME` - enable only real-time synchronization. If not set, then both Real-Time and Pull Task are enabled.
`ctSync Type`	No	Describes the way synchronization should be performed in Customer Tenant. It can be: • `PULL` - enable only Pull Task synchronization. • `REALTIME` - enable only real-time synchronization. If not set, then both Real-Time and Pull Task are enabled.

`entities`	No	Describes the way certain entity types should be processed. If the field is present, then only described entity types are processed.
		
		Contains the fields:
		
		• "types" - a list of types to which to apply the configuration. The way the parameter is used depends on the action being performed.
		• "action" - describes the mode by which an entity from DT should be imported. It can be:
		  ◦ AUTOSUBSCRIBE - import only if the DT entity has got a match with a CT entity (the DT entity's type must be present in the "types" list). The "matchRules" field can be used to describe the match rules set.
		  ◦ MANUAL_MATCH - this section defines match rules for each type, which is used during search of potential matches for an entity. If no match rules are specified for some type, then all match rules defined in data tenant configuration for a type are used. Otherwise, the union of match rules specified here along with match rules from the AUTOSUBSCRIBE section (per type) are applied.
		  ◦ COPY - just import the entity from the DT (the DT entity's type must be present in the "types" list).
		• "matchRules" - a list of match rules that is used to find matches. This field is used only if the AUTOSUBSCRIBE mode is set.
		
		Since 2017.3, if match rules attributes are missing in mappings section (not present neither in autogenerated nor in manually specified), then the "Check subscription mapping" section would be equal to WARN with enumeration of such missing attributes upon verifying subscription via the {DTSSURL}/subscriptions/verifySubscription endpoint. This type of subscription is considered invalid if the strictMappingCheck subscription parameter is set to true.

- "filter" - Reltio Search format filter. It can be used to select a subset of data to process.
- "connections" - describes the connections to be imported to the CT along with current entity (acceptable for AUTOSUBSCRIBE and COPY actions)
  - "types" - a list of relation types to copy to the CT.
  - "onCreateOnly" - if true, than connections are copied to CT on the ENTITY_CREATED event only.
  - "strategy" - strategy to decide which connection will be imported to CT.
    - ALL - import without condition (this is the default strategy. It is added automatically if omit).
    - IMPORT_IF_ENTITIES_SUBSCRIBED - only if CT is subscribed on DT relation's both startEntity and endEntity entities.
    - IMPORT_IF_START_ENTITY_SUBSCRIBED - only if CT is subscribed on at least DT relation's startEntity entity.
    - IMPORT_IF_END_ENTITY_SUBSCRIBED - only if CT is subscribed on at least DT relation's endEntity entity.
- "thresholds" - thresholds description. This field is used only if the AUTOSUBSCRIBE mode is set. A bucket below is a set of entities aggregated by a set of corresponding match rules.
  - "default" - a threshold for a default bucket (more about the threshold below).
  - "threshold" - a limited number of DT entities that a bucket can contribute at a single autosubscribe session (default: 1).
  - "extendSubscription" - states if a bucket can extend existing DT-CT entity subscription (default: false).
  - "matchRules" - a list of match rules for which the threshold configuration should be used.

```
"entities": [
  {
    "types": [
      "configuration/entityTypes/HCP"
    ],
    "action": "AUTOSUBSCRIBE",
    "matchRules": [
      "someMatchRule"
    ],
    "connections": [
        {
          "types": [
            "configuration/relationTypes/HasHealthCareRole"
          ],
          "onCreateOnly": false,
          "strategy": "ALL"
        }
      ]
    },
    {
      "types": [
        "configuration/entityTypes/HCO"
      ],
      "action": "AUTOSUBSCRIBE",
      "matchRules": [
        "configuration/entityTypes/HCO/matchGroups/HCObyIMSID"
      ],
      "connections": [
        {
          "types": [
            "configuration/relationTypes/Managed",
            "configuration/relationTypes/Leased"
          ],
          "onCreateOnly": false,
          "strategy": "IMPORT_IF_ENTITIES_SUBSCRIBED"
        }
      ]
    }
  ]
}
```

"entities": [{ "types": ["configuration/entityTypes/HCP"], "action": "MANUAL_MATCH", "matchRules": ["someOtherMatchRule"], "filter": "<filter>" }]"entities": [{ "types": ["configuration/entityTypes/HCO"], "action": "COPY" }]

`relations`	No	Describes the way certain relation types should be processed. If the field is present, then only described relation types are processed. Contains the fields: • `"types"` - a list of types to which to apply the configuration. The way the parameter is used depends on the action being performed. • `"action"` - describes the mode by which an entity from DT should be imported. It can be: • `COPY` - just import the relation from DT (the DT relation's type must be present in the `"types"` list). "relations": [{ "types": ["configuration/relationTypes/AffiliatedWith"], "action": "COPY" }]

Tenant Subscription Transform Configuration

Transform configuration is used to setup advanced behavior of data transformation capabilities.

Note: This is optional structure and is not enabled by default.

Configuration Parameter	Default Value	Description
`goldenRecordConfiguration`	N/A	Advanced transformation behavior subscription with GRC (`bringGoldenRecord=true only`).

GOLDEN RECORD CONFIGURATION

Configuration Parameter	Default Value	Description
`enddatingEnabled`	`false`	Enable enddating of golden record crosswalks in the case when you want `bringGoldenRecord=true` only. The value can be: • `true`: feature is on • `false`: feature is off (default) If enddating for GRC is on: • DT 1 enddated crosswalk and existing in CT -> CT crosswalk enddated; attributes are not removed • DT 1 enddated crosswalk and no GRC in match in CT -> no subscribe; no update (nothing to enddate) • DT 1 enddated crosswalk and no match in CT -> no import; no update (nothing to enddate) • DT 2 crosswalks -> legacy behavior If enddadting for GRC is off: • DT 1 enddated crosswalk and existing in CT -> CT crosswalk NOT enddated; attributes are removed • DT 1 enddated crosswalk and no match in CT -> import GRC only (without attributes) **Note:** The CT crosswalk enddate depends on the DT crosswalk enddate, as shown in the table below for the scenario and its expected results for crosswalk delete date synchronization.

enddatingReferenceEntitiesEnabled	false	If set to true, it end-dates reference entities if the Golden Record crosswalk in CT is end-dated due to the corresponding crosswalk being end-dated in DT. If the goldenRecordConfiguration parameter is set to true, then we do not enddate crosswalks in DT. This is true for both the following cases: — multiple DT crosswalks — single DT crosswalk However, in case of a single DT crosswalk, we do not transfer attributes of end-dated crosswalks.

Crosswalk Delete Date Synchronization

DT crosswalk delete date	CT crosswalk delete date
From null to sysdate	sysdate
From null to future date	future date
From null to past date	sysdate
From sysdate to null	null
From future date to null	null
From past date to null	null

Example

Example of relations config 1

```
"transformConfiguration": {
  "goldenRecordConfiguration": {
    "enddatingEnabled": true
  }
}
```

Note: This behavior is applicable for entities and relationships.

Tenant Subscription Removal of startDate/endDate in Relations

Configuration Parameter	Default Value	Description
`relationsSyncWithNullActiveness`	off	You need to turn this on to ensure that when the start date and end date of relations are removed in the Data Tenant, they are removed in the customer tenant as well.

Tenant Subscription Tasks Configuration

Tasks configuration is used to setup advanced tasks behavior and scheduling.

Note: This is optional structure and is not enabled by default.

Configuration Parameter	Default Value	Description
`onFinish`	N/A	Setting up on finish policies for tasks.

ONFINISH CONFIGURATION

Configuration Parameter	Default Value	Description
consistencyReport	N/A	Automatic consistency report (automatic start of ConsistencyReportTask).

CONSISTENCYREPORT CONFIGURATION

Configuration Parameter	Default Value	Description
`afterTasks`	N/A	Tasks selected for auto consistency reporting. Once a selected task finishes a ConsistencyReportTask is started automatically. List of supported tasks: CopyTask SubscribeTask SyncTask Example: **Example** `"afterTasks": { "manual_copy": true, "manual_subscribe": false, "manual_sync": false }`
`emails`	N/A	Email recipients for the report. The list is used to setup the emails field of auto-created ConsistencyReportTask.

Example

Example of ConsistencyReport Configuration

```
"tasksConfiguration": {
  "onFinish": {
    "consistencyReport": {
      "afterTasks": {
        "manual_copy": true,
        "manual_subscribe": false,
        "manual_sync": false
      },
      "reportEmails": [
        "default.report.test.email@reltio.com"
      ]
    }
  }
}
```

Validate Tenant Subscription

Validate the tenant subscription configuration by using a special validation endpoint.

TENANT SUBSCRIPTION VERIFICATION

There is a set of special endpoints to verify data tenant configuration, customer tenant configuration, and subscription configuration. To verify any configuration DTSS performs several checks and returns the result of all executed checks.

Data Tenant Configuration Verification

There are several checks for data tenant configuration:

Oauth authorization check: checks if DTSS is able to authorize on configured Oauth server using configured credentials.

API System Config check: compares sdk configuration of tenant with system configuration of API endpoint (they must be equal).

API search check: checks if DTSS is able to search data through configured api url.

JMS configuration check: checks if DTSS is able to connect to configured JMS broker and authorize on it. It uses `StreamingConfig` from the data tenant configuration, and performs this check even if `jms_enabled` is set to `false`.

Cassandra configuration check: checks if DTSS is able to access the configured Cassandra environment.

DataTenant search check: checks if DTSS is able to search entities in the Data Tenant.

Request

```
GET {ApplicationURL}/tenants/dataTenants/{tenantId}/verifyTenant
```

Parameters

	Name	Required	Description
Headers	Authorization	yes	Information about authentication access token in format `"Bearer <accessToken>"` (see details in Authentication API).

Response

```
[
  {
    "checkName": "Oauth endpoint check for data tenant
olegdt",
    "result": {
      "status": "SUCCESS"
    }
  },
  {
    "checkName": "API System config check for data tenant
olegdt",
    "result": {
      "status": "SUCCESS"
    }
  },
  {
    "checkName": "Reltio Platform Tenant exists check for
data tenant olegdt",
    "result": {
      "status": "SUCCESS"
    }
  },
  {
    "checkName": "Search number of entities through API for
data tenant olegdt",
    "result": {
      "status": "SUCCESS",
      "message": "Found 41 entities"
    }
  },
  {
    "checkName": "JMS broker check for data tenant olegdt",
    "result": {
      "status": "SUCCESS"
    }
  },
  {
    "checkName": "Check cassandra for customer tenant olegdt"
,
    "result": {
      "status": "SUCCESS"
    }
  },
  {
    "checkName": "Search entities in data tenant olegdt",
    "result": {
      "status": "SUCCESS",
      "message": "Found 41 entities in data tenant"
    }
  }
]
```


Customer Tenant Subscription Verification

There are several checks for the Data Tenant configuration:

Oauth authorization check: checks if DTSS is able to authorize on configured Oauth server using configured credentials.

API search check: checks if DTSS is able to search data through configured api url.

Cassandra configuration check: checks if DTSS is able to access configured the cassandra environment.

Request

```
GET {ApplicationURL}/tenants/customerTenants/{tenantId}/verifyTenant
```

Parameters

	Name	Required	Description

Headers	Authorization	yes	Information about authentication access token in format `"Bearer <accessToken>"` (see details in Authentication API).

Response

```
[
  {
    "checkName": "Oauth endpoint check for customer tenant
olegct",
    "result": {
      "status": "SUCCESS"
    }
  },
  {
    "checkName": "API System config check for customer tenant
olegct",
    "result": {
      "status": "SUCCESS"
    }
  },
  {
    "checkName": "Reltio Platform Tenant exists check for
customer tenant olegct",
    "result": {
      "status": "SUCCESS"
    }
  },
  {
    "checkName": "Search number of entities through API for
customer tenant olegct",
    "result": {
      "status": "SUCCESS",
      "message": "Found 51 entities"
    }
  },
  {
    "checkName": "Check cassandra for data tenant olegct",
    "result": {
      "status": "SUCCESS"
    }
  }
]
```

Subscription Verification

To verify subscription configuration DTSS performs verification of Data Tenant and Customer Tenant configuration, and performs two additional checks. It checks if DTSS created two column families with the names `CT_<CustomerTenant>_DT_<DataTenant>` and `DT_<DataTenant>_CT_<CustomerTenant>`.

Request

```
GET
{ApplicationURL}/subscription/verifySubscription
```

Parameters

	Name	Required	Description
Headers	Authorization	yes	Information about authentication access token in format `"Bearer <accessToken>"` (see details in Authentication API).
	DataTenant	yes	Source Data Tenant id.
	CustomerTenant	yes	Target Customer Tenant id.

Response

```
[
  {
```

```
    "checkName": "Oauth endpoint check for customer tenant
olegct",
    "result": {
      "status": "SUCCESS"
    }
  },
  {
    "checkName": "API System config check for customer tenant
olegct",
    "result": {
      "status": "SUCCESS"
    }
  },
  {
    "checkName": "Reltio Platform Tenant exists check for
customer tenant olegct",
    "result": {
      "status": "SUCCESS"
    }
  },
  {
    "checkName": "Search number of entities through API for
customer tenant olegct",
    "result": {
      "status": "SUCCESS",
      "message": "Found 51 entities"
    }
  },
  {
    "checkName": "Check cassandra for data tenant olegct",
    "result": {
      "status": "SUCCESS"
    }
  },
  {
    "checkName": "Oauth endpoint check for data tenant olegdt",
    "result": {
      "status": "SUCCESS"
    }
  },
  {
    "checkName": "API System config check for data tenant olegdt"
,
    "result": {
      "status": "SUCCESS"
    }
  },
  {
    "checkName": "Reltio Platform Tenant exists check for data
tenant olegdt",
    "result": {
      "status": "SUCCESS"
    }
  },
  {
    "checkName": "Search number of entities through API for data
tenant olegdt",
    "result": {
      "status": "SUCCESS",
      "message": "Found 41 entities"
    }
  },
  {
    "checkName": "JMS broker check for data tenant olegdt",
    "result": {
      "status": "SUCCESS"
    }
  },
  {
    "checkName": "Check cassandra for customer tenant olegdt",
    "result": {
      "status": "SUCCESS"
```

```
    }
  },
  {
    "checkName": "Search entities in data tenant olegdt",
    "result": {
      "status": "SUCCESS",
      "message": "Found 41 entities in data tenant"
    }
  },
  {
    "checkName": "Check column family ct_olegct_dt_olegdt",
    "result": {
      "status": "SUCCESS"
    }
  },
  {
    "checkName": "Check column family dt_olegdt_ct_olegct",
    "result": {
      "status": "SUCCESS"
    }
  },
  {
    "checkName": "Check subscription mapping",
    "result": {
      "status": "SUCCESS"
    }
  }
]
```

Get invalid subscriptions

Get a list of subscriptions that have not passed validation (for example, the wrong configuration).

Request

```
GET {DTSSURL}/admin/subscriptionHealthCheck
```

Parameters

	Name	Required	Description
Headers	Authorization	yes	Information about authentication access token in format `"Bearer <accessToken>"` (see details in Authentication API).

Response

```
[
  {
    "dataTenantId
": "data0",
    "customerTena
ntId": "customer0"
  },
  {
    "dataTenantId
": "data1",
    "customerTena
ntId": "customer1"
  }
]
```

Email Notification

DTSS will notify Subscription/Tenant/OAuthInstance owner in case of a failed validation. Email address can be set up through the *support email* fi eld (refer to Data Tenant Registration, Customer Tenant Registration, Tenant Subscription, and OAuthInstance for more information).

Configuration

Name	Default	Description

`notifications.emails.max`	`100`	Maximum number of emails that are to be processed simultaneously by DTSS.
`notifications.emails.baseList`		Default addresses to use in mailing.
`notifications.items.max`	`100`	Maximum number of failures that can be reported in a *notifications.timeout* for a single email.
`notifications.timeout`	`3600000ms` (1 hour)	Mailing timeout.
`notifications.enabled`	`true`	Feature toggler.

Get/Filter Tenant Subscription

HTTP METHOD	GET
URL	`{DTSSURL}/subscriptions?useCache={boolean}&validate={boolean}`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}` DataTenant: `{dataTenant}` CustomerTenant: `{customerTenant}`

Parameters

Name	Required	Default	Description
oauth_token	Yes		Authorization token of the user.
`dataTenantId`	No		Source Data Tenant id. If you specify this header, all subscriptions are filtered by the Data Tenant Id. However, you should have access to the requested subscription. All subscriptions are filtered according according to your access level.
`customerTenantId`	No		Target Customer Tenant id. If you specify this header, all subscriptions are filtered by the Customer Tenant Id. However, you should have access to the requested subscription. All subscriptions are filtered according to your access level.
`useCache`	No	`false`	Forces to use cache.
`validate`	No	`false`	Whether subscriptions found should be verified. Verification result is put into the `validationResult` subscription field.

Delete Tenant Subscription

HTTP METHOD	DELETE
URL	`{DTSSURL}/subscriptions`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}` DataTenant: `{dataTenant}` CustomerTenant: `{customerTenant}`

Parameters

Name	Required	Description
`oauth_token`	Yes	Authorization token of theuser.
`dataTenantId`	Yes	Source Data Tenant id.

customerTenantId	Yes	Target Customer Tenant id.

Change groupContributors Configuration Parameter

Use this special endpoint to change the groupContributors configuration parameter. It allows you to change the value only from False to True. For this change, a special migration task is started to update contributors for the existing customer tenant.

HTTP METHOD	POST
URL	{DTSSURL}/subscriptions/groupContributors?value={value}
HEADERS	Content-Type: application/json Authorization: Bearer {oauth_token} DataTenant: {dataTenant} CustomerTenant: {customerTenant}

Parameters

Name	Required	Description
oauth_token	Yes	Authorization token of the user.
dataTenant	Yes	Source Data Tenant id.
customerTenant	Yes	Target Customer Tenant id.
value	Yes	New value to be set.

Response Example

```
{
    "status": "success",
    "result": "Value changed. Task id is
fab66c05-0b51-46a7-864d-bb0ce22ee0bc. To check task status use:
http://test.reltio.com/reltio/tasks/fab66c05-0b51-46a7-864d-bb0ce22ee0bc
"
}
```

Entity Subscription

Import Entity from Data Tenant to Customer Tenant

HTTP METHOD	POST
URL	{DTSSURL}/entities/_import?dataEntityUri={uri}
HEADERS	Content-Type: application/json Authorization: Bearer {oauth_token} DataTenant: {dataTenant} CustomerTenant: {customerTenant}

Parameters

Name	Required	Description
`dataTenant`	Yes	Source Data Tenant id.
`customerTenant`	Yes	Target Customer Tenant id.
`uri`	Yes	Entity Uri from Data Tenant.. Example: `"00iPLVp"` or `"entities/00iPLVp"`.

Response Example

```
{
    "success" : true|false,
 "errors" : [<array of errors>],
 "uris" : [<uri of entity from customer tenant which is result of
import>]
}
```

Merge Entity

Use this function to create mapping between the entity from the Data Tenant and the entity from the Customer Tenant.

HTTP METHOD	`POST`
URL	`{DTSSURL}/entities/_merge?dataEntityUri={dtUri}&customerEntityUri={ctUri}`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}` DataTenant: `{dataTenant}` CustomerTenant: `{customerTenant}`

Parameters

Name	Required	Description
`dataTenant`	Yes	Source Data Tenant id.
`customerTenant`	Yes	Target Customer Tenant id.
`dtUri`	Yes	Entity Uri from the Data Tenant. Example: `"00iPLVp"` or `"entities/00iPLVp"`.
`ctUri`	Yes	Entity Uri from the Customer Tenant. Example: `"00iPLVp"` or `"entities/00iPLVp"`.

Response Example

```
{
    "success" : true|false,
 "errors" : [<array of errors>],
 "uris" : [<uri of entity from customer tenant which is result of
import>]
}
```

UnMerge Entity

Use this function to delete Information imported from the Data Tenant entity.

HTTP METHOD	POST
URL	{DTSSURL}/entities/_unmerge
HEADERS	Content-Type: application/json Authorization: Bearer {oauth_token} DataTenant: {dataTenant} CustomerTenant: {customerTenant}
BODY	**Example** [{ "value" : "{value}", "type" : "{type}", "sourceTable" : "sourceTable" }]

Parameters

Name	Required	Description
customerTenant	Yes	Target Customer Tenant id.
dataTenant	Yes	Target Data Tenant id.
value	Yes	Crosswalk value of crosswalk which has been imported to the Customer tenant.
type	Yes	Crosswalk type of crosswalk which has been imported to the Customer Tenant.
sourceTable	No	Source table of crosswalk which has been imported to the Customer Tenant.

Response Example

```
{
    "status": "success|error",
 "message" : "<some message which contains valuable information>"
}
```

Import Relations From Data Tenant to Customer Tenant

Import Single Relation

HTTP METHOD	POST
URL	`{DTSSURL}/relations/_importRelation?dataRelationUri={dataRelationUri}`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}` DataTenant: `{dataTenant}` CustomerTenant: `{customerTenant}`

Parameters

Name	Required	Description
`dataTenant`	Yes	Source Data Tenant id.
`customerTenant`	Yes	Target Customer Tenant id.
`dataRelationUri`	Yes	Relation Uri from the Data Tenant. Example: `"00iPLVp"` or `"relations/00iPLVp"`.

Response Example

```
{
    "importedEntities": [
        {
            "success": true,
            "errors": [],
            "uris": [
                "entities/00tgHZZ"
            ],
            "status": "success"
        },
        {
            "success": true,
            "errors": [],
            "uris": [
                "entities/00tgLpp"
            ],
            "status": "success"
        }
    ],
    "importedRelations": [
        {
            "success": true,
            "errors": [],
            "uris": [
                "relations/00MBUlV"
            ],
            "status": "success"
        }
    ]
}
```


Import Multiple Relations

HTTP METHOD	POST
URL	`{DTSSURL}/relations/_importRelations`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}` DataTenant: `{dataTenant}` CustomerTenant: `{customerTenant}`
BODY	`["dataRelationUri1", "dataRelationUri2"]`

Parameters

Name	Required	Description
`dataTenant`	Yes	Source Data Tenant id.
`customerTenant`	Yes	Target Customer Tenant id.

Response Example

```
{
    "importedEntities": [
        {
            "success": true,
            "errors": [],
            "uris": [
                "entities/00tgHZZ"
            ],
            "status": "success"
        },
        {
            "success": true,
            "errors": [],
            "uris": [
                "entities/00tgLpp"
            ],
            "status": "success"
        }
    ],
    "importedRelations": [
        {
            "success": true,
            "errors": [],
            "uris": [
                "relations/00MBUlV"
            ],
            "status": "success"
        }
    ]
}
```

Mark Entities as `notMatch`

Request

```
POST
{DTSSURL}/entities/_notMatch?dataEntityUri={uri}&customerEntityUri={uri}
```

Parameters

	Name	Required	Description
HEADERS	DataTenant	Yes	Source Data Tenant id.
	CustomerTenant	Yes	Target Customer Tenant id.

Response

```
{
    "status": "success"
}
```

Delete notMatch

Request

```
DELETE
{DTSSURL}/entities/_notMatch?dataEntityUri={uri}&customerEntityUri={uri}
```

Parameters

	Name	Required	Description
HEADERS	DataTenant	Yes	Source Data Tenant id.
	CustomerTenant	Yes	Target Customer Tenant id.

Response

```
{
    "status": "success"
}
```

Get entity notMatch

Use this function to search for potential matches provided in JSON array entities provided in the Data Tenant (for the subscribed Customer Tenant).

Request

```
GET
{DTSSURL}/entities/_notMatch?returnObjects={true|false}&customerEntityUr
i={uri}
```

Parameters

	Name	Re-quired	Description	Examples
Header	Customer Tenant	Yes	Customer Tenant id.	-

	`DataTenant`	Yes	Data Tenant id.	-
Uri Params	`customerEntityUri`	Yes	Entity Uri from the Customer Tenant.	
	`return Objects`	No	Specifies if response should contain entities. Default value is `false`.	

BODY	-	Yes	A JSON array of entities to be matched with entities in the Data Tenant.
			
			IMPORTANT! Entities should not have a `"label"` field in their body, or they will be rejected.
				[{
				"uri": "entities/0000I77",
				"type": "configuration/entityTypes/Individual",
				"attributes": {
				"FirstName": [{
				"type": "configuration/entityTypes/Individual/attributes/FirstName",
				"ov": true,
				"value": "Ernest",
				"uri": "entities/0000I77/attributes/FirstName/241"
				}],
				"MiddleName": [{
				"type": "configuration/entityTypes/Individual/attributes/MiddleName",
				"ov": true,
				"value": "E.",
				"uri": "entities/0000I77/attributes/MiddleName/1vl"
				}]
				}
				}]

Request Example 1

```
GET {DTSSURL}/entities/_notMatch?returnObjects=false&customerEntityUri=002Uk6b
Headers: Authorization: Bearer 204938ca-2cf7-44b0-b11a-1b4c59984512, Content-Type: application/json
```

Response Example 1

```
GET {DTSSURL}/entities/_notMatch?returnObjects=false&customerEntityUri=002Uk6b
Headers: Authorization: Bearer 204938ca-2cf7-44b0-b11a-1b4c59984512

[
    {
        "uri": "entities/001Ck6b"
    },
    {
        "uri": "entities/000j2p7",
        "updatedTime": 1393254467578,
        "updatedBy": "User"
    }
]
```

Request Example 2

```
GET {DTSSURL}/entities/_notMatch?returnObjects=true&customerEntityUri=002Uk6b
Headers: Authorization: Bearer 204938ca-2cf7-44b0-b11a-1b4c59984512, Content-Type: application/json
```

Response Example 2

```
GET {DTSSURL}/entities/_notMatch?returnObjects=true&customerEntityUri=002Uk6b
Headers: Authorization: Bearer 204938ca-2cf7-44b0-b11a-1b4c59984512, Content-Type: application/json

[
    {
        "object": {
            "uri": "entities/001Ck6b",
            "type": "configuration/entityTypes/HCP",
            ....
        },
         "uri": "entities/001Ck6b",
        "updatedTime": 1393254467578,
        "updatedBy": "User"
    },
    {
        "object": {
            "uri": "entities/000j2p7",
            "type": "configuration/entityTypes/HCP",
            ....
        },
        "uri": "entities/000j2p7",
        "updatedTime": 1393254467578,
        "updatedBy": "User"
    }
]
```

Potential Matches Search in DT

Use this function to search for potential matches of JSON array entities provided in the Data Tenant.

Note: Match rules defined in the `MANUAL_MATCH` section of the Synchronization Configuration are used for search (if the section exists).

Request

```
POST {DTSSURL}/entities/_matches

or

POST {DTSSURL}/entities/_matchesForCtEntities
```

Parameters

	Name	Re-quired	Description	Examples
Headers	`Customer Tenant`	Yes	Customer Tenant id.	-
	`DataTenant`	Yes	Data Tenant id.	-

Uri Params	skipAll ImportedEntities	No	If `true`, do not return any potential matches from the Data Tenant that are subscribed to by any entity in the customer tenant. Otherwise, return all potential matches from the data tenant regardless of whether or not each potential match is already subscribed to by an entity in the Customer Tenant. Default value is `false`.	`POST {DTSSURL}/entities/_matches?skipAllImportedEntities=true`

BODY	-	Yes	A JSON array of entities to be matched with entities in the Data Tenant.
			
			IMPORTANT! Entities should not have a `"label"` field in their body, or they will be rejected.
				["entities/0000I77"][{
				"uri": "entities/0000I77",
				"type": "configuration/entityTypes/Individual",
				"attributes": {
				"FirstName": [{
				"type": "configuration/entityTypes/Individual/attributes/
				FirstName",
				"ov": true,
				"value": "Ernest",
				"uri": "entities/0000I77/attributes/FirstName/241"
				}],
				"MiddleName": [{
				"type": "configuration/entityTypes/Individual/attributes/
				MiddleName",
				"ov": true,
				"value": "E.",
				"uri": "entities/0000I77/attributes/MiddleName/1vl"
				}]
				}
				}]

Response

The Response should contain a JSON array of potentially matched entities from the Data Tenant without entities that are marked as not a match.

```
[{
    "index": 0,
    "object": {

"configuration/entityTypes/Individual/matchGroups/PersonByFullNameInES":
[{
            "uri": "entities/0000I77",
            "type": "configuration/entityTypes/Individual",
            "createdBy": "admin",
            "createdTime": 1369070358438,
            "updatedTime": 1369070358438,
            "startDate": 35154000000,
            "roles": ["configuration/roles/Client"],
            "attributes": {
                "FirstName": [{
                    "type":
"configuration/entityTypes/Individual/attributes/FirstName",
                    "ov": true,
                    "value": "Ernest",
                    "uri": "entities/0000I77/attributes/FirstName/241"
                }],
                "LastName": [{
                    "type":
"configuration/entityTypes/Individual/attributes/LastName",
                    "ov": true,
                    "value": "Fusco",
                    "uri": "entities/0000I77/attributes/LastName/2CH"
                }]
 }
.....
}]
```

Potential Matches Search in CT

Use this function to search for potential matches of JSON array entities provided in the Customer Tenant.

Request

```
POST {DTSSURL}/entities/_matchesForDtEntities
```

Parameters

	Name	Re-quired	Description	Examples
HEADERS	Customer Tenant	Yes	Customer Tenant id.	-

	DataTenant	Yes	Data Tenant id.	-
Uri parameters	skipAll Imported Entities	No	If `true`, do not return any potential matches from the Data Tenant that are subscribed to by any entity in the Customer Tenant. Otherwise, return all potential matches from the Data Tenant regardless of whether or not each potential match is already subscribed to by an entity in the Customer Tenant. Default value is `false`.	`POST {DTSSURL}/entities/_matchesForDtEntities?skipAllImportedEntities=true`

BODY	-	Yes	A JSON array of entities to be matched with entities from the Customer Tenant.	["entities/0000I77"][{
				"uri": "entities/0000I77",
				"type": "configuration/entityTypes/Individual",
				"attributes": {
				"FirstName": [{
				"type": "configuration/entityTypes/Individual/attributes/FirstName",
				"ov": true,
				"value": "Ernest",
				"uri": "entities/0000I77/attributes/FirstName/241"
				}],
				"MiddleName": [{
				"type": "configuration/entityTypes/Individual/attributes/MiddleName",
				"ov": true,
				"value": "E.",
				"uri": "entities/0000I77/attributes/MiddleName/1vl"
				}]
				}
				}]

Response

The response should contain a JSON array of potentially matched entities from the Customer Tenant without entities that are marked as not a match.

```
[{
    "index": 0,
    "object": {

"configuration/entityTypes/Individual/matchGroups/PersonByFullNameInES":
[{
            "uri": "entities/0000I77",
            "type": "configuration/entityTypes/Individual",
            "createdBy": "admin",
            "createdTime": 1369070358438,
            "updatedTime": 1369070358438,
            "startDate": 35154000000,
            "roles": ["configuration/roles/Client"],
            "attributes": {
                "FirstName": [{
                    "type":
"configuration/entityTypes/Individual/attributes/FirstName",
                    "ov": true,
                    "value": "Ernest",
                    "uri": "entities/0000I77/attributes/FirstName/241"
                }],
                "LastName": [{
                    "type":
"configuration/entityTypes/Individual/attributes/LastName",
                    "ov": true,
                    "value": "Fusco",
                    "uri": "entities/0000I77/attributes/LastName/2CH"
                }]
 }
.....
}]
```

Match Explanation

You can search for potential matches of JSON array entities provided in the Customer Tenant.

Request

```
POST {DTSSURL}/entities/_verifyMatches
```

Parameters

	Name	Required	Description	Examples
HEADERS	`CustomerTenant`	Yes	Customer Tenant id.	-
	`DataTenant`	Yes	Data Tenant id.	-

URI PARAMETER	`rules`	Yes	Enabled match rules.	`POST {DTSSURL}/entities/_verifyMatches?rules=configuration/entityTyp /HCP/matchGroups/PersonByNPISuspect2`

BODY	-	Yes	A pair of entit ies to be matched.	<pre>{
				  "dtEntity": "entities/0000I77",
				  "ctEntity": "entities/0000BLK"
				}</pre>

```
{
  "dtEntity": {
    "type":
"configuration/entityTypes/Individual",
    "attributes": {
      "FirstName": [
        {
          "value": "Ernest"
        }
      ],
      "MiddleName": [
        {
          "value": "Miller"
        }
      ],
      "LastName": [
        {
          "value": "H."
        }
      ]
    }
  },
  "ctEntity": {
    "type":
"configuration/entityTypes/Individual",
    "attributes": {
      "FirstName": [
        {
          "value": "Ernest"
        }
      ],
      "MiddleName": [
        {
          "value": "M."
        }
      ],
      "LastName": [
        {
          "value": "Hemingway"
        }
      ]
    }
  }
}
```


Response

```
{
  "configuration/entityTypes/HCP/matchGroups/PersonByNPISuspect2": {
    "setAsNotMatch": false,
    "alreadySubscribed": false,
    "platformMatchRuleExplanation": {
      "matchTokens": {
        "first": {
          "foundInMatchTables": false,
          "tokens": [
            "0815092231"
          ]
        },
        "second": {
          "foundInMatchTables": true,
          "tokens": [
            "0815092231"
          ]
        },
        "intersection": {
          "tokens": [
            "0815092231"
          ]
        }
      },
      "rule": {
        "and": [
          {
            "and": [
              {
                "exact": {
                  "NPI": true
                }
              }
            ]
          }
        ]
      },
      "matched": true
    }
  }
}
```


Entities Search by DT Potential Matches

You can search for Customer Tenant entities by Data Tenant potential matches. Use the basic API facet search.

Available fields:

`dtPotentialMatches.matches`

`dtPotentialMatches.matchGroups`

`dtPotentialMatches.dataTenant`

Request Example (find Individual entities)

```
GET
{APIURL}/{TenantId}/entities?filter=(equals(dtPotentialMatches.matches,
'1') and equals(dtPotentialMatches.dataTenant,'dataTenant'))&max=2
Headers: Authorization: Bearer 204938ca-2cf7-44b0-b11a-1b4c59984512
```

Response

```
GET
{APIURL}/{TenantURL}/entities?filter=(equals(type,'configuration/entityT
ypes/Individual'))&max=2
Headers:
Authorization: Bearer 204938ca-2cf7-44b0-b11a-1b4c59984512,
Total: 3736124
[ { "uri": "entities/20",   ... }, { "uri": "entities/142",   ... } ]
```

Facet Search by DT Potential Matches

This function returns search counts for facet terms. Use the basic API facet search.

Available fields:

`dtPotentialMatches.matches`

`dtPotentialMatches.matchGroups`

`dtPotentialMatches.dataTenant`

Request Example

```
POST {APIURL}/{TenantId}/entities/_facets?filter=(equals(dtPotentialMatc
hes.dataTenant,'dataTenant'))
Headers: Authorization: Bearer 204938ca-2cf7-44b0-b11a-1b4c59984512
[
  {
    "fieldName" : "dtPotentialMatches.matches"
  },
  {
    "fieldName" : "dtPotentialMatches.matchGroups",
    "orderType" : "term",
    "pageSize" : 10,
    "pageNo" : 1
  }
]
```

Response

```
POST
{TenantURL}/entities/_facets?filter=(equals(type,'configuration/entityTy
pes/Individual'))
Headers: Authorization: Bearer 204938ca-2cf7-44b0-b11a-1b4c59984512
{
 "dtPotentialMatches.matches" : {
       "1" : 22,
       "2" : 33,
       "3" : 44
    },
    "dtPotentialMatches.matchGroups" : {
       "matchGroup1" : 17,
       "matchGroup2" : 32,
       "matchGroup3" : 6
    }
}
```

Entities Search in Data Tenants

Use this function to search in the Data Tenant for the subscribed Customer Tenant.

Request

```
GET {DTSSURL}/entities?filter=(equals(...))
```

Parameters

	Name	Required	Description	Examples

HEADERS	`CustomerTenant`	Yes	Customer Tenant id.	-
	`DataTenant`	Yes	Data Tenant id.	-

URI PARAMETERS	`filter`	No	Search query filter for Data Tenant.
			
			Enables entities filtering by a condition. Format for filter query parameter:
			
			`filter=({Condition Type}[AND/OR {Condition Type}]*)`
			
			The three Condition Types are:
			
			`equals(property, value)` - exact match condition.
			
			`startsWith(property, value)` - prefix condition; returns entities that have condition property starting with condition value.
			
			`fullText(property, value)` - full text search; returns entities that have condition property with condition value (any place).
			`property` in this condition is dot-delimited path until property on which to searc: `type`, `roles`, `attributes.Name`, `attribute.Education.Degree`.
			Search by multiple conditions is supported with AND and OR conditions.
			
			NOTE: Search in case insensitive.
			
			NOTE: If filter on activeness is not specified,then, by default it is: `filter=(equals(activeness, NOW()))`.
				Filter by entity type:
				
				`filter=(equals(type,'configuration/entityTypes/Organization'))`
				
				Filter by a String across all attributes:
				
				`filter=(equals(attributes,'Ann'))`
				
				Fill text search across all blog entries:
				
				`filter=(fullText(attributes.Blogs, 'Golf'))`
				
				Find Individual entities that have first name starting with 'An':
				
				`filter=(equals(type,'configuration/entityTypes/Individual') and startsWith(attributes.FirstName, 'An'))`
				
				Find active entities by 2005-10-17:
				
				`filter=(equals(activeness, 1129507200))`
				
				`OR`
				
				`filter=(gt(activeness.startDate, 1129507200) and lt(activeness.endDate,1129507200))`

URI PARAMETERS	`max`	No	Positive Integer value to identify maximum number of entities to return in a response. It can be used to organize pagination in combination with the `"offset"` parameter. *Default value is 50.*	`max=10`
	`offset`	No	Positive Integer value to identify starting what element in a result set should be returned in a response. It can be used to organize pagination in combination with the `"limit"` parameter. Default value is `0`.	`offset=120`

Response

The response contains search results (JSON array as in Reltio REST API entities search) from the Data Tenant.

```
{
    "uri": "entities/IzVRxyp",
    "type": "configuration/entityTypes/HCP",
    "attributes": {
      "FirstName": [
        {
          "bringToCT": true,
          "type": "configuration/entityTypes/HCP/attributes/FirstName",
          "ov": true,
          "value": "Leonides",
          "uri": "entities/IzVRxyp/attributes/FirstName/VlzMu25X"
        }
      ],
      "LastName": [
        {
          "bringToCT": false,
          "type": "configuration/entityTypes/HCP/attributes/LastName",
          "ov": true,
          "value": "Anwar",
          "uri": "entities/IzVRxyp/attributes/LastName/VlzMuJ8Z"
        }
      ]
```


Contracts

Configuration

Contracts are used to control objects import and are a part of billing.

Contract

```
[
   {
      "@type":"PURCHASE",
      "items":[
         {
            "type":"configuration/entityTypes/HCP"
         }
      ],
      "limit":2000,
      "warningPercentage":80,
      "expiryPolicy":"NO_SYNC",
      "expiryEmailTemplate":"There are no more remaining transactions
based upon your contract between ${dataTenant} and ${customerTenant}."
   }
]
```

Parameters

Name	**Required**	Description
`items`	Yes	Set of entity types to be processed.
`limit`	Yes	Import limit. Task is stopped if the limit is reached. Real time processing skips entities for those for which the limit is reached.
`warningPercentage`	No	A warning email is sent if the threshold is reached.
`expiryPolicy`	No	Controls receiving updates for entities already subscribed even after contract expiry. Possible values: `NO_SYNC` (default) `UPDATES_ONLY` `“expiryPolicy”` is a mutable field (can be changed by an admin or any user that has access to the subscription and Contracts API. `“expiryPolicy”` can be modified after contract creation. If `“expiryPolicy”` is changed after contract expiration, then synchronization should be changed according to the new value. The modification should not affect existing data, so that: if `“expiryPolicy”` is changed from `NO_SYNC` to `UPDATES_ONLY`, then there is no need to reprocess entities not consumed during `NO_SYNC`policy. if `“expiryPolicy”` is changed from `UPDATES_ONLY` to `NO_SYNC`, then there is no need to reprocess entities consumed during `UPDATES_ONLY`policy. If Contract expired and `“expiryPolicy”` is set to `NO_SYNC`, then legacy behavior should be preserved. If Contract expired and `“expiryPolicy”` is set to `UPDATES_ONLY`, then: for **already** subscribed entities Contract works as not expired. for **non-subscribed** entities legacy behavior is preserved.
`expiryEmailTemplate`	No	Email template for customizing email notification at contract expiration. At contract creation/modification, if the template is specified/modified, then a preview email is sent to the user who creates/modifies contract.

Email Template

A template can be added either when a contract is added or to an existing contract. When a contract is added or modified, if a template is specified or modified, then an email (with notification preview) is sent to the user who created or modified the contract.

Parameters

Name	Description
subject	Email subject
body	Email body pattern. Pattern is preformatted text except few variables that can be used inside it: ${dataTenant} ${customerTenant} ${contractId} ${threshold} ${limit} ${environment}

Reporting for Contract

Generates a CSV report S3 file with a contract report for a given contract id, date range (day, week, month, or custom range defined as two timestamps), and filter. The report contains two metrics (consumed entities per entity type and per user) and is generated for a date range provided by the user or evaluated by the default policy.

The report is generated by a Report Task for Contract of an ID provided in the request. Once the task is finished, an email with the report CSV file is sent to the email address provided in the request

Request

`POST {DTSSURL}/actions/report/contract`

Parameters

	Name	Required	Description
Headers	`CustomerTenant`	Yes	Customer Tenant id.
	`DataTenant`	Yes	Data Tenant id.
URI PARAMETERS	`emails`	Yes	The list of emails to receive the report.
	`contractId`	Yes	Contract id for which to build the report.

Body		Yes	[
			{
			"from":
			1537962579000,
			"to":
			1581262579000
			}
			]

Periodic Reporting

Periodic reporting is done by the Periodic Reporting Task.

DTSS Tasks

DTSS service is equipped with a set of different tasks that could be used to copy entities or relations from the Data Tenant to the Customer Tenant, to make data `autosubscription`, to pull the latest changes from the Data Tenant, etc.

Common Task Processing Parameters

The parameters are set in the `*.properties` files.

Name	Default	Description
`uploadThreadCount`	-	Maximum number of threads that can be used by a single task for uploading objects.
`entitiesPerThread`	-	Maximum number of objects that can be uploaded in a single pack.
`clustering.rangeSize`	`1000000`	Maximum number of objects that can be processed in total by a single subtask.
`clustering.clusterThreshold`	`0.8`	A quota of DTSS nodes that can process a single task and its subtasks simultaneously. (for example, `0.8` for a DTSS environment with 10 nodes means that only 8 nodes out of 10 can process a single task.
`subscription.auto.dbScan.pageSize`	-	Maximum number of objects that can be processed simultaneously.
`tasks.maxActiveTaskProcessors`	`10`	Maximum number of task executing threads on a node.
`oauth.timeout`	`120000`	If oauth is not reachable, then longer than `oauth.timeout` ms means a task fails.

Task Status

Status	Description
NOT_STARTED	Initial status after task creation
ASSIGNED	Once the task assigned to a cluster
STARTED	When the task is in progress (started it's job)
STOPPED	When the task is stopped
ERROR	When the task stopped due to errors
FINISHED	Task completed it's job and finished
WAITING	When the sub tasks are in progress. The parent task wait for them to finish.
WAITING_FOR_CT_RESOURCES	Child task moves to WAITING_FOR_CT_RESOURCES state while there are no synchronous credits on CT side and the task is not in progress. Once the CT gains positive credits, the sub tasks automatically move to Assigned and then Started state.

Common Task Fields

Input URIs List

Tasks can use explicitly setup input URIs instead of having to scan whole tenants.

A file with URIs should have the following format:

File format
<entityURI_1>

...
<entityURI_N>

File with URIs example:

Example
Uqg6Ve1Q
Vqg6D12Q
DSS6VeVC
Qqg6Ve4U
Vqg6Se98

Input Source

URIs set can be provided to a task by an input source fields. Input source can represent following sources: Amazon S3 file, file by external URL, list field in a task body.

Amazon S3 file

How to use:

Create an s3 file with uris.

Set S3 source in task body.

S3 source format:

S3 source format
"s3": {
"s3Bucket": "<bucket>",
"s3Key": "<path>"
}

S3 source example:

S3 source example
"s3": {
"s3Bucket": "reltio.dtss",
"s3Key": "Tasks/input_entities_uris.txt"
}

File by external URL

How to use:

Create a file with uris.

Set url of the file as a source in task body.

File with URIs source format:

URL source format
"url": "<address>"

File with URIs source example:

URL source example
"url": "example.com/file.txt"

List Field in Task Body

How to use:

Set list of URLs in task body.

URIs list source format:

List with URIs source format
"urisList": [

"<entityURI_1>" ,

...

"<entityURI_N>"

]

URIs list source example:

List with URIs source example
"url":[
"Vqg6D12Q",
"DSS6VeVC",
"Qqg6Ve4U",
"Vqg6Se98"

]

Common Task Endpoints

There are several endpoints in DTSS that are common for all tasks. The task type does not matter or what the subscription is for which it was created.

Get Task By Id

This method returns the DTSS task by task identifier.

HTTP METHOD	`GET`	Description
URL	`{DTSSURL}/task/{id}`	**Deprecated. Please use alias**: `{DTSSURL}/tasks/{id}`.
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`	
HTTP Parameters	`start_date` (optional)	Used for periodic tasks to set up the interval start of periodic runs.
	`optional_date` (optional)	Used for periodic tasks to set up the interval end of periodic runs.

Get\Search Tasks

HTTP METHOD	GET	Description
URL	`{DTSSURL}/task`	**Deprecated. Please use alias**: `{DTSSURL}/tasks`.

HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`	
HTTP Parameters	`customerTenantId`	Filter tasks by Customer Tenant.
	`dataTenantId`	Filter tasks by Data Tenant.

	`status`	Filter tasks by status. Status can be:
		
		`NOT_STARTED`
		`ASSIGNED`
		`STARTED`
		`STOPPED`
		`ERROR`
		`FINISHED`
		`WAITING`

	`type`	Filter tasks by task type. Type can be: `COPY`, `SUBSCRIBE`, `MATCH`, `BULKCOPYRELATIONSHIPS`, `IMPORTHIERARCHY`, `PULL`
	`clusterNode`	Filter tasks by `clusterNode`.
	`show`	Filter tasks by show parameter. It can be: `PARENT` - show only parent type tasks `CHILD` - show only child type tasks `ALL` - show all tasks
	`filter`	Filter tasks using different condition types as per the table below.

Available condition types for the filter parameter:

Name	Description	Examples
`equals`	Case-insensitive search by exact value.	`filter=equals(status,'stopped')` Would find tasks with status = `STOPPED`.
`equalsCaseSensitive`	Same as above but search is case-sensitive.	`filter=equalsCaseSensitive(type,'PULL')` Would find tasks with type = PULL.

`listEquals`	Case-insensitive search by exact value to be matched to a list of values.
	
	`listEquals(field, 'value1', 'value2', 'value3')` is the same as `equals(field, 'value1') or equals(field, 'value2')`.
		`filter=listEquals(emails,'test1','test2')`
		
		Would find tasks having following arrays of emails:
		
		`test1`
		`test2`
		`test1`, `test2`
		`test1`, `test3`
		`test2`, `test3`

`listEqualsCaseSensitive`	Same as above but search is case-sensitive.	
`lt`, `gt`	Search for tasks with values less(greater) than given value.	`filter=lt(pullStartTime, 1129507201)` Would find tasks with `pullStartTime` less than `1129507201filter=gt(startDate, 1129507202)`. Would find tasks with `startDate` greater than `1129507202`.

`lte`, `gte`	Same as above plus equal values will also be found.	`filter=lte(pullStartTime, 1129507203)`
		
		Would find tasks with `pullStartTime` less than or equal to `1129507203`.
		
		`filter=gte(startDate, 1129507204)`
		
		Would find tasks with `startDate` greater than or equal to `1129507204`.

`contains`	Search for tasks with values containing given value.	`filter=contains(type,'COPY')` Would find tasks with the types containing `COPY` such as `COPY` and `BULKCOPYRELATIONSHIPS`.
`startsWith`	Search for tasks with values starting with given value.	`filter=startsWith(status,'FINISH')` Would find tasks with the `status = FINISHED`.
`missing`	Search for tasks which do not have value for provided field.	`filter=missing(period)` Would find tasks without period value (non-periodic tasks).
`exists`	Search for tasks which have value for provided field.	`filter=exists(period)` Would find tasks with period value (periodic tasks).
`range`	Search for tasks with values belonging to given range of values.	`filter=range(startDate, 1129507205, 1129507216)` Would find tasks with `startDate` after `1129507205` and before `1129507216`.
`in`	Search for tasks which have at least one of the enumerated values.	`filter=in(type,'COPY,PULL')` Would find tasks with either `type = COPY` or `PULL`.
`regexp`	Search for tasks which meet `regexp` expression.	`filter=regexp(type,'S..S.*')` would find tasks with first and fourth 'S' letters for type (SUBSCRIBE)
`containsWordStartingWith`	Search for tasks which have a word of multi-word value starting with provided value.	`filter=containsWordStartingWith(error,'change')` Would find tasks with following error: `"(subscription configuration) has been changed"`.

Create Task

This method returns the DTSS task by task identifier.

HTTP METHOD	`POST`
URL	`{DTSSURL}/tasks/{type}`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`
BODY	Body is different for all tasks. See the requests example below.

Allowed types are:

`manual_copy`

`manual_subscribe`

`manual_match`

`manual_bulk_copy_relations`

`manual_import_hierarchy`

`manual_sync`

`manual_measure`

`manual_recovery`

`manual_full_import_import_log_reindex`

`manual_full_import_log_report`

`pull`

`periodic_report`

`periodic_recovery`

`periodic_pull`

Stop Task

This method can stop DTSS task.

HTTP METHOD	`PUT`
URL	`{DTSSURL}/tasks/{id}/_stop`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

Manual Tasks

Copy Task

Copies business objects from the Data Tenant to the Customer Tenant.

HTTP METHOD	`POST`
URL	`{DTSSURL}/tasks/manual_copy`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

BODY

Body

```
{
 "dataTenant": "dataTenantId",
  "customerTenant": "customerTenantId",
  "entityTypes": [ //optional: if not specified (as well as
"relationType)" - COPY section of Synchronization Configuration
is used
    {"set" of Entity Types from DT}
  ],
  "relationTypes": [ //optional: if not specified (as well as
"entityTypes") - COPY section of Synchronization Configuration
is used
    {"set" of Relationship Types from DT}
  ],
  "entityUris": {
    <source>
  },
  "relationUris": {
    <source>
  },
  "entitiesFilterInDtMetadata": {
    "<entityType1>": "<filter1>",
    ...
    "<entityTypeN>": "<filterN>"
  },
  "reportEmails": [
    {"set" of emails}
  ]
}
```

Example

Body

```
[
  {
    "dataTenant": "dataTenant",
    "customerTenant": "customerTenant",
    "entityTypes": [
      "configuration/entityTypes/HCP"
    ],
    "relationTypes": [
      "configuration/relationTypes/Relative"
    ],
    "entityUris": {
        "s3": {
          "s3Bucket": "reltio.dtss",
          "s3Key": "Tasks/input_entity_uris.txt"
        }
    },
    "relationUris": {
        "s3": {
          "s3Bucket": "reltio.dtss",
          "s3Key": "Tasks/input_relation_uris.txt"
        }
    },
    "entitiesFilterInDtMetadata": {
     "configuration/entityTypes/HCP": "equals(attributes.FirstName,
'Ola')"
    },
    "reportEmails": [
      user.name@reltio.com
    ]
  }
]
```

If no "`entityTypes`" and "`relationTypes`" were specified within the task body, `isConfigFromSubscription`, then the `true` flag is returned within the response (meaning, the configuration was taken from Synchronization Configuration).

Manual Copy Bulk Relationship Task

Copies relationships from the Data Tenant to the Customer Tenant.

HTTP METHOD	POST	
URL	`{DTSSURL}/tasks/`**`manual_bulk_copy_relations`**	
	HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

BODY	**Body**
	
	{
	"dataTenant": "dataTenantId",
	"customerTenant":
	"customerTenantId",
	"relationTypes": [
	{"set" of Relationship Types from
	DT}
	],
	"relationUris": {
	<source>
	},
	"reportEmails": [
	{"set" of emails}
	]
	}	

Example

Body

```
[
  {
   "dataTenant": "dataTenant",
    "customerTenant": "customerTenant",
    "relationTypes": [
      "configuration/relationTypes/Relative"
    ],
    "relationUris": {
        "s3": {
          "s3Bucket": "reltio.dtss",
          "s3Key": "Tasks/input_relation_uris.txt"
        }
    },
    "reportEmails": [
      user.name@reltio.com
    ]
  }
]
```

Manual AutoSubscribe Task

Finds matches for Data Tenant entities in the Customer Tenant, and subscribes corresponding Customer Tenant entities on those matched Data Tenant entities.

HTTP METHOD	`POST`
URL	`{DTSSURL}/tasks/manual_subscribe`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

BODY	**Body**
	
	`{`
	`  "dataTenant": "dataTenantId",`
	`  "customerTenant": "customerTenantId",`
	`  "entityType": "Entity Type from DT", //deprecated`
	`  "matchRules": [ {"set" of Match Rules from DT} ],`
	`//deprecated`
	`  "entityMappings": [ //optional: if not specified - AUTOSUBSCRIBE section of Synchronization Configuration from Subscription is used`
	`  {`
	`  "entityType": "Entity Type from DT",`
	`  "matchRules": [`
	`  {"set" of Match Rules from DT}`
	`  ]`
	`  }, ...`
	`  ],`
	`  "entityUris": {`
	`  <source>`
	`  },`
	`  "entitiesFilterInDtMetadata": {`
	`  "<entityType1>": "<filter1>",`
	`  ...`
	`  "<entityTypeN>": "<filterN>"`
	`  },`
	`  "reportEmails": [`
	`  {"set" of emails}`
	`  ]`
	`}`

Example

Body

```
[
  {
    "dataTenant": "dataTenant",
    "customerTenant": "customerTenant",
    "entityType": "configuration/entityTypes/HCP", //deprecated
    "matchRules": [
"configuration/entityTypes/HCP/matchGroups/PersonByExactNameVsZip" ],
//deprecated
    "entityMappings": [
   {
    "entityType": "configuration/entityTypes/HCP",
    "matchRules": [
   "configuration/entityTypes/HCP/matchGroups/PersonByExactNameVsZip"
  ]
   }
    ],
    "entityUris": {
        "s3": {
          "s3Bucket": "reltio.dtss",
          "s3Key": "Tasks/input_entity_uris.txt"
        }
    },
    "entitiesFilterInDtMetadata": {
     "configuration/entityTypes/HCP": "equals(attributes.FirstName,
'Ola')"
    },
    "reportEmails": [
      user.name@reltio.com
    ]
  }
]
```

If no "`entityMappings`" were specified within the task body, the `isConfigFromSubscription: true` flag is returned within the response (meaning, the configuration was taken from the Synchronization Configuration).

Match Task

Builds/updates a match table.

Can be executed to create/update a match table with the following `autosubscribe` task execution.

Note: Match rules defined in the "`MANUAL_MATCH`" section of the Synchronization Configuration is used by the task (if the section does exist).

HTTP METHOD	`POST`
URL	`{DTSSURL}/tasks/manual_match`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

BODY	**Body**
	{
	"dataTenant": "dataTenantId",
	"customerTenant": "customerTenantId",
	"entityUris": {
	<source>
	},
	"entitiesFilterInDtMetadata": {
	"<entityType1>": "<filter1>",
	...
	"<entityTypeN>": "<filterN>"
	}
	}

Parameters

Name	Required	Description
`oauth_token`	Yes	Authorization token of the user.

Since 2017.3, match task automatically starts upon the following configuration updates:

Subscription config
DT/CT cassandra config
DT/CT DTSS config (registering Customer Tenants and Data Tenants)
DT/CT L3

unless
a. `autoRematch` parameter is set to `false` in the subscription configuration (default is `true`).
b. Entities count in the Customer Tenant exceeds the "`subscription.auto.rematch.threshold`" property value from the DTSS properties file (default is `500000`).

If the `autoRematch` parameter is set to `false` in the subscription configuration, task notification is not sent.

Task notification is sent in either case if the task was started or was not started because of exceeding the `subscription.auto.rematch.threshold` value.

In the first case, an email message reports indicate that the task was started to rebuild the match table because of a configuration update.
In the second case, an email message recommends to start the match task manually to rebuild the match table because of a configuration update.

Manual Full Import Log Reindex Task

Performs an ES reindexing of all the actions stored in Cassandra.

HTTP METHOD	POST
URL	`{DTSSURL}/tasks/manual_full_import_log_reindex`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

BODY	**Body** `{ "dataTenant": "dataTenantId", "customerTenant": "customerTenantId", "reportEmails": [ {"set" of emails} ] }`

Manual Pull Event Task

Performs delayed processing of JMS events starting from a certain time. Uses `ActivityLog` to read the events.

Task does whatever can be done in real-time: auto subscribe, copy, business objects removal, etc.

HTTP METHOD	POST
URL	`{DTSSURL}/tasks/pull`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`
BODY	**Body** `{ "dataTenant": "dataTenantId", "customerTenant": "customerTenantId", "pullStartTime": "2015-09-22T01:22:12.143" }`

`"pullStartTime"` is a date in ISO8601 format or Unix time in milliseconds. Examples:

`2015-09-22T01:22:12.143, 2015-09-22T01:22:12, 2015-09-22T01:22, 2015-09-22`

`1442884932143`

Import Connections

DTSS can import relations and entities using a connections request. This request is similar to the connections request in API.

HTTP METHOD	POST
URL	`{DTSSURL}/entities/_connections/_hasNonImported`

HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}` DataTenant: `<dataTenantId>` CustomerTenant: `<customerTenantId>`
URI PARAMETER	`customerEntityUri`
BODY	Body of this request is the same as for the API Connections Request Method.

HTTP METHOD	`POST`
URL	`{DTSSURL}/entities/_connections/_import`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}` DataTenant: `<dataTenantId>` CustomerTenant: `<customerTenantId>`
URI PARAMETER	`customerEntityUri`
BODY	Body of this request is the same as for the API Connections Request Method.

Request below is used to get all relations for a particular Data Tenant Entity.

HTTP METHOD	`POST`
URL	`{DTSSURL}/entities/_connections/_getConnections`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}` DataTenant: `<dataTenantId>` CustomerTenant: `<customerTenantId>`
URI PARAMETER	`dataEntityUri`
BODY	Body of this request is the same as for the API Connections Request Method.

Import Hierarchy

Copy of a hierarchy from the Data Tenant to the Customer Tenant.

Bring hierarchy to the Customer Tenant for a Data Tenant in three modes:

whole hierarchy of certain graph type to which this entity belongs

descendant hierarchy of certain graph type

ancestor hierarchy of certain graph type

You should specify `graphTypeURIs` that determine which hierarchy should import DTSS.

		Description
HTTP METHOD	`POST`	
URL	`{DTSSURL}/entities/_tree/_canImport`	

HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}` DataTenant: `<dataTenantId>` CustomerTenant: `<customerTenantId>`	
URI PARAMETERS	`customerEntityUri`	Root entity to build a hierarchy.
	`graphTypeURIs`	Graph Type of hierarchy.
	`importHierarchyStrategy`	Import hierarchy strategy. Possible values: `WHOLE_HIERARCHY`, `ANCESTORS_HIERARCHY`, `DESCENDANTS_HIERARCHY`. `WHOLE_HIERARCHY` - Import entire hierarchy. `ANCESTORS_HIERARCHY` - Import tree of ancestors for the Data Tenant entity which has been subscribed by the Customer Tenant entity with `uri=customerEntityUri`. `DESCENDANTS_HIERARCHY` - Import tree of descendants for the Data Tenant entity which has been subscribed by the Customer Tenant entity with `uri=customerEntityUri`.
	`debug`	

HTTP METHOD	`POST`
URL	`{DTSSURL}/tasks/manual_import_hierarchy`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

BODY	**Body**
	
	`{`
	`  "dataTenant": "dataTenantId",`
	`  "customerTenant": "customerTenantId",`
	`  "graphTypes": [`
	`  "<grapTypeURI>"`
	`  ],`
	`  "importHierarchyStrategy":`
	`"WHOLE_HIERARCHY|ANCESTORS_HIERARCHY|DESCENDANTS_HIERARCHY",`
	`  "ctEntityUri": "<ctEntityUri>"`
	`}`

Measure Task

This task can show all inconsistencies between the Data Tenant and the Customer Tenant. Task history contains these counters which show inconsistencies for the following:

`notSetEndDatedCrosswalksEntities` - Number of entities which contain a crosswalk that has been removed from the Data Tenant but that still exists in the Customer Tenant, and for which the end date should be set.

`notDeletedCrosswalksEntities` - Number of entities which contain a crosswalk that has been removed from the Data Tenant but that still

exists in Customer tenant and should be deleted.

`notProcessedCrosswalksTotalEntities` - Number of entities which contain a crosswalk that has been removed from Data Tenant (it is the union of `notSetEndDatedCrosswalks` and `notDeletedCrosswalks`).

`notSetEndDatedCrosswalksRelations` - Number of relations which contain a crosswalk that has been removed from the Data Tenant but that still exists in the Customer Tenant, and for which the end date should be set.

`notDeletedCrosswalksRelations` - Number of relations which contain a crosswalk that has been removed from the Data Tenant but that still exists in the Customer Tenant and should be deleted.

`notProcessedCrosswalksTotalRelations` - Number of relations which contain crosswalk that has been removed from the Data Tenant (it is the union of `notSetEndDatedCrosswalks` and `notDeletedCrosswalks`).

`notAutoSubscribedEntities` - Number of entities which can be `Autosubscribed`.

`notCopiedEntities` - Number of entities which can be copied to the Customer Tenant.

`notCopiedRelations` - Number of relations which can be copied to the Customer Tenant.

`notUpdatedEntities` - Number of entities which have been updated in the Data Tenant, but changes have not been propagated to the Customer Tenant.

`notUpdatedRelations` - Number of relations which have been updated in the Data Tenant, but changes have not been propagated to the Customer Tenant.

`notCorrectlyIndexed` - Number of DTSS internal match table inconsistencies.

Task history contains counters by type and common counters. In addition, this task uploads the next report to S3:

Report per entity type which contains set of uris of entities in the Customer Tenant which can be `Autosubscribed`.

Report per entity type which contains set of uris of entities in the Data Tenant which can be copied to the Customer Tenant from the Data Tenant.

Report per entity type which contains set of uris of entities in the Customer Tenant which should be updated.

Report per relation type which contains set of uris of relations in the Data tenant which can be copied to the Customer Tenant from the Data Tenant.

Report per relation type which contains set of uris of relations in the Customer tenant which should be updated.

HTTP METHOD	POST
URL	`{DTSSURL}/tasks/manual_measure`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`
BODY	**Body**

```
{
  "dataTenant": "dataTenantId",
  "customerTenant": "customerTenantId",
  "dtEntityTypes": [
    {"set" of Entity Types from DT (this set should be subset
of types which specified in Synchronization Config). If this
field is empty, it will be take from Synchronization config.}
  ],
  "dtRelationTypes": [
    {"set" of Relationship Types from DT (this set should be
subset of types which specified in Synchronization Config).  If
this field is empty, it will be take from Synchronization
config.}
  ],
  "dtEntityUris": {
    "s3": {
      "s3Bucket": "reltio.dtss",
      "s3Key": "Tasks/input_dt_entity_uris.txt"
    }
  },
  "ctEntityUris": {
    "s3": {
      "s3Bucket": "reltio.dtss",
      "s3Key": "Tasks/input_ct_entity_uris.txt"
    }
  },
  "dtRelationUris": {
    "s3": {
      "s3Bucket": "reltio.dtss",
      "s3Key": "Tasks/input_dt_relation_uris.txt"
    }
  },
  "ctRelationUris": {
    "s3": {
      "s3Bucket": "reltio.dtss",
      "s3Key": "Tasks/input_ct_relation_uris.txt"
    }
  },
  "entitiesFilterInDtMetadata": {
    "<entityType1>": "<filter1>",
    ...
    "<entityTypeN>": "<filterN>"
  },
  "reportEmails": [
    {"set" of emails}
  ]
}
```

You can find a report containing a set of uris in the S3 service in the `reltio.dtss` bucket.

History Example


```
{
  "threadsStatuses": {
    "processCtEntities": "FINISHED",
    "processCtRelations": "FINISHED",
    "scanCtRelations": "FINISHED",
    "email": "FINISHED",
    "scanDtEntities": "FINISHED",
    "s3report": "FINISHED",
    "scanCtEntities": "FINISHED",
    "scanDtRelations": "FINISHED",
    "processDtEntities": "FINISHED",
    "processDtRelations": "FINISHED"
  },
  "counters": {
    "processedDtEntities": {
      "value": 1,
      "dynamic": {
        "1468391527741": 1
      }
    },
    "scannedDtEntities": {
      "value": 1,
      "dynamic": {
        "1468391527741": 1
      }
    },
    "notCopiedEntities": 1
  },
  "errorLog": [],
  "ctEntitiesCursor": "",
  "dtEntitiesCursor": "",
  "ctRelationsCursor": "",
  "dtRelationsCursor": "",
  "reports": [

"local_tests/measure_task/904315fd-41e2-494f-9cc4-51a501380edc/report_en
tity__copy__configuration_entityTypes_HCP.json.gz"
  ],
  "counters_by_type": {
    "configuration/entityTypes/HCP": {
      "notCopiedEntities": 1
    }
  }
}
```

Sync Task

Performs a full re-upload of slices for the subscribed Customer Tenant entities and relations. Synchronization is performed at the crosswalks level. The task doesn't maintain entity consistency but crosswalks. For full synchronization please use a Pull Task.

Use case:

Customer Tenant contains E1 with crosswalks A and B

Data Tenant contains E2 (crosswalk A) and E3 (crosswalk B)

SyncTask executed

Customer Tenant contains E1 with crosswalks A and B (as before)

Data Tenant contains E2 (crosswalk A) and E3 (crosswalk B) (as before)

E2 is consistent to E1 by crosswalk A

E3 is consistent to E1 by crosswalk B

HTTP METHOD	`POST`
URL	`{DTSSURL}/tasks/manual_sync`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

BODY

Body

```
{
  "dataTenant": "dataTenantId",
  "customerTenant": "customerTenantId",
  "entityCtTypes": [
    {"set" of Entity Types from CT}
  ],
  "relationCtTypes": [
    {"set" of Relationship Types from CT}
  ],
  "entityUris": {
    <source>
  },
  "relationUris": {
    <source>
  },
  "entitiesFilterInDtMetadata": {
    "<entityType1>": "<filter1>",
    ...
    "<entityTypeN>": "<filterN>"
  },
  "reportEmails": [
    {"set" of emails}
  ]
}
```

Example

Body

```
[
  {
    "dataTenant": "dataTenant",
    "customerTenant": "customerTenant",
    "entityCtTypes": [
      "configuration/entityTypes/HCP"
    ],
    "relationCtTypes": [
      "configuration/relationTypes/Relative"
    ],
    "entityUris": {
        "s3": {
          "s3Bucket": "reltio.dtss",
          "s3Key": "Tasks/input_entity_uris.txt"
        }
    },
    "relationUris": {
        "s3": {
          "s3Bucket": "reltio.dtss",
          "s3Key": "Tasks/input_relation_uris.txt"
        }
    },
    "entitiesFilterInDtMetadata": {
     "configuration/entityTypes/HCP": "equals(attributes.FirstName,
'Ola')"
    },
    "reportEmails": [
      user.name@reltio.com
    ]
  }
]
```

Recovery Task

Recovers inconsistencies that satisfy these constrains: filter, ranges, and taskIds.

Recovery task scans FIL for JmsEvent, Endpoint, and Scan to determine unfinished operations.

Builds a report that contains two sections, describing:

consistency state prior to the recovery

consistency state after the recovery

HTTP METHOD	POST
URL	{DTSSURL}/tasks/manual_recovery

HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

BODY	**Body**
	
	`{`
	`"dataTenant": "dataTenantId",`
	`"customerTenant": "customerTenantId",`
	`"taskIds": [`
	`"19c8e163-0058-482a-862c-73a31f61c7fd",`
	`"2f2d8a9f-8362-4391-a086-1220512ac6ed"`
	`],`
	`"filter": "equalsCaseSensitive(TenantType,'DATA')",`
	`"ranges": [`
	`{`
	`"from": "2015-09-22T01:22:12.143",`
	`"to": "2015-09-23T05:31:10.761"`
	`}`
	`]`
	`}`

Report Task

Generates a CSV report S3 file for given date range (day, week, month, or custom range defined as two timestamps) and filter.

Each report entry (defined in `"reportEntries"`) corresponds to a separate counter.

HTTP METHOD	`POST`
URL	`{DTSSURL}/tasks/manual_full_import_log_report`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

BODY

Body

```
{
  "dataTenant": "dataTenantId",
  "customerTenant": "customerTenantId",
  "reportEntries": [
    {
      "name": "ConsumedPerType",
      "keyField": "type"
    },
    {
      "name": "ConsumedPerUser",
      "keyField": "user"
    }
  ],
  "taskIds": [
    "19c8e163-0058-482a-862c-73a31f61c7fd",
    "2f2d8a9f-8362-4391-a086-1220512ac6ed"
  ],
  "filter": "equalsCaseSensitive(Success,'true') and
equalsCaseSensitive(ActionType,'Upload') and
equalsCaseSensitive(TenantType,'DATA')",
  "ranges": [
    {
      "from": "2018-11-15T00:00:00.000Z",
      "to": "2018-11-26T00:00:00.000Z"
    }
  ],
  "output": {
    "exportAction": true,
    "s3Destination": {
      "s3Bucket": "customer-bucket",
      "s3Key": "customer/reports"
    }
  },
  "distinctCounters": true,
  "reportEmails": [
    "example@mail.com"
  ]
}
```

Parameters

Name	Required	Description
`dataTenant`	Yes	Data Tenant id.
`customerTenant`	Yes	Customer Tenant id.

`reportEntries`		Action fields to build metrics for. It is not recommended to use high-cardinality fields here. Fields available:
		`actionType`
		`type`
		`user`
		`success`
		`syncType`
		`objectType`
		`metadataType`
		`tenantType`
		`jmsEventType`

`taskIds`	No	Task ids for which to build report.

`filter`	No	Search by the query filter in Full Import Log (refer to EntitySearch for more information).
		Enables actions conditional filter with following format:
		`filter=({Condition Type}[AND/OR {Condition Type}]*)`
		Examples:
		`filter=equals(SubscriptionId, 'staticData_staticCustomer')`
		`filter=gt(Timestamp, 1129507200) and lt(Timestamp,1129507200)`
		`filter=(equals(ObjectUri,'00uwwbI'))`

`ranges`	Yes	Date ranges for which to build report.

`output`	No	Contains the following sub fields:
		`exportAction`
		Default is `false`. If `true`, then actions are uploaded to S3.
		`s3Destination`
		If used, then S3 report is put into a custom S3 location. DTSS may not have access to the custom location; therefore, you should create a ticket and forward to Support/DevOps. After access to DTSS is granted the location can be used.
		If used, then both the s3Bucket and s3Key should be present.

`distinctCounters`	No	Default is `false`. If `true`, then distinct counters are calculated and sent by email.
`emails`	No	The list of emails to receive the report.

Consistency Report Task

Description

Evaluates subscription consistency by building a DT-CT comparison report (the task is similar to MeasureTask but works on the attributes level). Profiles are consistent if they are consistent against DTSS subscription; that is, if their synchronized state is compliant to Synchronization Configuration. The task finds inconsistencies of the following types:

Attributes mismatch (CT entities that have a DT crosswalk): mismatch on the attributes level; for example, the phone number for entity A should be exactly the same in both the CT and DT.

Missing entities: DT entities that should be copied into CT by COPY strategy but are absent in CT.

Not subscribed entities: DT entities that should be copied into CT by `AUTOSUBSCRIBE` strategy but are absent in CT.

Report

The report is stored in zipped CSV files. The report is uploaded to DTSS S3 bucket or custom output folder.

Here is an example report:

Name	DT URI	DT Type	CT URIs	CT Types	Sources	Mismatched Attributes
Mismatch	001iNlo	HCP	001iaYa	HCP	FB	FirstName|LastName
Not copied	001k6KG	HCO			JAZZ	
Not copied	001jgki	Location			JAZZ	
Not copied	001iWIK	HCP			CIS	
Mismatch	001iS24	HCP	001ij56	HCP	CIS	FirstName|LastName
Not subscribed	001jtXU	HCP	001ieoq	HCP	JAZZ	

Report is stored in multiple volumes if the zipped report size exceeds the volume size limit. By default, the limit is 10MB and is configured by the DTSS service property `'task.config.consistency.report.default.volume.size.limit.bytes'`. To use a custom value for a single task, please use the `reportVolumeSizeLimitInBytes` task field (see the example below).

Scheduling a Consistency Report Task

Report can be scheduled in two ways:

In a periodic manner: once per period in milliseconds or by CRON (see DTSS Configuration Procedures (Save)#ConsistencyReportTask for more information).

As a hook on other tasks finish (see ConsistencyReportConfiguration for more information).

Starting a Consistency Report Task

Example 1: Default (`dtEntityTypes` and `dtEntitiesFilterInDtMetadata` skipped; therefore, to be pulled from Synchronization Configuration)

HTTP METHOD	POST
URL	`{DTSSURL}/tasks/manual_consistency_report`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`
BODY	**Body** `[ { "dataTenant": "dt", "customerTenant": "ct", "reportEmails": [ "default.report.test.email@reltio.com" ] } ]`

Example 2: Manually selected DT types and filters

HTTP METHOD	POST
URL	{DTSSURL}/tasks/manual_consistency_report
HEADERS	Content-Type: application/json Authorization: Bearer {oauth_token}

BODY	**Body**
	[
	  {
	  "dataTenant": "dt",
	  "customerTenant": "ct",
	  "dtEntityTypes": [
	  "configuration/entityTypes/HCP"
	  ],
	  "dtEntitiesFilterInDtMetadata": {
	  "configuration/entityTypes/HCP": "equals(createdBy, 'batch-processor')"
	  },
	  "reportEmails": [
	  "default.report.test.email@reltio.com"
	  ]
	  }
	]

Example 3: Run for certain DT uris (dtEntityUris)

HTTP METHOD	POST
URL	{DTSSURL}/tasks/manual_consistency_report
HEADERS	Content-Type: application/json Authorization: Bearer {oauth_token}

BODY	**Body**
```
[
  {
    "dataTenant": "dt",
    "customerTenant": "ct",
    "dtEntityTypes": [
      "configuration/entityTypes/HCP"
    ],
    "dtEntitiesFilterInDtMetadata": {
      "configuration/entityTypes/HCP": "equals(createdBy,
'batch-processor')"
    },
    "dtEntityUris": {
      "urisList": [
        "P5GfyQ3",
        "44AbyQ3",
        "Z1Gfyz2",
        "01ifyQ3"
      ]
    },
    "reportEmails": [
      "default.report.test.email@reltio.com"
    ]
  }
]
``` |

Example 4: Custom S3 destination key (make sure DTSS AWS account has access to the destination)

HTTP METHOD	POST
URL	`{DTSSURL}/tasks/manual_consistency_report`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

BODY	**Body**
	[
	  {
	  "dataTenant": "staticData0",
	  "customerTenant": "staticCustomer0",
	  "output": {
	  "s3Bucket": "customer-bucket",
	  "s3Key": "customer/reports"
	  },
	  "reportEmails": [
	  "default.report.test.email@reltio.com"
	  ]
	  }
	]

Example 5: Custom report volume size limit (about default value and more at Report).

HTTP METHOD	POST
URL	`{DTSSURL}/tasks/manual_consistency_report`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

BODY	**Body**
	[
	  {
	  "dataTenant": "staticData0",
	  "customerTenant": "staticCustomer0",
	  "reportVolumeSizeLimitInBytes": 1048576,
	  "reportEmails": [
	  "default.report.test.email@reltio.com"
	  ]
	  }
	]

Periodic Tasks

Common Fields

Name	Description
`startTime`	Start time constraint.
`period`	A period in CRON format or Unix time in milliseconds. Examples: `"15 14 1 * *"` (execute at the first day of the each month at 3:14 PM) `5000` (execute every 5000 milliseconds)

Periodic Pull Event Task

Periodically starts a Manual Pull Event Task**.**

HTTP METHOD	POST
URL	`{DTSSURL}/tasks/periodic_pull`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

BODY	**Body**
	`{`
	`  "dataTenant": "dataTenantId",`
	`  "customerTenant": "customerTenantId",`
	`  "pullStartTime": "2015-09-22T01:22:12.143",`
	`  "period": "* * * * *"`
	`}`

(`"pullStartTime"` is an alias for `"startTime"`.)

Periodic Reporting Task

Builds a report for all the inconsistencies that appeared after the `"startTime"` and which satisfy these constrains: filter, ranges, and taskIds.

HTTP METHOD	POST
URL	`{DTSSURL}/tasks/periodic_report`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

BODY

Body

```
{
  "dataTenant": "dataTenantId",
  "customerTenant": "customerTenantId",
  "reportEntries": [
    {
      "name": "ConsumedPerType",
      "keyField": "type"
    },
    {
      "name": "ConsumedPerUser",
      "keyField": "user"
    }
  ],
  "period": "0 0 * * 0",   // each Sunday at midnight
  "filter": "equalsCaseSensitive(Success,'true') and
equalsCaseSensitive(ActionType,'Upload') and
equalsCaseSensitive(TenantType,'DATA')",
  "output": {
    "exportAction": true,
    "s3Destination": {
      "s3Bucket": "customer-bucket",
      "s3Key": "customer/reports"
    }
  },
  "reportEmails": [
    "example@mail.com"
  ]
}
```

Parameters

Name	Required	Description
`dataTenant`	Yes	Data Tenant id.
`customerTenant`	Yes	Customer Tenant id.

`reportEntries`		Action fields to build metrics for. It is not recommended to use high-cardinality fields here. Fields available:
		`actionType`
		`type`
		`user`
		`success`
		`syncType`
		`objectType`
		`metadataType`
		`tenantType`
		`jmsEventType`

`taskIds`	No	Task ids for which to build report.

`filter`	No	Search by the query filter in Full Import Log (refer to Entity Search for details).
		Enables actions conditional filter with the following format:
		`filter=({Condition Type}[AND/OR {Condition Type}]*)`
		
		Examples:
		`filter=equals(SubscriptionId, 'staticData_staticCustomer')`
		`filter=gt(Timestamp, 1129507200) and lt(Timestamp,1129507200)`
		`filter=(equals(ObjectUri,'00uwwbI'))`

`output`	No	Contains the following sub fields:
		`exportAction`
		Default is `false`. If `true`, then actions are uploaded to S3.
		`s3Destination`
		If used, then S3 report is put into a custom S3 location. DTSS may not have access to the custom location; therefore, you should create a ticket and forward to Support/DevOps. After access to DTSS is granted, the location can be used.
		If used, then both s3Bucket and s3Key should be present.

emails	No	The list of emails to receive the report.

Periodic Recovery Task

Periodically starts a Recovery Task.

HTTP METHOD	POST
URL	`{DTSSURL}/tasks/periodic_recovery`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

BODY	**Body**
	{
	"dataTenant": "dataTenantId",
	"customerTenant": "customerTenantId",
	"startTime": "2015-09-22T01:22:12.143",
	"period": "* * * * *",
	"taskIds": [
	"19c8e163-0058-482a-862c-73a31f61c7fd",
	"2f2d8a9f-8362-4391-a086-1220512ac6ed"
	],
	"filter": "equalsCaseSensitive(TenantType,'DATA')"
	}

Periodic Measure Task

Periodically starts a Measure Task.

HTTP METHOD	POST
URL	{DTSSURL}/tasks/periodic_measure
HEADERS	Content-Type: application/json Authorization: Bearer {oauth_token}

BODY	**Body**
	{
	"dataTenant": "dataTenantId",
	"customerTenant": "customerTenantId",
	"period": "* * * 1 *"
	}

Periodic Consistency Reporting Task

Scheduled execution of DTSS Configuration Procedures (Save)#ConsistencyReportTask based on interval in milliseconds or CRON format string. History of all executions is kept in PeriodicConsistencyReportingTask history.

Examples

Example 1: Default with period (schedules default version of DTSS Configuration Procedures (Save)#ConsistencyReportTask once in 24 hours)

HTTP METHOD	POST
URL	{DTSSURL}/tasks/periodic_recovery
HEADERS	Content-Type: application/json Authorization: Bearer {oauth_token}

BODY	**Body**
	{
	"dataTenant": "dataTenantId",
	"customerTenant": "customerTenantId",
	"startTime": "2015-09-22T01:22:12.143",
	"period": "* * * * *",
	"taskIds": [
	"19c8e163-0058-482a-862c-73a31f61c7fd",
	"2f2d8a9f-8362-4391-a086-1220512ac6ed"
	],
	"filter": "equalsCaseSensitive(TenantType,'DATA')"
	}

Example 2: Default CRON (schedules default version of DTSS Configuration Procedures (Save)#ConsistencyReportTask each Sunday at midnight).

HTTP METHOD	POST
URL	{DTSSURL}/tasks/periodic_recovery
HEADERS	Content-Type: application/json Authorization: Bearer {oauth_token}

BODY	**Body**
	{
	"dataTenant": "dataTenantId",
	"customerTenant": "customerTenantId",
	"startTime": "2015-09-22T01:22:12.143",
	"period": "* * * * *",
	"taskIds": [
	"19c8e163-0058-482a-862c-73a31f61c7fd",
	"2f2d8a9f-8362-4391-a086-1220512ac6ed"
	],
	"filter": "equalsCaseSensitive(TenantType,'DATA')"
	}

Example 3: Customizing DT entity types and filters

HTTP METHOD	POST
URL	{DTSSURL}/tasks/periodic_recovery

HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

BODY	**Body**
	
	`{`
	`  "dataTenant": "dataTenantId",`
	`  "customerTenant": "customerTenantId",`
	`  "startTime": "2015-09-22T01:22:12.143",`
	`  "period": "* * * * *",`
	`  "taskIds": [`
	`  "19c8e163-0058-482a-862c-73a31f61c7fd",`
	`  "2f2d8a9f-8362-4391-a086-1220512ac6ed"`
	`  ],`
	`  "filter": "equalsCaseSensitive(TenantType,'DATA')"`
	`}`

Example 4: Custom S3 destination key (make sure DTSS AWS account has access to the destination)

HTTP METHOD	POST
URL	`{DTSSURL}/tasks/periodic_recovery`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

BODY	**Body**
	
	`{`
	`  "dataTenant": "dataTenantId",`
	`  "customerTenant": "customerTenantId",`
	`  "startTime": "2015-09-22T01:22:12.143",`
	`  "period": "* * * * *",`
	`  "taskIds": [`
	`  "19c8e163-0058-482a-862c-73a31f61c7fd",`
	`  "2f2d8a9f-8362-4391-a086-1220512ac6ed"`
	`  ],`
	`  "filter": "equalsCaseSensitive(TenantType,'DATA')"`
	`}`

Example 5: Custom report volume size limit (about default value and more at Report)

HTTP METHOD	POST

URL	`{DTSSURL}/tasks/periodic_recovery`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

BODY	**Body**
	
	`{`
	`  "dataTenant": "dataTenantId",`
	`  "customerTenant": "customerTenantId",`
	`  "startTime": "2015-09-22T01:22:12.143",`
	`  "period": "* * * * *",`
	`  "taskIds": [`
	`  "19c8e163-0058-482a-862c-73a31f61c7fd",`
	`  "2f2d8a9f-8362-4391-a086-1220512ac6ed"`
	`  ],`
	`  "filter": "equalsCaseSensitive(TenantType,'DATA')"`
	`}`

Tasks TTL

In order not to have too big CFs, manual/periodic tasks and history are cleaned once task TTL has expired.

Name	Default	Description
`task.ttl.seconds`	`2592000` (30 days)	Task TTL for Cassandra

Tasks Background Services

There are additional task background services which are managed by properties parameters.

a. Task failover service restarts DTSS tasks from nodes that are no longer responding.

`task.failover.enabled` (default = `true`) - defines if task failover is enabled.

`task.failover.waitTimeout` (default = `600000` (10 minutes in milliseconds)) - defines the period to restart tasks.

b. Task consistency service reports if metadata or subscription configuration has been changed during the task run. The report is presented in the `"configurationErrorReport"` task section.

`task.consistency.enabled` (default = `true`) - defines if task consistency is enabled.

`task.consistency.waitTimeout` (default = `600000` (10 minutes in milliseconds)) - defines the period to check tasks for configuration changes.

Admin Features

Full Import Log

Overview

Full Import Log is a log to store information about related base DTSS operations. The log can be used to obtain precise statistics, improve reliability, investigate performance issues, realize a clearer understanding of what is happening in the DTSS cluster, and so forth.

The base unit of the Full Import Log is an Action record (refer to Action for details).

Full Import Log records are stored in Cassandra in the `fullImportLogCF` column family and indexed into ES into the `fullImportLogESIndex` index (refer to Tenant Subscription for details).

Full Import Log Service also builds a `*.csv` report after a task finishes and then uploads them to S3. You can find the report paths in the tasks history `fullImportLogPaths` field.

Action

Name	Description
`actionId`	Unique ID.
`actionType`	Type of the action (see ActionType for details).
`syncType`	Type of the synchronization to which the action belongs (see SyncType for details).
`success`	Whether the operation has succeeded.
`timestamp`	Timestamp.
`objectUri`	A URI of an object that has been processed at the operation.
`type`	Type of the object.
`metadataType`	Either `"ENTITY"` or `"RELATION"`.
`tenantId`	Tenant of `objectUri`.
`tenantType`	Tenant type of `tenantId`.
`subscriptionId`	`SubscriptionId` of the action.
`description`	Description of the action.
`exception`	Exception description.
`taskId`	Id of a task of the action.
`taskType`	Type of the task with `taskId`.
`mainTaskId`	Main task of the task with `taskId`.
`threadId`	Task thread id of the task with `taskId`.
`jmsEventType`	JMS event type of the operation event.
jmsInvocationId	Unique ID.
`jmsEventSyncType`	Can be either "PULL" or "REALTIME"
`jmsEventId`	JMS event id of the operation event.
`jmsEventAdditionalData`	A map that describes parameters.
`jmsQueueType`	JMS queue type of the operation event.
`pullEventHandler`	`PullEventHandler` field (in case of a `PullTask`).
`endpointUrl`	Request endpoint URL.
`endpointInvocationId`	Unique ID.
endpointAdditionalData	A map that describes parameters (path variables and headers).
`user`	A user associated with the currently used credentials.
`recoveryTaskId`	An ID of Recovery Task.

`originalMainTaskId`	Original ID of the main task. This filed was added to preserve the main task ID during recovery.
`checkpoint`	Transaction event mark.
`initiatorUri`	`ObjectUri` associated with the operation.

ActionType

Value
`Copy`
`Autosubscribe`
`Create`
`Update`
`Index`
`MatchFound`
`MatchNotFound`
`Skip`
`MatchTableUpdate`
`Upload`
`CrosswalkRemoval`
`Unmerge`
`Split`
`MappingFilterInvocation`
`JmsEvent`
`Endpoint`
`EndpointInvoked`
`TaskProcessorStarted`
`Scan`
`Error`

SyncType

Value
Jms
Task
Pull
Endpoint

Full Import Log API

The standard Reltio Search syntax can be used to search for Actions.

Search Actions (Deprecated)

It is not recommended to use this feature (disabled by default). Please use Exporting Actions instead.

Request

```
GET {DTSSURL}/actions/es/search?filter=(equals(...))
```

Parameters

	Name	Required	Description	Examples
HEADERS	`CustomerTenant`	Yes	Customer Tenant id.	-
	`DataTenant`	Yes	Data Tenant id.	-

URI PARAMETERS	`filter`	No	Search by the query filter in Full Import Log (refer to Entity Search for details).
			
			Enables actions conditional filter with following format:
			
			`filter=({Condition Type}[AND/OR {Condition Type}]*)`
				`filter=equals(SubscriptionId, 'staticData_staticCustomer')`
				
				`filter=gt(Timestamp, 1129507200) and lt(Timestamp,1129507200)`
				
				`filter=(equals(ObjectUri,'00uwwbI'))`

	`max`	No	Positive Integer value to identify maximum number of objects to return in a response. Can be used to organize pagination in combination with "offset" parameter.	`max=10`
	`offset`	No	Positive Integer value to identify starting what element in a result set should be returned in a response. Can be used to organize pagination in combination with "max" parameter.	`offset=120`

Response

```
{
  "total": 2,
  "actions": [
    {
      "ActionId": "657245993fb94b829d87b84a2fd75eec",
      "ActionType": "Scan",
      "SyncType": "Task",
      "Success": true,
      "SubscriptionId": "staticData_staticCustomer",
      "Description": "Scan: CT object (entities/00ux0rY,
CustomerTenantId{id='staticCustomer'}) by ManualAutoSubscribeTask
(source: RangeUrisRetriever).",
      "TaskId": "9550464ae3bc400e8176ee414e2b17f0",
      "TaskType": "SUBSCRIBE",
      "MainTaskId": "0a8f030d31df49929616948032dea332",
      "ThreadId": "scanCtEntities",
      "Timestamp": 1484581853189,
      "ObjectUri": "00ux0rY",
      "TenantId": "staticCustomer",
      "TenantType": "CUSTOMER"
    },
    {
      "ActionId": "d26176358ddd42b9ac88e7422d0192d6",
      "ActionType": "Scan",
      "SyncType": "Task",
      "Success": true,
      "SubscriptionId": "staticData_staticCustomer",
      "Description": "Scan: CT object (entities/00ux57o,
CustomerTenantId{id='staticCustomer'}) by ManualAutoSubscribeTask
(source: RangeUrisRetriever).",
      "TaskId": "9550464ae3bc400e8176ee414e2b17f0",
      "TaskType": "SUBSCRIBE",
      "MainTaskId": "0a8f030d31df49929616948032dea332",
      "ThreadId": "scanCtEntities",
      "Timestamp": 1484581853189,
      "ObjectUri": "00ux57o",
      "TenantId": "staticCustomer",
      "TenantType": "CUSTOMER"
    }
  ]
}
```


Facet Search (Deprecated)

It is not recommended to use this feature (disabled by default). Please use Exporting Actions instead.

Request

```
POST {DTSSURL}/actions/_facets
```

Parameters

	Name	Required	Description	Examples
HEADERS	`CustomerTenant`	Yes	Customer Tenant id.	-
	`DataTenant`	Yes	Data Tenant id.	-
URI PARAMETERS	`Body`	No	Search for facets in Full Import Log (refer to Facet Search for details). Enables actions conditional filter with the following format: `filter=({Condition Type}[AND/OR {Condition Type}]*)`	`[ { "fieldName": "ActionId", "pageSize": 6, "pageNo": 1 } ]`

Response

```
{
  "ActionId": {
    "199ae7ae0ae142af90c89f8d8acaf2aa": 1,
    "1a4fe065d58549b089be58e9e39eff75": 1,
    "28c3c9a8dfdd48a8bab89bff8731e7a0": 1,
    "47c4caf916bb41e79219fc60042555f5": 1,
    "5d1f3be9eb7b46e3aaa208d7e83f1183": 1,
    "601f441b670e4af7b4b07285963e00e4": 1
  }
}
```

Reporting

Generates a CSV report S3 file for given date range (day, week, month or custom range defined as two timestamps), task ids and filter. Report contains two metrics (consumed entities per entity type and per user) and is generated for a date range provided by user or evaluated by default policy.

Report is generated by a Report Task. Once the task is finished, an email with the report CSV file is sent to the email provided in the request.

Request

```
GET {DTSSURL}/actions/report
```

Parameters

	Name	Required	Description
HEADERS	`CustomerTenant`	Yes	Customer Tenant id.
	`DataTenant`	Yes	Data Tenant id.
URI	`emails`	Yes	The list of emails to receive the report.

PARAMETERS	`taskIds`	No	Task ids for which to build report.
	`ranges` (DEPRECATED)	No	Date ranges to process. Deprecated: now please use Body.

	`filter`	No	Search by the query filter in Full Import Log (refer to Entity Search for details).
			
			Enables actions conditional filter with the following format:
			
			`filter=({Condition Type}[AND/OR {Condition Type}]*)`
			
			Examples:
			
			`filter=equals(SubscriptionId, 'staticData_staticCustomer')`
			
			`filter=gt(Timestamp, 1129507200) and lt(Timestamp,1129507200)`
			
			`filter=(equals(ObjectUri,'00uwwbI'))`

	`detailedReport` (DEPRECATED)	No	Whether DTSS should create detailed report that is uploaded to S3.
	`exportAction`	No	Default is `false`. If `true`, then actions are uploaded to S3.
	`distinctCounters`	No	Default is `false`. If `true`, then distinct counters are calculated and sent by email.
	`s3Bucket`	No	If used then S3 report will be put into custom S3 location. DTSS may not have access to the custom location so ticket to Support/DevOps should be created. After access to DTSS is granted the location can be used. Can be used only in combination (both `s3Bucket` and `s3Key` should be either present or absent).
	`s3Key`	No	
Body		Yes	Date ranges for which to build the report. Example: `[ { "from": "2015-09-22T01:22:12.143", "to": "2015-09-23T05:31:10.761" } ]`

Exporting Actions

Performs actions export for given date range (day, week, month or custom range defined as two timestamps), task ids and filter. Report contains two metrics (consumed entities per entity type and per user) and is generated for a date range provided by user or evaluated by default policy.

Report is generated by a Report Task. Once the task is finished an email with the report CSV file is sent to email provided in the request

Request

```
GET {DTSSURL}/actions
```

Parameters

	Name	Required	Description
HEADERS	`CustomerTenant`	Yes	Customer Tenant id.
	`DataTenant`	Yes	Data Tenant id.
URI PARAMETERS	`emails`	Yes	The list of emails to receive the report.
	`taskIds`	No	Task ids for which to build the report.
	`ranges` (DEPRECATED)	No	Date ranges to process. Deprecated. Please use BODY.

	`filter`	No	Search by the query filter in Full Import Log (more at EntitySearch).
			
			Enables actions conditional filter with following format:
			
			`filter=({Condition Type}[AND/OR {Condition Type}]*)`
			
			Examples:
			
			`filter=equals(SubscriptionId, 'staticData_staticCustomer')`
			
			`filter=gt(Timestamp, 1129507200) and lt(Timestamp,1129507200)`
			
			`filter=(equals(ObjectUri,'00uwwbI'))`

	`s3Bucket`	No	If used then S3 report will be put into custom S3 location. DTSS may not have access to the custom location so ticket to Support/DevOps should be created. After access to DTSS is granted the location can be used. Can be used only in combination (both s3Bucket and s3Key should be either present or absent).
	`s3Key`	No	
BODY		Yes	Date ranges for which to perform export. Example: `[ { "from": "2015-09-22T01:22:12.143", "to": "2015-09-23T05:31:10.761" } ]`

Reporting for Contract

Generates a CSV report S3 file for given contract id, date range (day, week, month or custom range defined as two timestamps) and filter. Report contains two metrics (consumed entities per entity type and per user) and is generated for a date range provided by user or evaluated by default policy.

Report is generated by a Report Task for Contract of ID provided in the request. Once the task is finished an email with the report CSV file is sent to email provided in the request

Request

```
POST {DTSSURL}/actions/report/contract
```

Parameters

	Name	Required	Description
	Name	**Required**	**Description**
Headers	CustomerTenant	Yes	Customer Tenant id
Headers	DataTenant	Yes	Data Tenant id
URI param	emails	Yes	The list of emails to receive the report.
URI param	contractId	Yes	Contract id to build report for
URI param	s3Bucket	No	If used then S3 report will be put into custom S3 location. DTSS may not have access to the custom location so ticket to Support/DevOps should be created. After access to DTSS is granted the location can be used. Can be used only in combination (both s3Bucket and s3Key should be either present or absent).
URI param	s3Key	No	

Body		Yes	Date ranges for which to to perform export. Example: `[ { "from": "2015-09-22T01:22:12.143", "to": "2015-09-23T05:31:10.761" } ]`

Periodic Reporting

Periodic reporting is done by the Periodic Reporting Task.

DTSS Rollover for Elastic Search Full Import Log index

Since 2018.1 release, DTSS keeps current Elastic Search Full Impot Log index size not more than <fullImportLog.es.rollover.maxShardSize> bytes per shard. Once this threshold is reached, DTSS rolls over to new ES FIL index without losing ability to search for actions stored in previous ES FIL index.

Following dtss properties parameters manage ES FIL rollover process:

es.rollover.enabled (default: true) - determines if automatic ES FIL index rollover is enabled

es.rollover.waitTimeout in milliseconds (default: 1200000 which is 20 minutes) - determines the period to check if threshold below is reached

fullImportLog.es.rollover.maxShardSize in bytes (default: 30*1024*1024*1024 which is 30GB) - ES FIL rollover automatically starts when the shard size reaches the size defined by this parameter (in Elastic Search 6, rollover automatically starts when the index (not shard) size reaches the size defined by this parameter).

It's possible to start ES FIL rollover manually by invoking following endpoint (for Admins only):

HTTP METHOD	POST
URL	`{DTSSURL}/fil/es/rollover?indicesGroups=dtss-full-import-log-index-default`
HEADERS	Content-Type: application/json Authorization: Bearer {oauth_token}

DTSS Service Overall Statistics

You can request information for the following statistics based on the duration you indicate in the request:

DTSS QUEUE health statistics

List of current DTSS sync tasks statistics

Endpoint

`GET /dtss/service/statistics?fromDateTimeMillis=1555921800000`

Headers:

List of data tenants: `DataTenant1`, `DataTenant2`

List of customer tenants: `CustomerTenant1`, `CustomerTenant2`

DTSS Monitoring

DTSS monitoring generates alerts to any potential issues so you can promptly resolve them to ensure DTSS is functioning properly.

`dtss.monitoring.enabled` (default = `false`): defines if the monitoring feature is enabled.

DTSS monitoring provides an overall status of the DTSS service, both currently operational and non-operational (down). Continual monitoring for reporting overall status means that you are informed of the following:

Load on the current DTSS service; that is, DTSS QUEUE health over a specified time period (last hour, last 4 hours, last 24 hours, last 48 hours, or last 5 days):

Number of sent messages

Number of received messages

Number of messages delayed

List of all current DTSS sync tasks/jobs that are currently in progress and completed (with errors and successfully) and metrics about each task, including:

Task name

Task configuration parameter

Entities/relationships processed successfully

Entities/relationships not processed

Start Date

End Date

Task duration (time it took to complete taskCurrent duration)

All statistics are delivered using a single endpoint

DTSS QUEUE Health Statistics

DTSS consumes messages from multiple sources such as PubSub, ActiveMQ, and SQS.

Stores the DTSS Queue message information such as the number of messages received and the number of messages processed successfully.

Collects the messages count for specified data and customer tenants and provides the DTSS queue statistics over a period of time such as the last hour, last 4 hours 30 minutes, last 2 days, and last 5 days 10 hours 30 minutes.

Saves all of the JMS messages received with their status as either processed or failed. Data is saved in the database (with `messageId`, `tenant Id`, `status`, and `lastUpdatedDateTime`).

An example endpoint to get the statistics for the past hour is
`GET /dtss/service/statistics?fromDateTimeMillis=1555921800000`

`"fromDateTimeMillis"` supports the statistics period as date/time in milliseconds values. You can provide a past date/time in milliseconds format to check the statistics.
Example: You need statistics for the past two days, and today is 04/22/2019 10:12:00, and two days previous is: 04/20/2019 10:12:00, which is equal to 1555735320000 Milliseconds. Use the Request endpoint, `GET /dtss/service/statistics?fromDateTimeMillis=1555735320000`

If `“fromDateTimeMillis”` is not provided, the system uses the minimum period value from DTSS properties, `“dtss.health.statistics.min.period.limit”` (default value is 1 day)

You can request a maximum period of `“dtss.health.statistics.max.period.limit.default”` (default value is 5 days).

Clean data for these conditions:

`DataTenantRemovedEvent`

`CustomerTenantRemovedEvent`

Records beyond the maximum statistics period. For example, if maximum period is five days, then the records beyond five days remain unused.

TTL is used to reduce the unwanted records.

Also, remove the records for a specific tenant when the above events are triggered.

List of Current DTSS Sync Tasks Statistics

A task history mechanism tracks tasks that are in progress or completed.

Monitoring collects all in-process or complete **parent** sync tasks for specific data and customer tenants.

Monitoring provides the statistics for all sync tasks for the time period based on the request parameter `fromDateTimeMillis`:

`manual_copy`

`manual_subscribe`

`manual_match`

`manual_bulk_copy_relations`

`manual_import_hierarchy`

`pull`

`periodic_pull`

If not provided in the request, the default value is used from the DTSS properties file.

You can request a maximum period statistic equal to the `dtss.health.statistics.max.period.limit.default` value (default is 5 days).

JSON Response (Completed Task)

JSON Response (Completed Task)


```
{
  "dtssQueueStatistics":{
     "numberOfMessageReceived":20,
     "numberOfMessageProcessed":10,
     "numberOfMessageFailed":10
  },
  "dtssSyncTaskStatistics":[
     {
        "id":"843a56b7-133c-4ca1-a339-3142246ce667",
        "name":"manual_sync",
        "status":"Completed",
        "configParameter":{
           "dtEntityTypes":[
              "configuration/entityTypes/HCP"
           ],
           "dtRelationTypes":[
           ],

           "entitiesFilter":{

           }
        },
        "entityProcessedSuccessfully":1,
        "relationshipsProcessedSuccessfully":0,
        "entityNotProcessed":0,
        "relationshipsNotProcessed":0,
        "stateDate":"2019-03-26T12:35:14.394+05:30",
        "endDate":"2019-03-26T12:35:18.783+05:30",
        "totalTimeToCompleteTask":"0:0:4",
        "subTaskIds":[
           "656585cd-9b77-497d-a662-ed88a9a3f43f",
           "b0ca607b-3a44-4839-9ea6-60002ebac7e8",
           "d66eefdc-75e7-43f9-9989-99cbd5d870be"
           ]
      }
]
}
```


JSON Response (In-progress Task)

JSON Response (In-progress Task)


```
{
  "dtssQueueStatistics":{
     "numberOfMessageReceived":20,
     "numberOfMessageProcessed":10,
     "numberOfMessageFailed":10
  },
  "dtssSyncTaskStatistics":[
     {
        "id":"683e0376-9f81-4a81-93e2-03315031ac9a",
        "name":"manual_sync",
        "status":"In progress",
        "configParameter":{
           "dtEntityTypes":[
              "configuration/entityTypes/HCP"
           ],
           "dtRelationTypes":[

           ],
           "entitiesFilter":{

           }
        },
        "entityProcessedSuccessfully":0,
        "relationshipsProcessedSuccessfully":0,
        "entityNotProcessed":0,
        "relationshipsNotProcessed":0,
        "stateDate":"2019-04-09T10:35:29.382+05:30",
        "subTaskIds":[

        ],
        "currentDuration":"0:0:1"
     }
]
}
```


OAuth Instances

OAuth instances are entities maintained by administrators which contain all credentials required for OAuth authentication (such as OAuth server URL, OAuth user and password, and Client ID and password). Regular users do not have direct access to them.

Instances are referenced by Customer or Data Tenants. To register a tenant, you must specify OAuth URL only. The appropriate OAuth instance is picked automatically and bound to the tenant. This type of mechanism allows sensitive information (such as passwords) to be kept and closed to regular users, thereby, improving security.

Model

OAuth Instance Model

OAuth Instance Model

```
{
 "oauth_url": URL of OAuth server,
   "oauth_user": OAuth login,
   "oauth_password": OAuth password,
   "client_id": Client ID,
   "client_password": Client password,
    "support_email": Support email
   //Audit fields omitted: created_date, created_by, updated_date,
updated_by
}
```

Endpoints

Create/Update Instance

HTTP METHOD	POST
URL	`{DTSSURL}/admin/oauth/instance`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`

BODY	**Example**
	`{`
	` "oauth_url": "https://auth-stg-dev.reltio.com/oauth/token",`
	`  "oauth_user": "MyUser",`
	`  "oauth_password": "MyPassword",`
	`  "client_id": "MyClientID",`
	`  "client_password": "MyClientPassword"`
	`  "support_email": "example@mail.com"`
	`}`

RESPONSE

Example: OK

```
{
    "successful": true,
    "result": {
        "oauth_url":
"https://auth-stg.reltio.com/oauth/token",
        "oauth_user": "MyUser",
        "oauth_password": "MyPassword",
        "client_id": "MyClientID",
        "client_password": "MyClientPassword",
        "created_date": 1458153917789,
        "updated_date": 1458153917789,
        "created_by": "MyReltioUser",
        "updated_by": "MyReltioUser",
        "support_email": "example@mail.com"
    }
}
```

Example: Bad Request

```
{
    "validationResults": [
        {
            "checkName": "Oauth endpoint check for instance
https://auth-stg.reltio.com/oauth/token",
            "result": {
                "status": "FAILED",
                "message": "Unable to obtain token. Got an
error [400]."
            }
        }
    ],
    "successful": false
}
```

Get Single Instance

HTTP METHOD	`GET`
URL	`{DTSSURL}/admin/oauth/instance?useCache={boolean}`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`
PARAMETERS	`oauth_url: {oauth_server_url}`

RESPONSE	`{Model}` or `404` error

Get All Instances

HTTP METHOD	`GET`
URL	`{DTSSURL}/admin/oauth/instance?useCache={boolean}`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`
RESPONSE	Collection of `{Model}`

Delete Instance

Note: Only non-referenced instances (no registered tenants using it) can be deleted. See Get Related Tenants. A corresponding error is returned after an attempt to remove this type of instance.

HTTP METHOD	`DELETE`
URL	`{DTSSURL}/admin/oauth/instance`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`
PARAMETERS	`oauth_url: {oauth_server_url}`

RESPONSE	**Example**
	`{`
	`  "successful": true,`
	`  "result": "https://auth-stg.reltio.com/oauth/token"`
	`}`

Verify Instance

HTTP METHOD	`GET`
URL	`{DTSSURL}/admin/oauth/instance/verify`
HEADERS	Content-Type: `application/json` Authorization: `Bearer {oauth_token}`
PARAMETERS	`oauth_url: {oauth_server_url}`

RESPONSE	**Example**
	[
	{
	"checkName": "Oauth endpoint check for instance https://auth-stg.reltio.com/oauth/token",
	"result": {
	"status": "SUCCESS"
	}
	}
	]

Get Related Tenants

Returns all of the customer and data tenants that reference the specified OAuth instance.

HTTP METHOD	GET
URL	{DTSSURL}/admin/oauth/instance/tenants
HEADERS	Content-Type: application/json Authorization: Bearer {oauth_token}
PARAMETERS	oauth_url: {oauth_server_url}

RESPONSE	**Example**
	{
	"customerTenants": [{Tenants}],
	"dataTenants": [{Tenants}]
	}

Get Modification History

Get OAuth Instance Modifications

HTTP METHOD	GET
URL	{DTSSURL}/admin/oauth/instance/history
HEADERS	Content-Type: application/json Authorization: Bearer {oauth_token}
PARAMS	oauth_url: {oauth_server_url}