



**DISCOVER  
THE NEW WAY  
FOR YOUR ENTERPRISE  
DATA**

# **DTS Product Manual**

**© 2023 Realworld Systems B.V.**



<b>1. Introduction</b>	<b>9</b>
1.1 Copyrights .....	10
1.2 Welcome .....	12
1.3 Manual Version .....	13
1.4 Release Notes .....	13
<b>2. Deployment</b>	<b>15</b>
2.1 Prerequisites .....	17
2.2 Basic Deployment .....	18
<b>3. Web UI</b>	<b>25</b>
3.1 Login .....	26
3.2 Workspace .....	27
3.2.1 Home .....	28
3.2.1.1 Project Info Dialog .....	29
3.2.1.2 Add New Project .....	30
3.2.2 Project .....	30
3.2.2.1 Project Information .....	31
3.2.2.2 Project Notifications .....	32
3.2.2.3 Project Dashboard .....	34
3.2.2.4 Published Projects .....	40
3.2.2.5 All Projects .....	41
3.2.3 Sources .....	41
3.2.3.1 Connectors Drawer .....	42
3.2.3.1.1 Create New Connector .....	44
3.2.3.1.2 Connector Details .....	47
3.2.3.2 Assets Drawer .....	49
3.2.3.3 Asset Details Drawer .....	51
3.2.3.3.1 Collection Details Drawer .....	52
3.2.3.3.2 Routine Details Drawer .....	55
3.2.3.3.3 Topic Details Drawer .....	57
3.2.4 Aggregates .....	59
3.2.4.1 Aggregates Drawer .....	60
3.2.4.1.1 Create New Aggregate .....	61
3.2.4.1.2 Aggregate Details .....	63
3.2.4.2 Aggregate Assets .....	64
3.2.4.2.1 Add New Aggregate Source .....	65
3.2.4.3 Aggregate Asset Details Drawer .....	66
3.2.5 Webservices .....	70
3.2.5.1 Webservices Drawer .....	71
3.2.5.1.1 Create New Webservice .....	72
3.2.5.1.2 Webservice Details .....	73

3.2.5.2	Websocket Assets Drawer .....	76
3.2.5.3	Websocket Asset Details Drawer .....	78
3.2.5.3.1	Websocket Stream Operations Drawer .....	79
3.2.5.3.2	Websocket Routine Details Drawer .....	86
3.2.5.3.3	Websocket Topic Details Drawer .....	89
3.2.6	Left-Side Menu Toolbar .....	94
<b>3.3</b>	<b>Top Menu Toolbar .....</b>	<b>95</b>
3.3.1	Preferences Menu .....	96
3.3.1.1	Websocket Deployers .....	97
3.3.1.2	Connector Types .....	104
3.3.1.3	Notification Senders .....	109
3.3.2	User Menu .....	111
<b>3.4</b>	<b>Filters &amp; Relationships .....</b>	<b>111</b>
<b>3.5</b>	<b>Errors &amp; Warnings .....</b>	<b>114</b>
<b>4.</b>	<b>Connectors .....</b>	<b>117</b>
<b>4.1</b>	<b>Apache Kafka .....</b>	<b>118</b>
4.1.1	Connection Parameters .....	119
4.1.2	Types .....	119
4.1.3	Topics .....	120
4.1.4	Limitations .....	120
<b>4.2</b>	<b>MariaDB .....</b>	<b>121</b>
4.2.1	Connection Parameters .....	121
4.2.2	Types .....	122
4.2.3	Geometry .....	123
4.2.4	Tables and Views .....	125
4.2.5	Routine Calls .....	125
4.2.6	Limitations .....	126
<b>4.3</b>	<b>MS SQL Server .....</b>	<b>127</b>
4.3.1	Connection Parameters .....	127
4.3.2	Types .....	128
4.3.3	Geometry .....	129
4.3.4	Tables and Views .....	131
4.3.5	Routine Calls .....	131
4.3.6	Limitations .....	133
<b>4.4</b>	<b>MySQL .....</b>	<b>133</b>
4.4.1	Connection Parameters .....	134
4.4.2	Types .....	134
4.4.3	Geometry .....	135
4.4.4	Tables and Views .....	137
4.4.5	Routine Calls .....	138

4.4.6	Limitations .....	139
<b>4.5</b>	<b>Oracle .....</b>	<b>139</b>
4.5.1	Connection Parameters .....	140
4.5.2	Types .....	140
4.5.3	Geometry .....	141
4.5.4	Tables and Views .....	143
4.5.5	Routine Calls .....	144
4.5.6	Limitations .....	145
<b>4.6</b>	<b>PostgreSQL .....</b>	<b>146</b>
4.6.1	Connection Parameters .....	147
4.6.2	Types .....	147
4.6.3	Geometry .....	149
4.6.4	Tables and Views .....	151
4.6.5	Routine Calls .....	152
4.6.6	Limitations .....	153
<b>4.7</b>	<b>SAP Hana .....</b>	<b>153</b>
4.7.1	Connection Parameters .....	154
4.7.2	Types .....	154
4.7.3	Geometry .....	155
4.7.4	Tables and Views .....	156
4.7.5	Routine Calls .....	157
4.7.6	Limitations .....	158
<b>4.8</b>	<b>Smallworld .....</b>	<b>159</b>
4.8.1	Connection Parameters .....	160
4.8.2	Smallworld Environment .....	160
4.8.3	Types .....	161
4.8.4	Geometry .....	163
4.8.5	Collections .....	164
4.8.6	Routine Calls .....	166
4.8.6.1	Registration .....	166
4.8.6.2	Stream Results .....	169
4.8.6.3	Registration Example .....	170
4.8.7	Smallworld Client .....	176
4.8.8	Limitations .....	181
<b>4.9</b>	<b>Web Service .....</b>	<b>181</b>
4.9.1	Connection Parameters .....	182
4.9.2	Types .....	182
4.9.3	Routine Calls .....	183
4.9.4	Limitations .....	184

## 5. Webservices 185

<b>5.1</b>	<b>Types .....</b>	<b>186</b>
<b>5.2</b>	<b>Functionality .....</b>	<b>187</b>
<b>5.3</b>	<b>Specification .....</b>	<b>195</b>
<b>5.4</b>	<b>Access .....</b>	<b>195</b>
<b>5.5</b>	<b>Integration .....</b>	<b>199</b>
5.5.1	Application Server .....	199
5.5.2	External Security and Load Balancing .....	201
5.5.3	Logging .....	203
<b>5.6</b>	<b>Limitations .....</b>	<b>206</b>
<b>6.</b>	<b>Technical Guide .....</b>	<b>207</b>
<b>6.1</b>	<b>Architecture .....</b>	<b>208</b>
6.1.1	Controller .....	210
6.1.2	GUI Controller .....	211
6.1.3	Producer .....	212
6.1.4	Client .....	213
6.1.4.1	Webservice Client .....	214
6.1.4.2	Smallworld Client .....	215
6.1.5	Aggregator .....	215
6.1.6	Agent .....	216
6.1.7	GUI .....	217
6.1.8	CLI .....	217
<b>6.2</b>	<b>Communication .....</b>	<b>218</b>
<b>6.3</b>	<b>Types .....</b>	<b>218</b>
<b>6.4</b>	<b>Streams .....</b>	<b>220</b>
<b>6.5</b>	<b>Geometry .....</b>	<b>222</b>
6.5.1	Structure .....	223
6.5.2	Corrections .....	224
6.5.3	Examples .....	226
6.5.3.1	Simple Point .....	227
6.5.3.2	Oriented Point .....	227
6.5.3.3	Multi Point .....	228
6.5.3.4	Annotation .....	228
6.5.3.5	Simple Line String .....	229
6.5.3.6	Arc Line String .....	229
6.5.3.7	Compound Line String .....	230
6.5.3.8	NURBS .....	231
6.5.3.9	Multi Line String .....	232
6.5.3.10	Polygon .....	233
6.5.3.11	Compound Polygon .....	234
6.5.3.12	Rectangle .....	235

6.5.3.13	Circle .....	236
6.5.3.14	Multi Polygon .....	237
6.5.3.15	Geometry Collection .....	238
<b>6.6</b>	<b>Predicate .....</b>	<b>239</b>
<b>6.7</b>	<b>Aggregation .....</b>	<b>242</b>
<b>6.8</b>	<b>Security .....</b>	<b>246</b>
6.8.1	Outline .....	247
6.8.2	Registration and Authentication .....	248
6.8.3	Setup .....	252
<b>7.</b>	<b>Development .....</b>	<b>255</b>
<b>7.1</b>	<b>Client .....</b>	<b>256</b>
7.1.1	Pre-Built .....	257
7.1.2	Java Library .....	257
7.1.2.1	Direct Usage Example .....	259
7.1.2.2	Custom Implementation Usage .....	263
7.1.3	.NET Library .....	266
7.1.3.1	Direct Usage Example .....	267
7.1.3.2	Custom Implementation Example .....	270
7.1.4	Redis/JSON .....	274
7.1.4.1	Registration Command .....	278
7.1.4.2	Project Command .....	280
7.1.4.3	Connector Command .....	282
7.1.4.4	Record Stream Command .....	284
7.1.4.5	Execute Remote Command .....	286
<b>7.2</b>	<b>Producer .....</b>	<b>288</b>
<b>7.3</b>	<b>Project Artifacts .....</b>	<b>288</b>
7.3.1	CONNECTOR_CATEGORY .....	290
7.3.2	CONNECTOR_TYPE .....	291
7.3.3	NOTIFICATION_SENDER .....	293
7.3.4	WEBSERVICE_DEPLOYER .....	294
7.3.5	PROJECT_WIP .....	295
7.3.6	PROJECT_RESOURCE .....	296
7.3.7	PROJECT_OPERATION .....	297
7.3.8	CONNECTOR .....	298
7.3.9	COLLECTION_RESOURCE .....	300
7.3.10	COLLECTION_DETAILS .....	301
7.3.11	REMOTE_CALL_RESOURCE .....	302
7.3.12	REMOTE_CALL_DETAILS .....	303
7.3.13	TOPIC_RESOURCE .....	304
7.3.14	TOPIC_DETAILS .....	305
7.3.15	AGGREGATE .....	306

7.3.16	WEBSERVICE .....	307
7.3.17	PROJECT .....	308
7.3.18	PROJECT_STATUS .....	309
7.3.19	PRODUCER_STATUS .....	310
7.3.20	CONSUMER_STATUS .....	311
7.3.21	NotificationConfig .....	313
7.3.22	NotificationTarget .....	313
7.3.23	AttributeDescriptor .....	314
7.3.24	ConstraintDefinition .....	315
7.3.25	TypeResource .....	316
7.3.26	AggregateSource .....	316
7.3.27	AggregateRelationship .....	317
7.3.28	WebresourceResource .....	318
<b>8.</b>	<b>Known Limitations</b>	<b>321</b>
<b>9.</b>	<b>Licenses</b>	<b>323</b>
<b>9.1</b>	<b>3rd Party Licenses .....</b>	<b>324</b>
9.1.1	Apache 2.0 .....	325
9.1.2	BSD 2-clause .....	329
9.1.3	BSD 3-clause .....	329
9.1.4	CCO 1.0 .....	330
9.1.5	CDDL 1.1 .....	330
9.1.6	GPL 2.0 .....	336
9.1.7	LGPL 2.1 .....	341
9.1.8	MIT .....	348
9.1.9	OTNLA .....	349
9.1.10	SAP Developer License Agreement .....	354
<b>Index</b>		<b>0</b>



# Introduction

What is DTS, versions and copyrights

## 1 Introduction



# DATA TRANSIT SYSTEM



**Version 2023.1.1**

Copyright © 2023 Realworld Systems B.V.. All rights reserved.

### 1.1 Copyrights

---

#### Copyright information

© 2023 Realworld Systems B.V.. All rights reserved.

The Software Product described in this documentation may only be used strictly in accordance with the applicable written License Agreement. The Software Product and Associated Documentation are deemed to be “commercial computer software” and “commercial computer software”

documentation,” respectively, pursuant to DFAR Section 227.7202 and FAR Section 12.212, as applicable, and are licensed with Restricted Rights as identified in the License Agreement, and as set forth in the “Restricted Rights Notice” contained in paragraph (g) (3) (Alternate III) of FAR 52.227-14, Rights in Data - General, including Alternate III (June 1987).

The information contained in this online publication is the exclusive property of Realworld Systems B.V., except as otherwise indicated. You may view, copy and print documents and graphics incorporated in this online publication (the “Documents”) subject to the following: (1) the Documents may be used solely for personal, informational, non-commercial purposes; (2) the Documents may not be modified or altered in any way; and (3) Realworld Systems B.V. withholds permission for making the Documents or any portion thereof accessible via the Internet. Except as expressly provided herein, you may not use, copy, print, display, reproduce, publish, licence, post, transmit or distribute the Documents in whole or in part without the prior written permission of Realworld Systems B.V..

The information contained in this online publication is subject to change without notice.

The software described in this online publication is supplied under license and may be used or copied only in accordance with the terms of such license.

The licenses can be found in the [Licenses](#)<sup>324</sup> annex of this document.

## Trademarks and Registered Trademarks

Apache is a registered trademark or trademark of the Apache Software Foundation in the United States and/or other countries.

Docker and the Docker logo are trademarks or registered trademarks of Docker, Inc. in the United States and/or other countries. Docker, Inc. and other parties may also have trademark rights in other terms used herein.

JBoss is a registered trademark of Red Hat Inc., in the United States and other countries.

Java, WebLogic, Oracle, Oracle Spatial, and Oracle Express are trademarks of Oracle Corporation, registered in the United States and other countries.

Microsoft and SQL Server are registered trademarks of Microsoft Corporation in the United States and other countries.

Mongo and MongoDB are registered trademarks of MongoDB, Inc. in the United States and other countries.

Postgres and PostgreSQL are registered trademarks of the PostgreSQL Community Association of Canada.

Redis is a trademark of Redis Labs Ltd. Any rights therein are reserved to Redis Labs Ltd. Any use by Realworld Systems B.V. is for referential purposes only and does not indicate any sponsorship, endorsement or affiliation between Redis and Realworld Systems B.V..

Smallworld and GE Smallworld are trademarks of the General Electric Company.

WebSphere is a trademark of International Business Machines("IBM") Corporation, registered in the United States and other countries.

Other company or product names mentioned in this documentation may be trademarks or registered trademarks of their respective companies.

## 1.2 Welcome

---

### About

DTS is a distributed middleware solution for providing real-time access to data and functionality from various sources and making it available in multiple forms for consumption and execution.

This manual seeks to present and explain all user, administration and development aspects of DTS and serve as a reference when deploying, configuring and expanding it.

### Audience and required skills

This manual is intended for System Administrators, End Users and Developers who will be interacting with DTS.

The required skill set varies depending on the section. Wherever external knowledge is required, it will be so noted.

In general, the following skills will be helpful:

Role	Tasks	Skills
System Administrator	Deployment and Maintenance	Bash command line, Docker, Kubernetes, Nginx
End User	Project Creation and Management, Day-to-day Operations	Knowledge of the systems that will directly interact with DTS
Developer	Custom Endpoint Development	Java / .NET, Redis, MongoDB

### Distribution forms

PDF, HTML

## Support

For issues not covered by the manual or any extra information regarding the deployment, configuration and use of our product, please contact us by email at [info@datatransitsystem.com](mailto:info@datatransitsystem.com) or through your sales representative.

### 1.3 Manual Version

---

This section contains version information regarding this document. Version information regarding the product itself is available under [Release Notes](#) <sup>13</sup>.

Version	Date	Description
2023.1.1	October 2023	2023.1 Release Manual
2022.1.1	December 2022	2022.1 Release Manual
2021.1.2	January 2022	2021.1 Final Release Manual
2021.1.1	July 2021	2021.1 Beta Release Manual

### 1.4 Release Notes

---

Version	Information
2023.1	<ul style="list-style-type: none"><li>• Apache Kafka Connector</li><li>• MariaDB Connector</li><li>• Improved Kubernetes Integration</li><li>• UI Updates</li><li>• Engine Optimizations</li></ul>
2022.1	<ul style="list-style-type: none"><li>• .NET Client API</li><li>• Administration CLI</li></ul>

	<ul style="list-style-type: none"><li>• MySQL Connector</li><li>• UI Updates</li><li>• Engine Optimizations</li><li>• Webservice extensions</li></ul>
<b>2021.1</b>	<ul style="list-style-type: none"><li>• Dashboard</li><li>• Aggregation</li><li>• Notifications</li><li>• Routine Streaming</li><li>• Collection Filters</li><li>• SAP Hana Connector</li><li>• Field Name Customization</li><li>• UI Updates</li><li>• Bug Fixes</li></ul>
<b>2020.2</b>	First commercial release



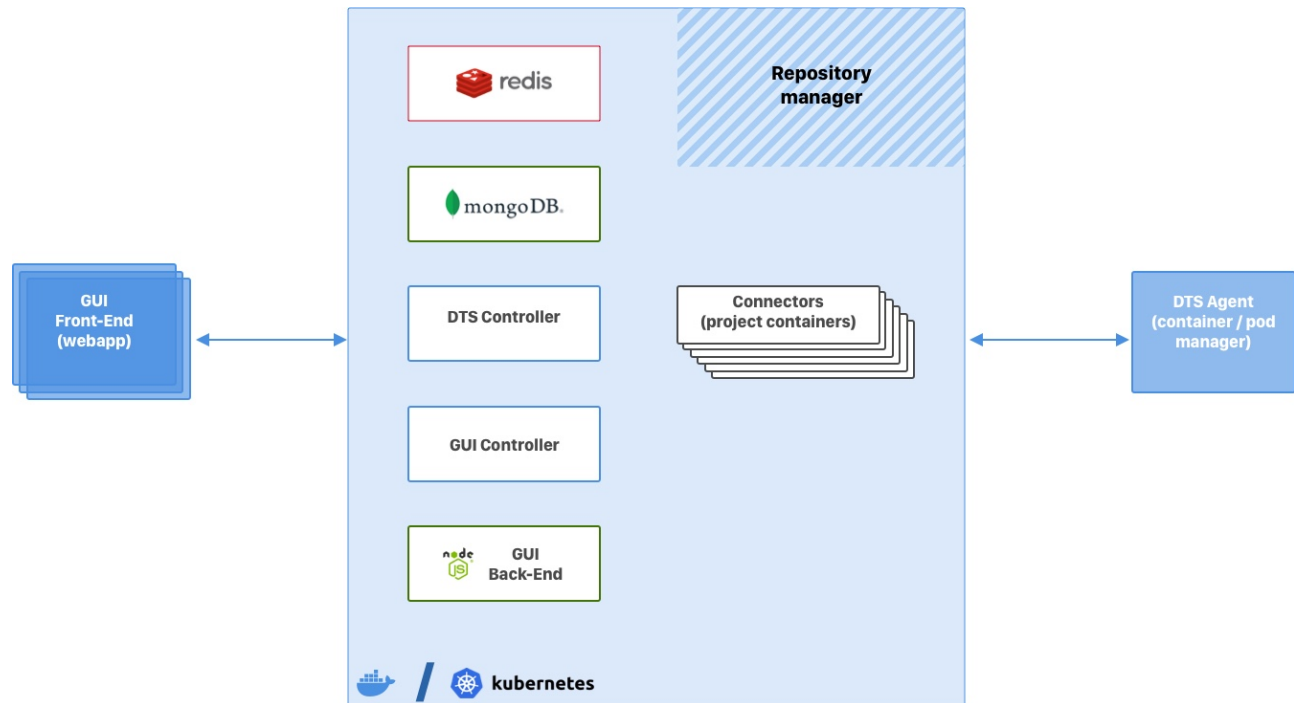
# Deployment

Administrative guide for deploying DTS

## 2 Deployment

DTS is a very flexible software product, built on a distributed architecture and can be deployed in a number of ways, depending on usage patterns and environmental limitations.

However, the preferred deployment scheme for DTS involves containerization and clustering of the various components using solutions like Docker™ and Kubernetes®.




While Docker is required as a container platform, Kubernetes is optional as DTS also provides its own minimal orchestration solution in the form of the [DTS Agent](#)<sup>[216]</sup>.

Orchestrated deployment allows DTS to self-regulate, automatically scale, provide redundancy and guarantee uptime. For this reason, the orchestrated container deployment is the only type of deployment we support in production environments.

**i** For more information on clustered deployment, please see [Prerequisites](#)<sup>[17]</sup>.

**i** The most turnkey type of DTS deployment uses the Agent and docker-compose. Please see [Basic Deployment](#)<sup>[18]</sup> for more information.

**i** For more information on the various DTS components and their containment, please see [Architecture](#)<sup>[208]</sup>.

 For information regarding other types of deployment or special scenarios, please contact us.

## 2.1 Prerequisites

---

This section summarizes the prerequisites specific to the containerized deployment of DTS.

### Hardware

DTS can be deployed on physical as well as virtual machines. The required resources are highly dependent on the scale of the deployment (i.e. the planned producer count and types) and the orchestration choice.

 Please see [Basic Deployment](#)<sup>[18]</sup> for a basic installation's requirements or contact us for help in sizing your own hardware resources.

### Operating System

While DTS can be theoretically be deployed on any Operating System that can run Docker™, we supply all components as Linux®-based images and do the bulk of integration testing on Linux machines, so a Linux OS is recommended.

Windows™ environments are also fully supported, each release being tested and validated against them. However, running in DTS component containers in Windows™ requires an extra layer of virtualization (WSL 2) and may incur a performance penalty.

 In the specific case of using DTS with Smallworld® 4.3 producers, we recommend CentOS™ 7 as a host OS as well as a base for the Smallworld producer images.


 When deploying in Kubernetes®, the platform's own OS restrictions apply, as well as the above guidelines.

### Supported platforms

Kubernetes® is used to manage a cluster of DTS producer containers. To meet the requirements of a secure DTS deployment, we support Kubernetes running on a Linux® distribution. This determines the supported platforms for DTS deployments as summarized here.

### Third-party products

DTS requires the following third-party products:


Product	Note
Docker™	DTS requires at least version 18.01, but 20+ is recommended.
Kubernetes®	DTS requires at least version 1.19.  <b>Not required when deploying with DTS Agent orchestration</b>
Java	Oracle JDK 8 or OpenJDK 8 is required for running the DTS <a href="#">Agent</a> <sup>216</sup> and <a href="#">CLI</a> <sup>217</sup>

The [Kubernetes documentation website](#) and [Docker desktop website](#) include download and release information, as well as reference information and tutorials.

## 2.2 Basic Deployment

The easiest way to deploy DTS for testing or production purposes is in a Docker-only environment using Docker Compose for easy control and configuration and the [DTS Agent](#)<sup>216</sup> for orchestration.

This section outlines the requirements and steps involved in setting up such a DTS deployment.

 For information regarding integrating DTS into new or existing orchestrated clusters (e.g. Kubernetes), please contact us.

### Hardware

A DTS standard (minimal) host machine spec is defined as:

- 2 CPUs
- 16 GB RAM
- 100 GB Hard Drive
- 1 Network Card

The recommended host machine spec can vary greatly and is defined by the type and numbers of containers expected to be running.

## Host Operating System

Any Windows, Linux or Macintosh operating system that can run a supported Java version and meets the [Docker Minimum Requirements](#) will work. Other operating systems may work, but they are not tested by DTS. The most widely used operating system for Data Transit System is Linux and therefore customers should consider it the best tested platform.

## Java

The [Data Transit System agent](#)<sup>216</sup> will run outside the Docker network (without a container) and requires a Java 8 Runtime Environment (JRE).

The distributions for OSX and Windows include suitable runtime environments for the specific operating system.

The distributions for Unix do not include the runtime environment.

If you prefer to use an external runtime or use a Unix operating system, you can choose to install the full JDK or the JRE only.

You can confirm the installed Java version with the **java -version** command, for example:


```
$ java -version
openjdk version "1.8.0_191"
OpenJDK Runtime Environment (build 1.8.0_191-b12)
OpenJDK 64-Bit Server VM (build 25.191-b12, mixed mode)
```

When multiple JDK or JRE versions are installed, you need to ensure the correct version is configured by running the above command as the operating system user that is used to run the DTS agent.

In the event you have a non-standard location, you need to update the configuration to specify a specific JDK or JRE installation path.

## Docker and Docker Compose

Docker is the containment engine of choice for DTS and is the only one the product is tested with. An appropriate Docker Engine should be installed on the machine.

 For information on how to install Docker Engine on a specific host OS, please see the [Docker Engine installation instructions](#)

 DTS is supported on Docker versions starting with version 18.01 but the latest version of Docker is advised for all customers

The most efficient way to configure an entire DTS deployment for a quick start of testing procedures is using Docker Compose. This should also be installed on the machine.

**i** For information on how to install Docker Compose on a specific host OS, please see the [Docker Compose installation instructions](#).

## Docker Images Repository

For easy access to the Docker images specific to the deployment, we recommend creating a bespoke repository.

This repository can run on the same host machine within a Docker container or can be installed on a different machine or even in the cloud.

**i** For local installations, [Nexus Repository OSS](#) is the most common option

**i** The initial DTS images and subsequent updates can be pulled from our central repository and pushed into the local one

## Ports

A standard test deployment requires the following ports to be opened on the host machine:

- The GUI Webapp port (configurable - default is 7875).
- The ICB (Redis) port (configurable - default is 7877).
- The SSH port (22).
- [Optional] For remote debugging, the Artifact Datastore (MongoDB) port (configurable - default is 7879).

## Deployment and Configuration

The functional deployment of DTS for testing purposes consists of a directory containing the following:

<b>agent</b>	Home of the <a href="#">DTS Agent</a> <sup>[216]</sup> component - responsible interfacing the session management system with the container/cluster platform (in this case Docker).
<b>blades</b>	Directory for storing the system's <a href="#">data blades</a> <sup>[118]</sup> (libraries used for connecting to various data sources).
<b>cli</b>	Directory for the <a href="#">DTS CLI</a> <sup>[217]</sup> binaries (portable).
<b>licence</b>	Directory for storing the product licence certificates.
<b>security</b>	Home of the DTS certificate authority - includes scripts for setting up the security system and generating certificates for new components.
<b>wss</b>	Home of the DTS Web Service stacks, which includes the tools, libraries and work directories used by the system to consume and generate Web Services.

<b>docker-compose.yml</b>	The centralized docker-compose configuration for the DTS core containers.
<b>dts-start.sh</b>	Script to start the entire system.
<b>dts-stop.sh</b>	Script to stop the entire system.

The configuration of the deployment is done in two places:

- **docker-compose.yml** is used to set up host paths and ports, as well as certain environment variables used to configure the behavior of core components.
  - Each container definition under **services** has an **image** parameter which should point to the name of the image to be used by the container (as known by Docker).
  - The **ports** parameter under the **dts-redis**, **dts-mongo** and **dts-webapp** container definitions allow customizing the mappings of the respective access ports on the host (left side value).
  - The **dts-mongo** container can be configured to map the **mongodata volume** to a custom directory on the host if desired (e.g. for backup purposes). If not mapped, the project data will be stored in Docker's own storage pattern.
  - The **dts-controller** container maps the **licence** directory under the **volumes** parameter and can be altered for a custom placement of the licence directory on the host.
  - The **dts-gui-controller** container maps the **blades** and **wss** directories from the host and can be altered for custom placements of the respective directories.
  - The **environment** parameter for each container should generally remain unmodified, but certain environment variables can be added:
    - The **TZ** environment variable can be defined to enforce a specific timezone on each container (e.g.: `- TZ=EEST`).
    - **DTS\_DEBUG\_LOGGING** can be added to any container to activate its verbose output mode for debugging purposes (e.g.: `- DTS_DEBUG_LOGGING=true`).
- **agent/start-agent.sh** is used to set up the Agent's operation mode, connection points and local paths.
  - **DTS\_AGENT\_MODE** can be **DOCKER** or **K8S** and determines whether Kubernetes or Docker will be the platform used. In the case of the standard test deployment, we use Docker: `export DTS_AGENT_MODE=DOCKER`
  - **DTS\_CLUSTER\_HOST** is the URL that the agent will use to send commands to the container/cluster platform. In the case of the standard test deployment, we use Docker and run the agent on the same machine, so we use Docker's default socket: `export DTS_CLUSTER_HOST=unix:///var/run/docker.sock`

- **DTS\_REDIS\_HOST\_NAME** is the name or address of the machine that provides access to the Redis communication platform. In the case of the standard test deployment, it should be **localhost** or the IP address of the host machine.
- **DTS\_REDIS\_PORT** is the port that provides access to the Redis communication platform. In the case of the standard test deployment, this should be the host port mapped in **docker-compose.yml** for the **dts-mongo** container.
- **DTS\_HOST\_BLADES\_DIR** is the absolute path to the **blades** directory of the deployment.
- Optionally, a default producer image can be defined using **DTS\_PRODUCER\_IMAGE** (which will be used if none is provided in a given connector's definition).
- Also optionally, the **DTS\_DEBUG\_LOGGING** variable can be set to true to enable verbose output from the agent for debug purposes. This option is also passed on to all producer containers that the agent starts.

## Interaction

After deployment, the entire system can be started with a single command:

```
<dts_dir> $ ./dts-start.sh
```

The system can also be shut down with a single command:

```
<dts_dir> $ ./dts-stop.sh
```

The DTS core components can be started without agent support using:

```
<dts_dir> $ [sudo] docker-compose up -d
```

The DTS core components can also be shutdown independently using:

```
<dts_dir> $ [sudo] docker-compose down
```

The agent can be started individually using:

```
<dts_dir>/agent $ ./start-agent.sh
```

The agent can also be stopped individually using:

```
<dts_dir>/agent $ ./stop-agent.sh
```

Logs for various containers can be accessed using this command:

```
<dts_dir> $ [sudo] docker logs --tail=<n_lines> <container_name>
```

Logs for the current (or last run) agent instance can be found in the

```
<dts_dir>/agent/agent.log file.
```

## User Permissions

In order for the **user** which will interact with DTS on the host machine to be able to invoke all the necessary commands and pass enough authority to the agent to perform its tasks, the following are required:

- The **user** should have full **rwX** permissions on all files in the DTS deployment listed above in the Deployment and Configuration section.
- For the start and stop scripts, as well as the agent to be able to control Docker, either:
  - The **user** should be part of the **docker** group on the host machine.  
or
  - The **user** should have **sudo** permission for the **docker**, **docker-compose** and **java** commands (with **SETENV** and, optionally but preferably **NOPASSWD**).
- For the agent stop script to be able to kill the agent process, either:
  - The **user** should have **sudo** permission for the **kill** command.  
or
  - The **user** should have **sudo** permission to run the **<dts\_dir>/agent/stop-agent.sh** script, on which write access for the user can be removed to ensure no alterations are made.







# Web UI

Features and workflow using the DTS Web UI application

## 3 Web UI

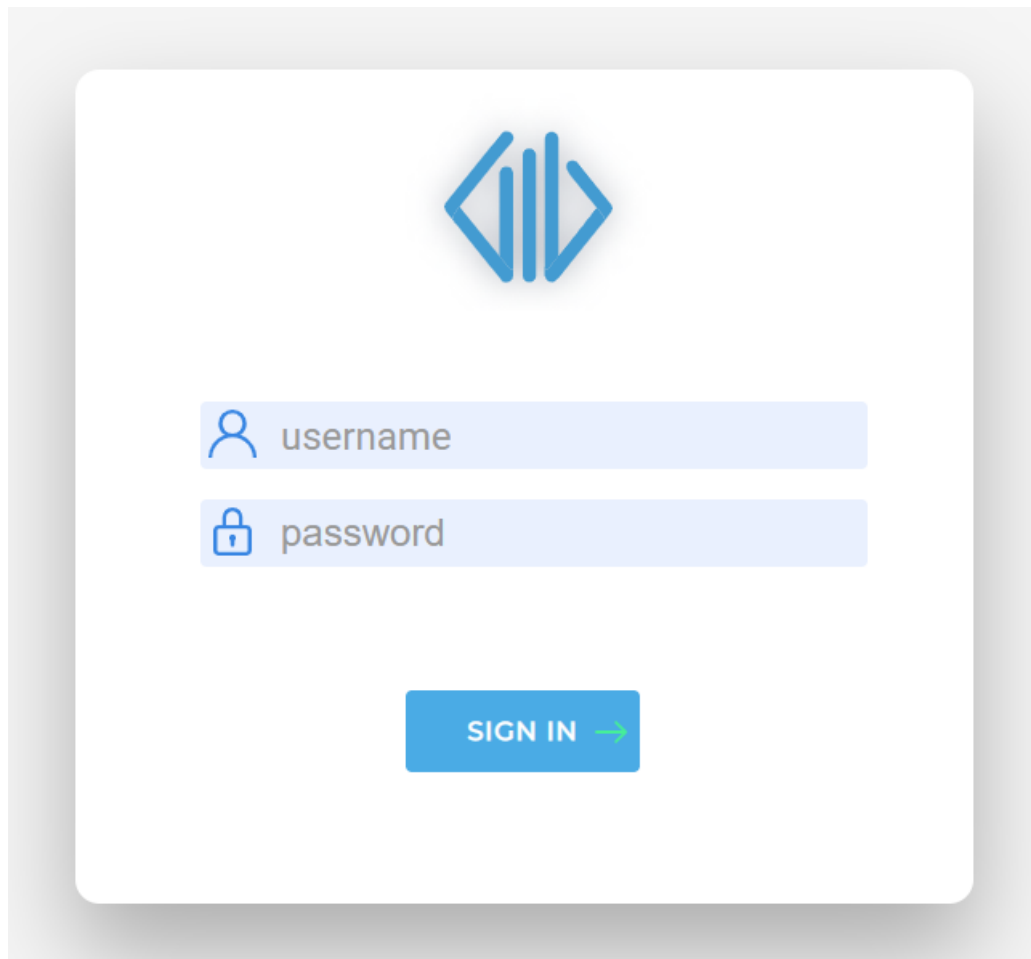
This section describes the various views and controls in the Web UI, detailing their usage and provides a reference for project workflows.

- [Login](#)  26
- [Workspace](#)  27
- [Left-Side Menu Toolbar](#)  94
- [Top Menu Toolbar](#)  95
- [Errors & Warnings](#)  114
- [Filters & Relationships](#)  111

### 3.1 Login

---

The login form consists of the DTS logo, a username field, a password field, and the login button.

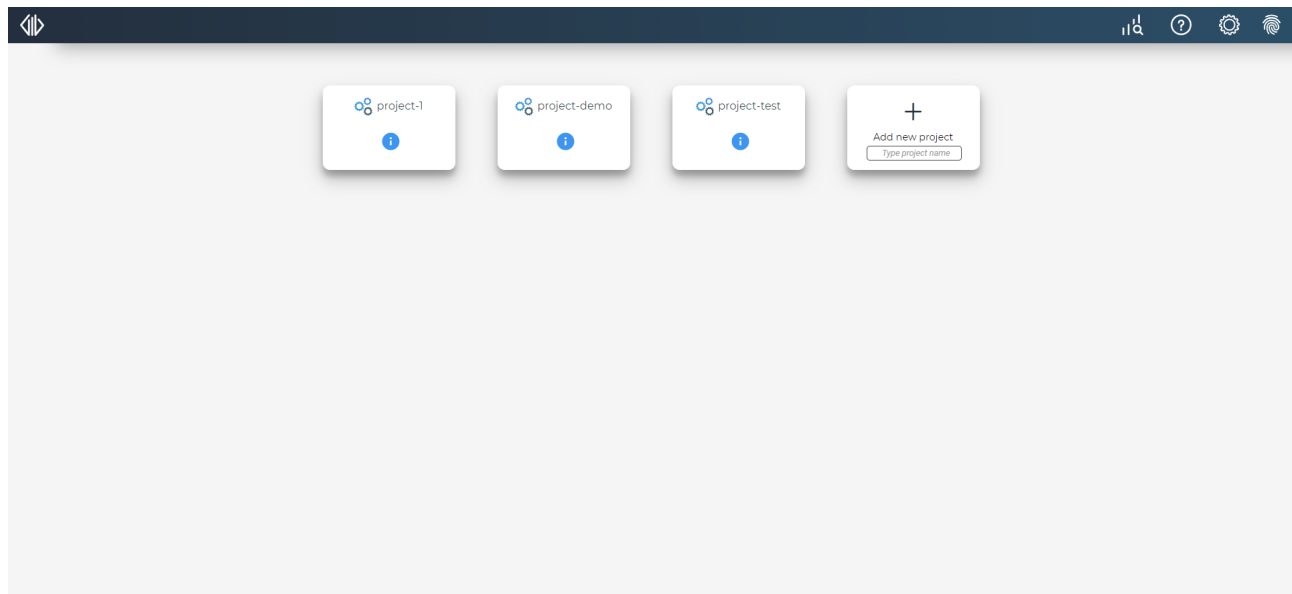


**i** Use the **"username"** and **"password"** fields to enter your credentials.  
Press the **"sign in"** button to proceed to the [Workspace](#)<sup>[27]</sup> view.

## 3.2 Workspace

---

The Workspace page displays the [Top menu toolbar](#)<sup>[95]</sup> and the [Projects](#)<sup>[28]</sup> area .



Workspace

**i Related topics:**


- [Top Menu Toolbar](#)<sup>[95]</sup>
- [Projects](#)<sup>[28]</sup>

### 3.2.1 Home

In the Workspace view, you can either edit an existing project or add a new project.

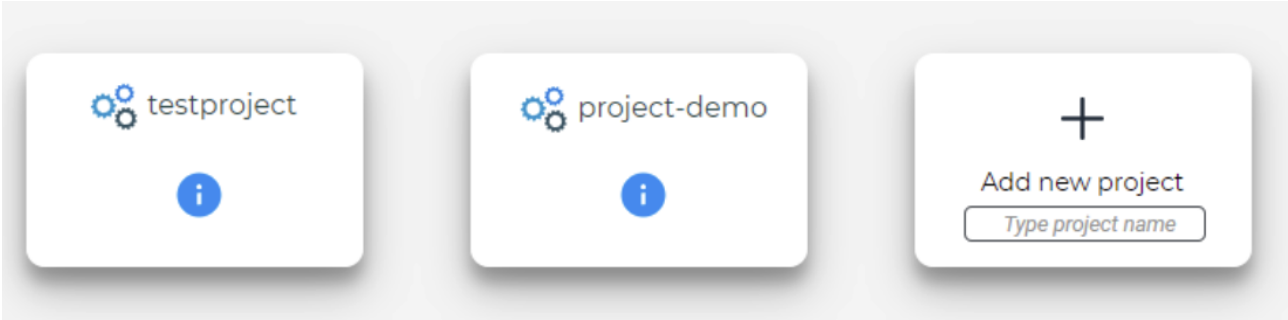
Each project is depicted as a project card.

Use the [Add new project](#)<sup>[30]</sup> card to add a new project.

To view information on an existing project, click the info icon . This action will open the [Project Info Dialog](#)<sup>[29]</sup>.

To open a specific project, click on the corresponding project card. By default, this will take you to the [Project](#)<sup>[30]</sup> tab located on the [Left-Side Menu Toolbar](#)<sup>[94]</sup>.

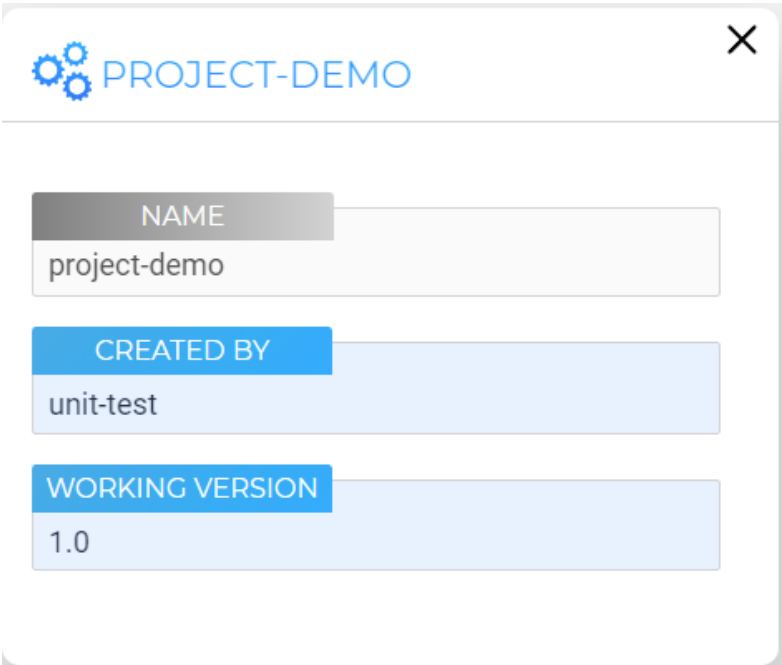
The current Project name is displayed at the top of the Top Menu Toolbar, next to the Home button.



Projects Area

3.2.1.1 Project Info Dialog


The Project Info Dialog shows the name, author and version of a Project.

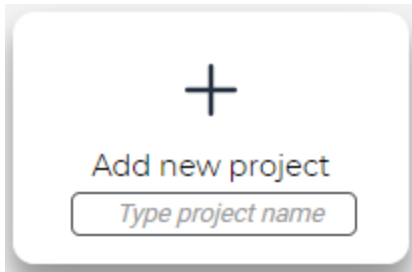


Project Info Dialog

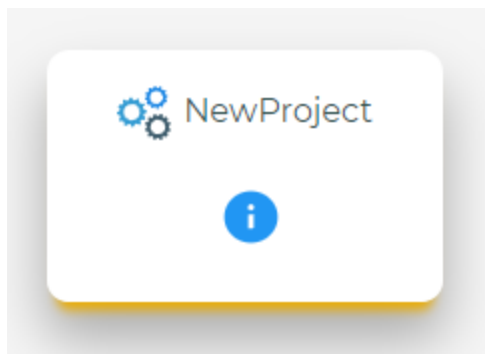
<b>Name</b>	The name of the selected project
<b>Created By</b>	The author of the selected project (identified by the DTS username)
<b>Working Version</b>	The version of the selected project (the version automatically increments as changes are published)


### 3.2.1.2 Add New Project

To add a new project, type the project name into the text field and press the **add button** . This will create a new project card in the projects view.



Newly added projects are underlined in the project list:

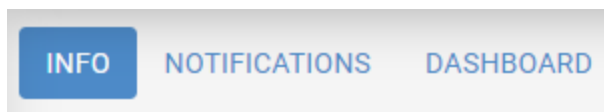


-  Trying to add a new project with the same name as an existing project triggers an error message pop-up;
- Leaving the name field empty when creating a new project does not add a new project card;
- The name field has a 64 character limit; going over the limit triggers an error message pop-up;

### 3.2.2 Project

The Project view offers general information about a Project.

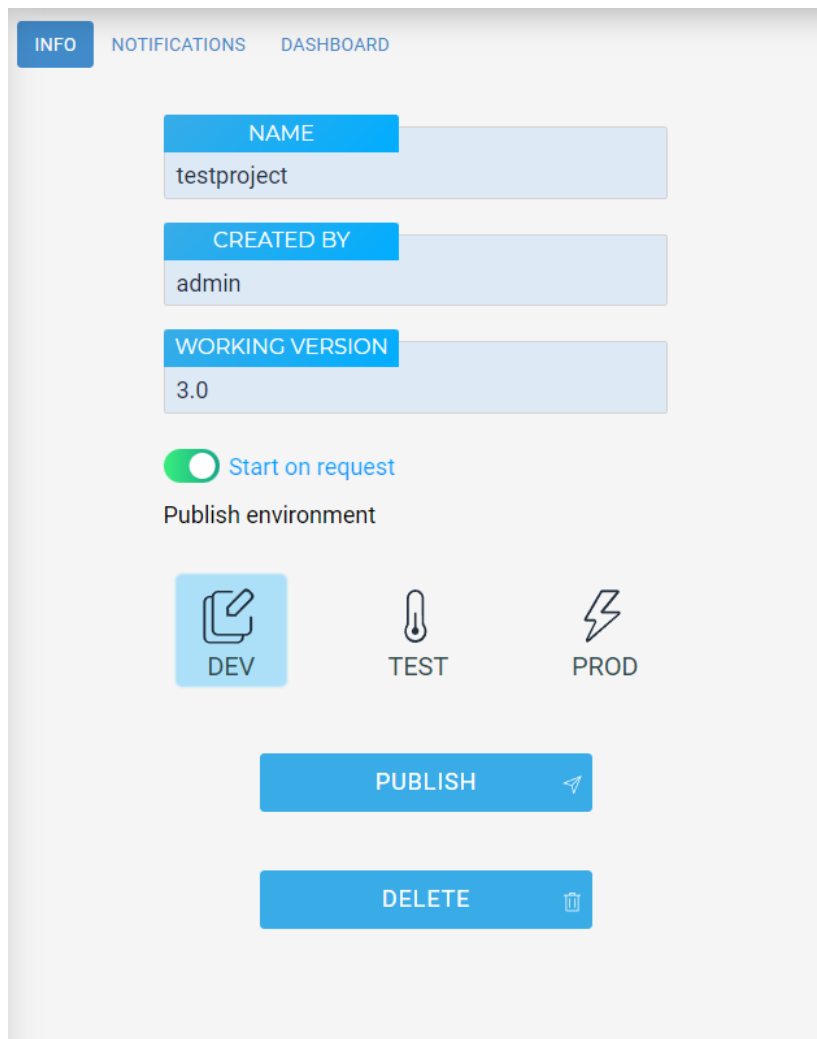
The view is split into 3 main parts:



- [Project Information](#) <sup>31</sup>
- [Project Notifications](#) <sup>32</sup>
- [Project Dashboard](#) <sup>34</sup>

### 3.2.2.1 Project Information









The Project Information Tab contains general information about a Project and allows you to Publish or Delete it.



The screenshot displays the 'Project Information' tab within a web application. At the top, there are three tabs: 'INFO' (selected), 'NOTIFICATIONS', and 'DASHBOARD'. Below the tabs, the form contains several input fields and controls:

- NAME:** A text input field containing 'testproject'.
- CREATED BY:** A text input field containing 'admin'.
- WORKING VERSION:** A text input field containing '3.0'.
- Start on request:** A toggle switch that is currently turned on (green).
- Publish environment:** A label below the toggle switch.
- Environment Selection:** Three buttons with icons: 'DEV' (notepad icon), 'TEST' (thermometer icon), and 'PROD' (lightning bolt icon). The 'DEV' button is highlighted with a blue background.
- PUBLISH:** A large blue button with a paper plane icon.
- DELETE:** A large blue button with a trash can icon.

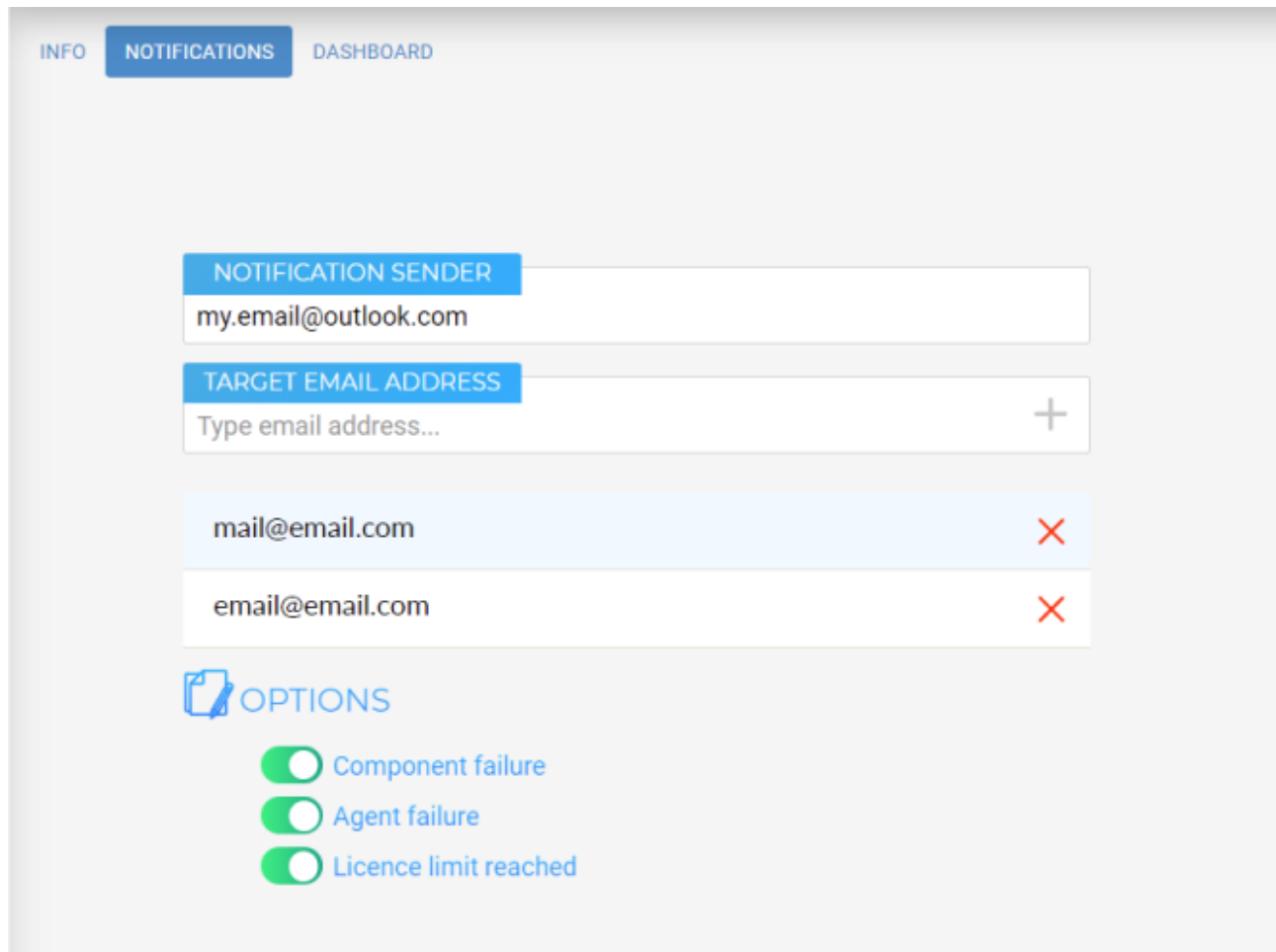
*Project Information Tab*

<b>Name</b>	The name of the selected project
<b>Created By</b>	The author of the selected project (identified by the DTS username)
<b>Working Version</b>	The version of the selected project (the version automatically increments as changes are published)
<b>Start on Request</b>	Automatically start the project when a consumer request for it is registered. If disabled, projects can only be started from the DTS GUI and CLI
<b>Publish Environment</b>	The preferred environment for which you want to publish the project
  <b>DEV</b>	Development Environment
  <b>TEST</b>	Test Environment
  <b>PROD</b>	Production Environment
 <b>Publish</b>	Publish the project
 <b>Delete</b>	Delete the project

### 3.2.2.2 Project Notifications

The **Project Notifications Tab** provides an overview of where notifications are sent and the available features.

A notification is an email message that DTS displays outside the UI to provide the user with reminders or other timely information.




INFO NOTIFICATIONS DASHBOARD

NOTIFICATION SENDER  
my.email@outlook.com

TARGET EMAIL ADDRESS  
Type email address... +

mail@email.com ✕

email@email.com ✕

 OPTIONS


- ☒ Component failure
- ☒ Agent failure
- ☐ Licence limit reached

*Project Notifications Tab*

To choose a **Notification Sender**, select one from the Notification Sender drop-down.

 If you want to add a new Notification Sender, please use the [Notification Senders](#)  dialog.

To add a new recipient for your notifications, click on the **Target Email Address** input field and type a valid email

address. Next, add the recipient to the list by clicking the  **Add button** or by pressing the Enter button on your

keyboard.

 Please note that if the email address is invalid, the Add button will remain disabled.

The available Target Emails are displayed as a list below the Target Email Address input field.

To remove a Target email Address, click on the **✗ Delete button**.

To view the options available for a specific Target Email Address, simply select one from the list.

Use the toggles provided to turn alerts on or off.

<b>Notification Sender</b>	The email address from which the notifications are to be sent
<b>Target Email Address</b>	The email address of the notifications recipient
<b>Options</b>	Toggle preferences for specific alerts
↳ Component Failure	Toggle to receive alerts in case of a component failure
↳ Agent Failure	Toggle to receive alerts in case of an agent failure
↳ License Limit Reached	Toggle to receive alerts when the license limit has been reached

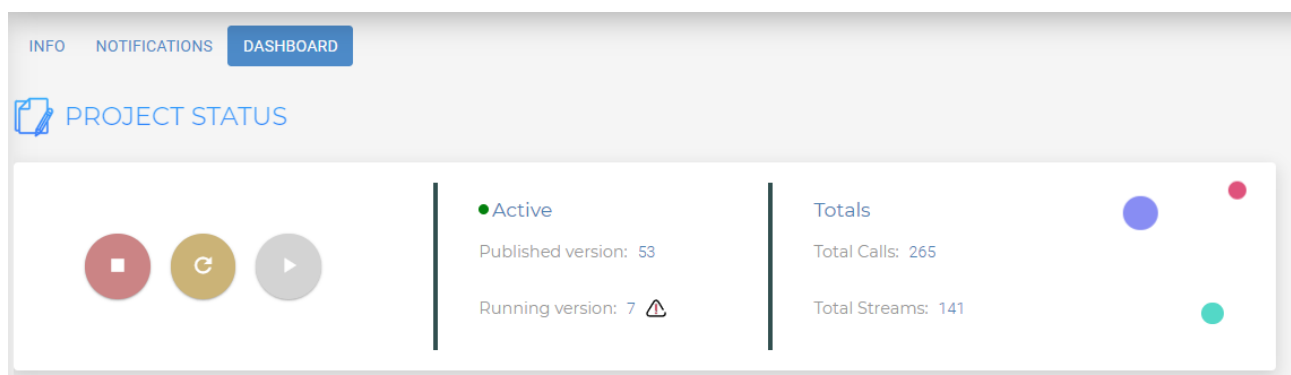
### 3.2.2.3 Project Dashboard

The Project Dashboard Tab provides detailed information about the Project and its available assets.

This view has separate sections for Project Status and Connectors Status, as well as detailed tables for the Producers, Aggregators and Consumers available for the selected project.

## Project Status Section

This section provides general information about the status of the selected project.



*Project Status Section*

**Project Status Buttons**

↳ Stop

The **Stop**, **Reload** and **Start** buttons can be used to Stop, Reload or Start a project.

↳ Reload

Unloads the Project from the Controller, stops all of its producers and aggregators and notifies all of its consumers that the project is has closed.

Performs the Stop action followed by the Start action, but notifies consumers that they should reconnect.

**i** This action should be used to update the running version of a republished project, as it has the least impact on connected consumers.

↳ Start

Loads the latest published version of the project into the Controller and starts all of its baseline producers and aggregators.


**Project Status**

This can either be Active or Inactive

**Published version**

The published version of the selected project (the version automatically increments as changes are published)

**Running version**

The version of the selected project that is currently running. If the running version and the published version of the project are different, the  icon will be present.

**Totals**

↳ Total Calls

The total number of Routine Calls served by the project

↳ Total Streams

The total number of Streams served by the project

**i** If a project is **Active**, the **Start** button is disabled, the project can only be stopped or reloaded.  
If a project is **Inactive**, the **Stop** and **Reload** buttons are disabled, the project can only be started.

## Connectors Status Section

This section provides a real-time visual interpretation of the number of Active Calls, Active Streams, Total Calls and Total Streams registered on the active producers for each connector at any given time through a dedicated chart.

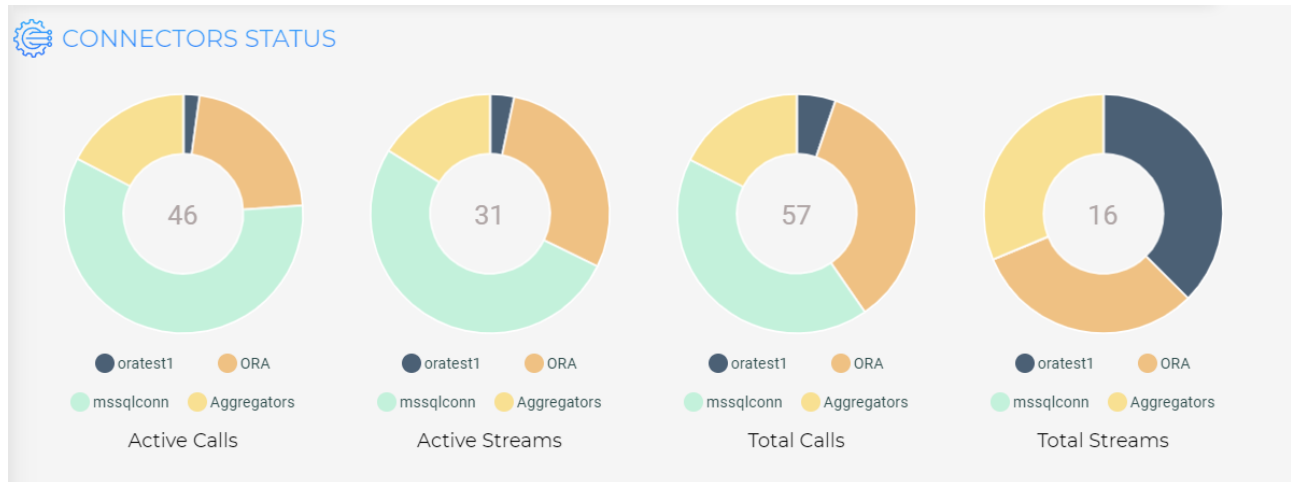
Each color-coded section of a chart represents a connector.

**i** If you hover over a section of a chart a tool-tip will appear that contains the name of the connector followed by the number of Active Calls, Active Streams, Total Calls and Total Streams registered on all of its producers.

In the center of each chart you can find the total number of Active Calls, Active Streams, Total Calls or Total Streams registered on the active producers for all the connectors.

**i** If no producers are yet available for the connectors or there aren't any active calls or streams , the total number will not be available and the number will be replaced with N/A.

The chart legend is interactive. Clicking on a Connector name will take you to its corresponding table in the Producers section.



Connectors Status Section

**i** Topic Operations (Poll/Push) register as Calls, while Subscribed topics appear as Streams.

## Producers Section

This section provides a real-time visual interpretation of each of the active producers from a project connector.

Each Connector has its own section (table) that contains information about its corresponding producers.

Connector	Component Id	Start Time	Session Name	Active Calls	Active Streams	Total Calls	Total Streams	Status	Stop
ORA	DTS:Oracle:4	2021-10-19T11:04:42.854Z	dts-producer-ora-1	5	1	5	0	✓ ACTIVE	⛔
	DTS:Oracle:6	2021-10-19T11:04:42.854Z	dts-producer-ora-3	3	4	7	0	✓ ACTIVE	⛔
	DTS:Oracle:3	2021-10-18T11:06:42.854Z	dts-producer-ora-2	2	4	8	5	⚙ STARTING	⛔
	+ Add new producer			10	9	20	5		

10

Producers Table

The **Start Time**, **Active Calls**, **Active Streams**, **Total Calls** and **Total Streams** table headings are interactive.

Click on them to sort the producers by your chosen criteria in ascending ▲ or descending ▼ order.

This section provides a real-time visual interpretation of the number of **Active Calls**, **Active Streams**, **Total Calls** and **Total Streams** registered on the active producers at any given time through a dedicated chart.

To activate the chart, click on any of the following table headings: **Active Calls**, **Active Streams**, **Total Calls**, **Total Streams**.

Each color-coded section of the chart represents an active producer, identified by its component ID.

**i** If you hover over a section of a chart a tool-tip will appear that contains the component ID of the producer followed by the number of Active Calls, Active Streams, Total Calls and Total Streams registered on it.

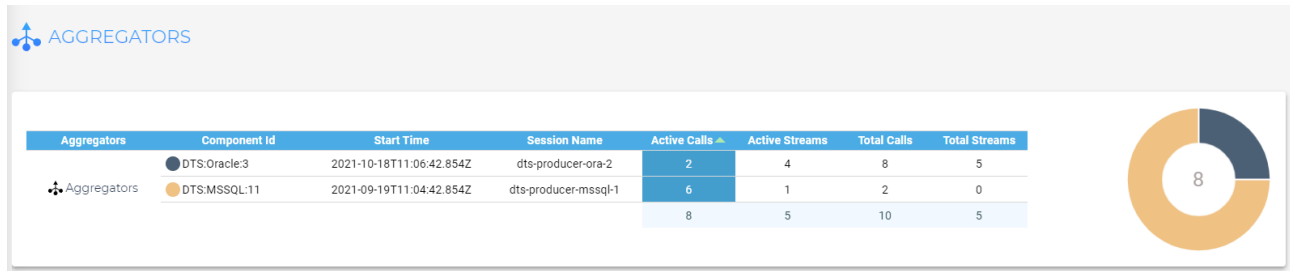
In the center of each chart you can find the total number of Active Calls, Active Streams, Total Calls or Total Streams registered on the active producers for a specific connector. The totals are also available at the bottom of the table under their corresponding sections.

**i** If no producers are yet available or there aren't any active calls or streams, the total number will not be available and the number will be replaced with N/A.

<b>Connector</b>	Displays the name of the connector
<b>Component ID</b>	Displays the Component ID of a producer. (color-coded to chart)
<b>Start Time</b>	Displays the exact time and date of when the producer was started
<b>Session Name</b>	Displays the session name of a producer
<b>Active Calls</b>	Displays the number of Active Calls for a specific producer
<b>Active Streams</b>	Displays the number of Active Streams for a specific producer
<b>Total Calls</b>	Displays the number of Total Calls for a specific producer
<b>Total Streams</b>	Displays the number of Total Streams for a specific producer
<b>Status</b>	Displays the status of a producer. This can either be Active, or Starting (if the producer has just been added).
<b>Stop</b>	Click on the Stop button to stop a specific producers.  <b>i</b> A manually stopped producer will not be replaced by the Controller.
<b>Add New Producer</b>	Click on the Add button to start a new producer for the Connector.  <b>i</b> A manually started producer will not be subject to automatic scale-downs by the Controller, but will be automatically replaced if it becomes unresponsive.

## Aggregators section

This section provides a real-time visual interpretation of each of the active aggregators at any given time through a dedicated chart.



Aggregators Table

The **Start Time**, **Active Calls**, **Active Streams**, **Total Calls** and **Total Streams** table headings are interactive.

Click on them to sort the aggregators by your chosen criteria in ascending ▲ or descending ▼ order.

This section provides a real-time visual interpretation of the number of **Active Calls**, **Active Streams**, **Total Calls** and **Total Streams** registered on the active aggregators at any given time through a dedicated chart.

To activate the chart, click on any of the following table headings: **Active Calls**, **Active Streams**, **Total Calls**, **Total Streams**

Each color-coded section of the chart represents an active aggregator, identified by its component ID.

**i** If you hover over a section of a chart a tool-tip will appear that contains the component ID of the aggregator followed by the number of Active Calls, Active Streams, Total Calls and Total Streams registered on it.

In the center of each chart you can find the total number of Active Calls, Active Streams, Total Calls or Total Streams registered on the active aggregators. The totals are also available at the bottom of the table under their corresponding sections.

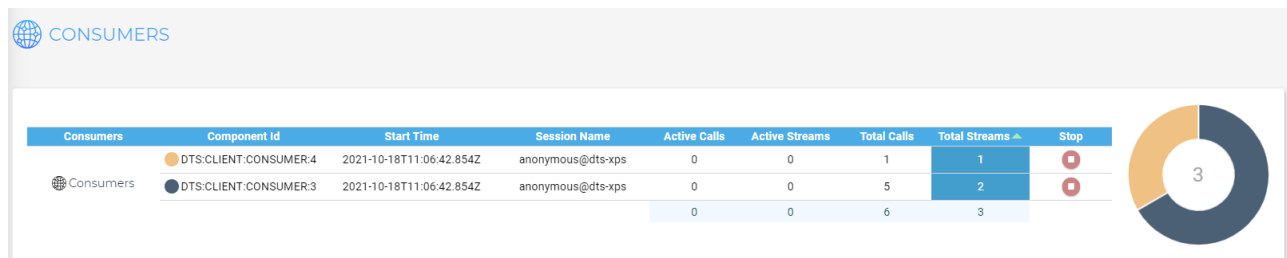
**i** If no aggregators are yet available or there aren't any active calls or streams, the total number will not be available and the number will be replaced with N/A.

<b>Aggregators</b>	Displays the name of the table
<b>Component ID</b>	Displays the Component ID of an aggregator. (color-coded to chart)
<b>Start Time</b>	Displays the exact time and date of when the aggregator was started
<b>Session Name</b>	Displays the session name of an aggregator

<b>Active Calls</b>	Displays the number of Active Calls for a specific aggregator
<b>Active Streams</b>	Displays the number of Active Streams for a specific aggregator
<b>Total Calls</b>	Displays the number of Total Calls for a specific aggregator
<b>Total Streams</b>	Displays the number of Total Streams for a specific aggregator

## Consumers section

This section provides a real-time visual interpretation of each of the active consumers at any given time through a dedicated chart.



Consumers Table

The **Start Time**, **Active Calls**, **Active Streams**, **Total Calls** and **Total Streams** table headings are interactive.

Click on them to sort the consumers by your chosen criteria in ascending ▲ or descending ▼ order.

This section provides a real-time visual interpretation of the number of **Active Calls**, **Active Streams**, **Total Calls** and **Total Streams** registered on the active consumers at any given time through a dedicated chart.

To activate the chart, click on any of the following table headings: **Active Calls**, **Active Streams**, **Total Calls**, **Total Streams**

Each color-coded section of the chart represents an active consumer, identified by its component ID.

**i** If you hover over a section of a chart a tool-tip will appear that contains the component ID of the consumer followed by the number of Active Calls, Active Streams, Total Calls and Total Streams registered on it.

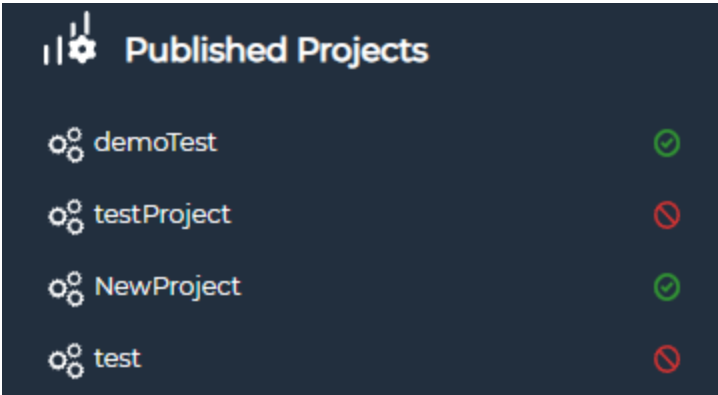
In the center of each chart you can find the total number of Active Calls, Active Streams, Total Calls or Total Streams registered on the active consumers. The totals are also available at the bottom of the table under their corresponding sections.



**i** If no consumers are yet available or there aren't any active calls or streams , the total number will not be available and the number will be replaced with N/A.

<b>Consumers</b>	Displays the name of the table
<b>Component ID</b>	Displays the Component ID of a consumer. (color-coded to chart)
<b>Start Time</b>	Displays the exact time and date of when the consumer was started
<b>Session Name</b>	Displays the session name of a consumer
<b>Active Calls</b>	Displays the number of Active Calls for a specific consumer
<b>Active Streams</b>	Displays the number of Active Streams for a specific consumer
<b>Total Calls</b>	Displays the number of Total Calls for a specific consumer
<b>Total Streams</b>	Displays the number of Total Streams for a specific consumer

### 3.2.2.4 Published Projects

The Published Projects Tab offers information about the status of your Published Projects.



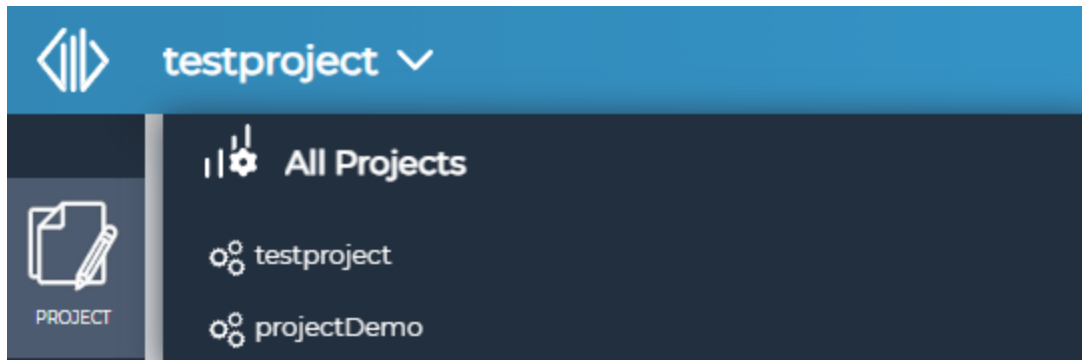
<b>Name</b>	The name of the selected project
 <b>Active</b>	Shows that the status of the project is active
 <b>Inactive</b>	Shows that the status of the project is inactive

**i** Selecting a project from the list will open the [Project Dashboard](#)<sup>34</sup> tab.

The status information is available for Published Projects only.

### 3.2.2.5 All Projects

The All Projects Tab lists all the projects available.



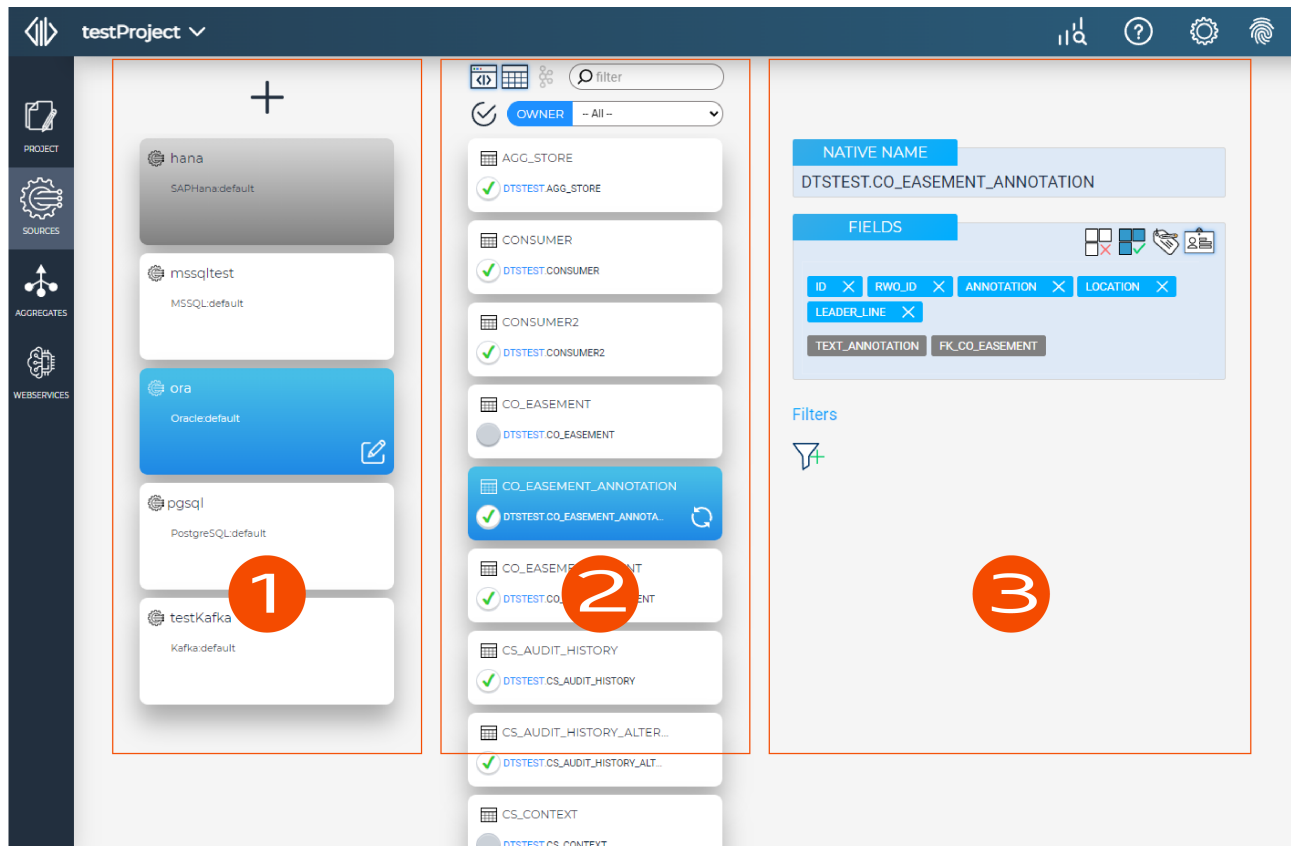
**i** Selecting a project from the list will open the Project from where you left off or the [Project Information](#) tab for newly created projects.

The list is available for all projects, including Published Projects.

### 3.2.3 Sources

The Sources view shows the Datasource Connections included in a Project, their available assets and those assets' structure. as well as providing functional interactions with them.

The view is split into 3 main parts:

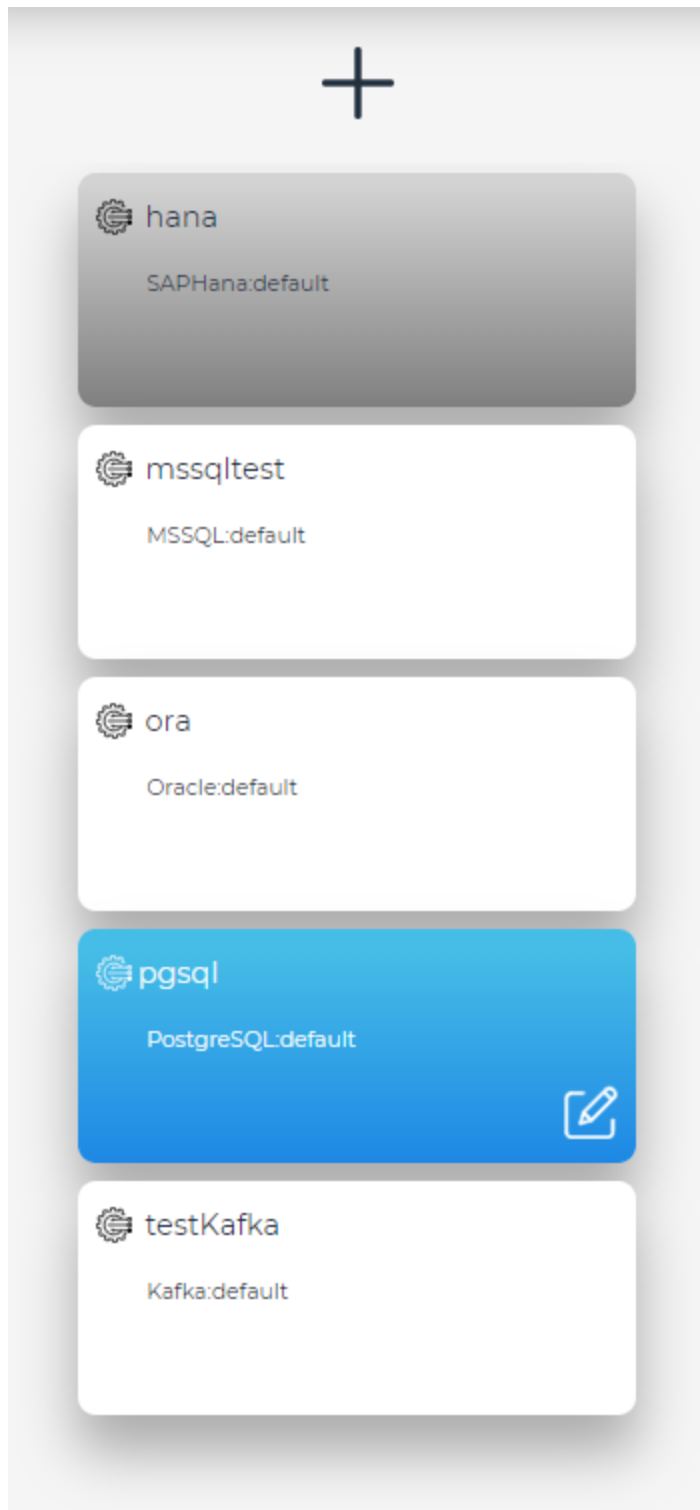


Sources View

1. [Connectors Drawer](#) <sup>42</sup>
2. [Assets Drawer](#) <sup>49</sup>
3. [Asset Details Drawer](#) <sup>51</sup>



### 3.2.3.1 Connectors Drawer


The Connectors Drawer lists the Datasource Connectors currently defined in the Project as a list of Connector Cards. From here, you can add more Datasource Connectors or interact with the existing ones.



*Connectors Drawer*



To add a new Connector, click on the **Add Button** . This will open the [Create New Connector](#) <sup>44</sup> dialog.

To edit or view details for a specific Connector Instance, click on the **Edit Button**  on the respective card. This will open the [Connector Details](#) <sup>47</sup> dialog.

The selected connector has a blue background. Disabled connectors have a gray background. To enable or disable a connector go to the [Connector Details](#) <sup>47</sup> dialog.

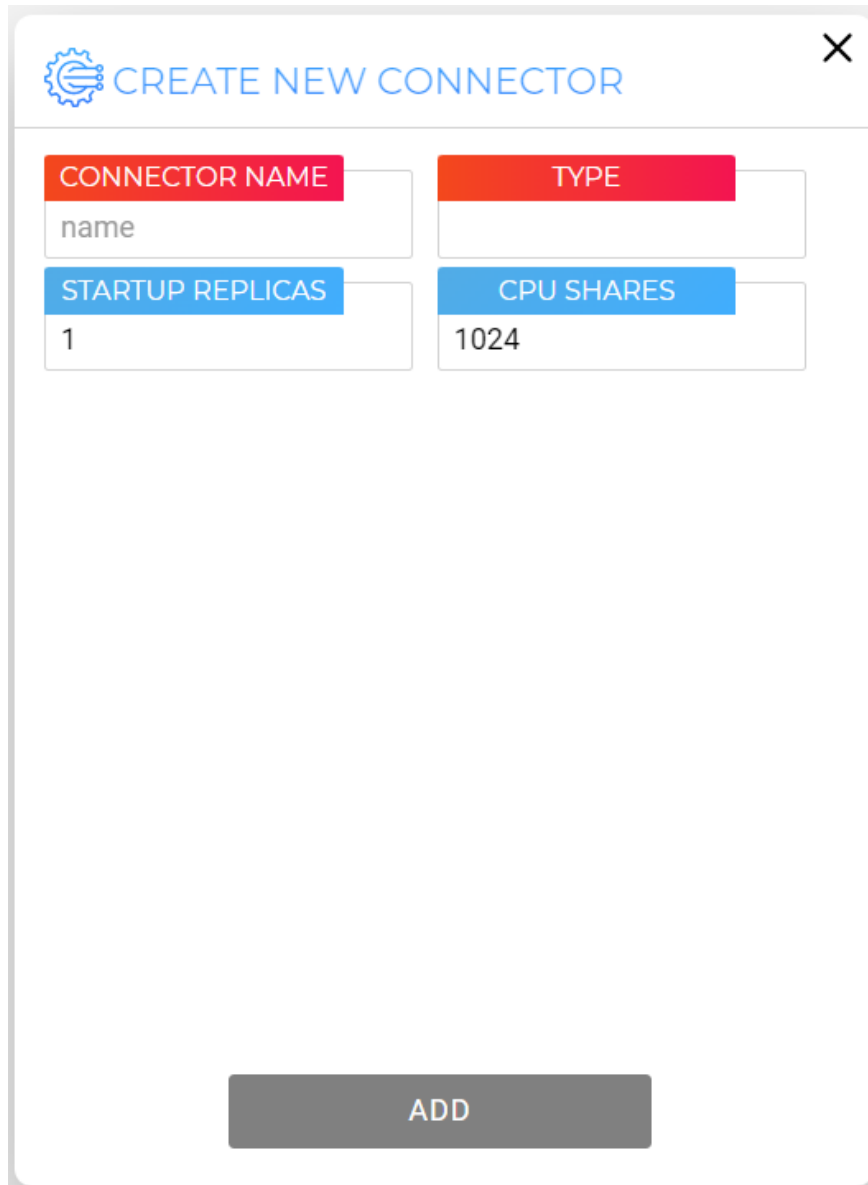
 Selecting a Connector Card will update the [Assets Drawer](#) <sup>49</sup> with the list of available assets for that particular Datasource.

All connectors are enabled by default.

 Activating a Connector Card triggers a background update operation that might generate messages, errors or warnings. To read more about this subject go to [Errors & Warnings](#) <sup>114</sup>.

### 3.2.3.1.1 Create New Connector

The Create New Connector dialog allows you to create a new Datasource Connection for your Project.



The dialog box is titled "CREATE NEW CONNECTOR" with a gear icon on the left and a close button (X) on the right. It contains four input fields arranged in a 2x2 grid. The top-left field is labeled "CONNECTOR NAME" in red and contains the text "name". The top-right field is labeled "TYPE" in red and is empty. The bottom-left field is labeled "STARTUP REPLICAS" in blue and contains the value "1". The bottom-right field is labeled "CPU SHARES" in blue and contains the value "1024". At the bottom center of the dialog is a large grey button labeled "ADD".

*Add New Connector Dialog*

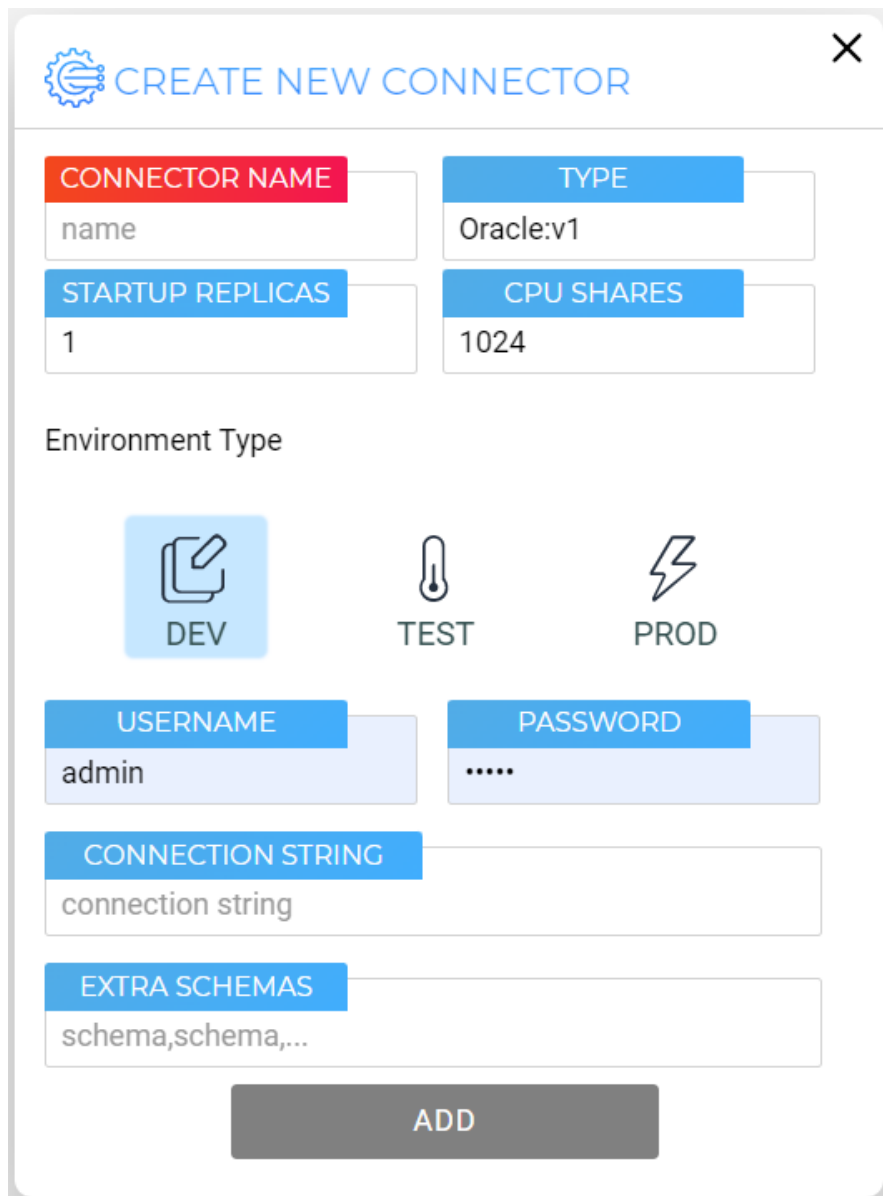
Initially, you will be presented with only two fields:

<b>Connector Name</b>	The name given to the Datasource Connector (must be unique within the Project)
<b>Connector Type</b>	The type of Datasource Connector
<b>Startup Replicas</b>	The initial number of instances (producers) that should be started for the Connector when the Project is booted.
<b>CPU Shares</b>	The priority for CPU time for instances (producers) of this connector. This is a Docker-specific parameter and its default value is 1024.

**i** Connector Types are defined using the [Connector Types Dialog](#) <sup>104</sup>

**i** For more information regarding Docker CPU Shares, please see the [relevant Docker documentation](#).

After choosing a Connector Type, the dialog will show more options:



The screenshot shows a 'CREATE NEW CONNECTOR' dialog box with a close button (X) in the top right corner. The dialog contains the following fields and options:

- CONNECTOR NAME:** A text field with the value 'name'.
- TYPE:** A dropdown menu showing 'Oracle:v1'.
- STARTUP REPLICAS:** A text field with the value '1'.
- CPU SHARES:** A text field with the value '1024'.
- Environment Type:** Three radio button options: 'DEV' (selected, represented by a document icon), 'TEST' (represented by a thermometer icon), and 'PROD' (represented by a lightning bolt icon).
- USERNAME:** A text field with the value 'admin'.
- PASSWORD:** A password field with masked characters '.....'.
- CONNECTION STRING:** A text field with the value 'connection string'.
- EXTRA SCHEMAS:** A text field with the value 'schema,schema,...'.
- ADD:** A large grey button at the bottom center.

*Add new connector Dialog - Oracle*


<b>Environment Type</b>	The type of environment the connector will be published in
<b>Connection Parameters</b>	The parameters required to connect to the Datasource implied by the Connector Type

To add a new Connector, click on the **Add button**.

 The fields that appear in the Connection Parameters area will depend on the Category of the Connector Type and can be explored in the respective [Connector](#)<sup>118</sup>'s Connection Parameters.

### 3.2.3.1.2 Connector Details

The Connector Details dialog shows information about a given Datasource Connector and allows editing certain properties, as well as removing it from the Project.

 ORA ✕

☒ Enabled

TYPE

Oracle:default


STARTUP REPLICAS


1


CPU SHARES

1024

Environment Type

 DEV

 TEST

 PROD

USERNAME

dtstest

PASSWORD

\*\*\*\*\*


CONNECTION STRING

jdbc:oracle:thin:@172.16.10.212:1521/dtstestpdb

EXTRA SCHEMAS

schema1,schema2,...

DELETE



Connector Details Dialog

**Enabled** Choose if the connector is enabled or disabled (default : enabled)

 Disabled connectors will have no instances (producers) started when the Project boots up.




**Type** The type of Datasource Connector (unmodifiable)

 Connector Types are defined using the [Connector Types Dialog](#) <sup>104</sup>

<b>Startup Replicas</b>	The initial number of instances (producers) that should be started for the Connector when the Project boots up.
<b>CPU Shares</b>	The priority for CPU time for instances (producers) of this connector. This is a Docker-specific parameter and its default value is 1024.
<b>i</b> For more information regarding Docker CPU Shares, please see the <a href="#">relevant Docker documentation</a> .	
<b>Environment type</b>	The type of environment the connector will be published in
<b>Connection Parameters</b>	The parameters required to connect to the Datasource implied by the Connector Type
<b>i</b> The fields that appear in the Connection Parameters area will depend on the Category of the Connector Type and can be explored in the respective <a href="#">Connector</a> 's Connection Parameters	
<b>Delete</b>	Remove the Datasource Connector from the Project

### 3.2.3.2 Assets Drawer

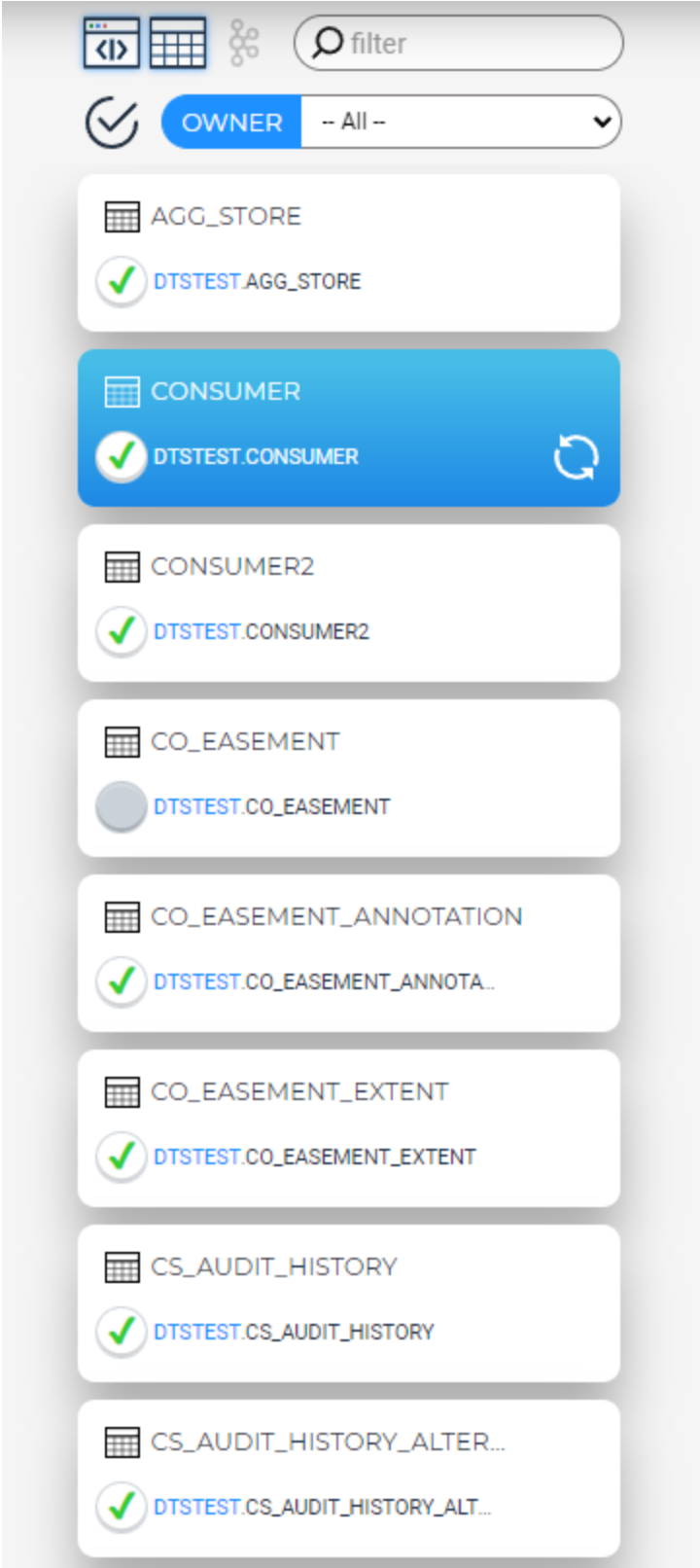
The Assets Drawer shows the available assets for the Datasource selected in the [Connectors Drawer](#) as a list of cards and allows you to search and filter them as well as include or exclude them from the Project.

Items in the Asset Drawer can be either **Collections**  (tables, views, etc.), **Routines**  (methods, procedures, functions, etc.) or **Topics** .

Depending on the Connector chosen, the asset type availability may differ:

- All available asset type icons are displayed in black
- Unavailable asset type icons are displayed in light gray
- Selected asset type icons are displayed with a blue border

**i** Multiple asset type icons can be selected at a time to filter items accordingly. However, at least one of the available asset types must remain selected.



Assets Drawer

 **Show only selected toggle**

Shows only the items currently included in the Project

 **Collections toggle**

Shows the available collections

 **Routines Toggle**

Shows the available routines

 **Topics Toggle**

Shows the available topics

**Search/ Filter bar**

Search for a specific item by name and owner

**Owners Drop-down**

Filters available items by their owner:


- **-- All --** : Displays all items (for all owners) - this is the default setting
- **-- None --** : Displays only items that have no declared owner
- **[Owner name entry]** : Displays only items with that specific owner

Additionally, the owner is displayed on each item card as a blue text button. Clicking on this button will filter all items by that specific owner.

Assets can be included in or excluded from the Project by using the **Enable Button**  on each card.


Assets can be reloaded from the source metadata using the **Reload Button**  on each card.

 Selecting a resource card will open the [Asset Details Drawer](#) <sup>[51]</sup>.

 Activating a resource card triggers an operation that might generate messages, errors or warnings. To read more about this subject go to [Errors & Warnings](#) <sup>[114]</sup>.

### 3.2.3.3 Asset Details Drawer

The Asset Details Drawer shows details about the Asset currently selected in the [Assets Drawer](#) <sup>[49]</sup>. Depending on the type of Asset that is selected, the Asset Details Drawer can present in three ways:

1. [Collection Details Drawer](#)  52
2. [Routine Dettails Drawer](#)  55
3. [Topic Details Drawer](#)  57

### 3.2.3.3.1 Collection Details Drawer

The drawer allows you to see the **Fields** of the selected Collection.



Collections Details Drawer

- Native Name** Displays the native name of the selected collection (i.e. its full given name in the source data model)
- Fields** Displays the fields (columns) available in the collection

## Fields

-  A **blue** field indicates that the field is included in the Project.  
A **gray** field indicates that the field is available but not included in the Project.

To edit a field, or view the available information about the field from the Project simply click on the field to expand.

To close the field information window click on the close  icon.

The Fields container also presents 4 buttons:



**Deselect All**

Removes all the fields available



**Select All**

Includes all the fields available



**Show Custom Names**


Will show the fields' DTS Names in the Fields container's tiles.




**Show Native Identifiers**

Will show the fields' Native Identifiers in the Fields container's tiles.

-  Show Custom names and Show Native Identifiers can be active at the same time, and at least one of them needs to be active.

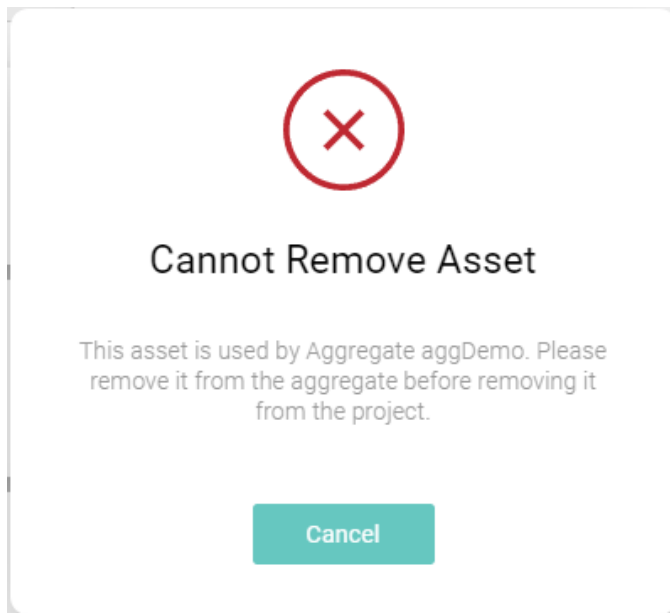
-  A **blue** background of an input field indicates that the field is read-only and its contents cannot be modified.

A white background of an input field indicated that the value of the field can be modified.

To remove a field from the Project simply click on the **delete**  icon . The field will become gray.

To add a field to a Project simply click on the gray field to change its state.

If the field is being used by another resource in the project, it cannot be removed, and an Alert Dialog will appear on the screen.



*Alert Dialog*

## Filters

This section allows you to set up, edit or remove fundamental filters on the collection asset.

A fundamental filter will only allow DTS Clients to consume records that match the filter's clauses. Any query made on the collection through DTS will be combined with the fundamental filter before being applied in the data source.



**Add Filter**

Opens the Insert Filter dialog, allowing you to add a complex filter



**Edit Filter**

Opens the Edit Filter dialog, allowing you to edit an existing filter



**Remove Filter**

Removes an existing filter



For more information about filters, please check the [Filters](#) section.

### 3.2.3.3.2 Routine Details Drawer

The drawer allows you to see the **Arguments** and **Results** of the selected Routine.

ARGUMENTS

CONSUMER\_ID

Native ID : CONSUMER\_ID

Custom Name : CONSUMER\_ID

Type : DECIMAL ( NUMBER )

RESULTS

SOURCE\_ASSET

AFFECTED\_CONSUMERS

Native ID : AFFECTED\_CONSUMERS

Custom Name : AFFECTED\_CONSUMERS

Type : DTSTEST.T\_CONSUMER ( DTSTEST.T\_CONSUMER )

Routine Details Drawer



Show Custom Names

Will show the arguments'/results' DTS Names in the Fields container's tiles.

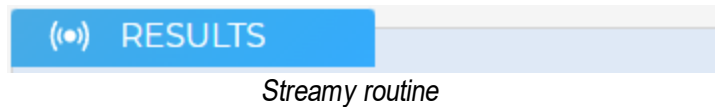


Show Native Identifier

Will show the arguments'/results' Native Identifiers in the Fields container's tiles.

Streamy Routines

If a Routine is Streamy (i.e. it returns a Stream), the <sup>(•)</sup> **Streamy icon** will appear in the Results section.



**i** A **blue** background of an input field indicates that the field is read-only and its contents cannot be modified.

A white background of an input field indicated that the value of the field can be modified.

To edit an argument or a result, or view the available information about them from the Project simply click to expand.

To close the information window click on the  icon.

### 3.2.3.3.3 Topic Details Drawer

The drawer offers details about the selected Topic.

NATIVE NAME

test-topic-2

PARTITIONS

0

1

KEY TYPE

STRING

MESSAGE TYPE

BYTE VECTOR

☒ Read Enabled

☒ Write Enabled

PROPERTIES

key=value  
key=value  
...

*Topic Details Drawer*

<b>Native Name</b>	Displays the native name of the selected topic (i.e. its full given name in the source data model)
<b>Partitions</b>	Displays the partitions available in the topic
<b>Key Type</b>	Select key type (STRING or BYTE VECTOR)
<b>Message Type</b>	Select message type (STRING or BYTE VECTOR)

**i** Key Type and Message Type define how keys and messages will be serialized and deserialized when pushed or polled from the Topic respectively.

**Read Enabled** Enable reading (polling) from the selected Topic.

**Write Enabled** Enable writing (pushing) to the selected Topic.

**Properties** Allows you to add or change connection properties for a specific topic

**i** The properties field accepts only one key=value property per line. Typing errors will trigger a warning.

The label becomes red when the property file is invalid, and blue when it becomes valid.

**i** Read/Write Enabled controls affect all DTS consumers that interact with the Topic and also reflect in the operations available for generated [Webservices](#)<sup>89</sup>.

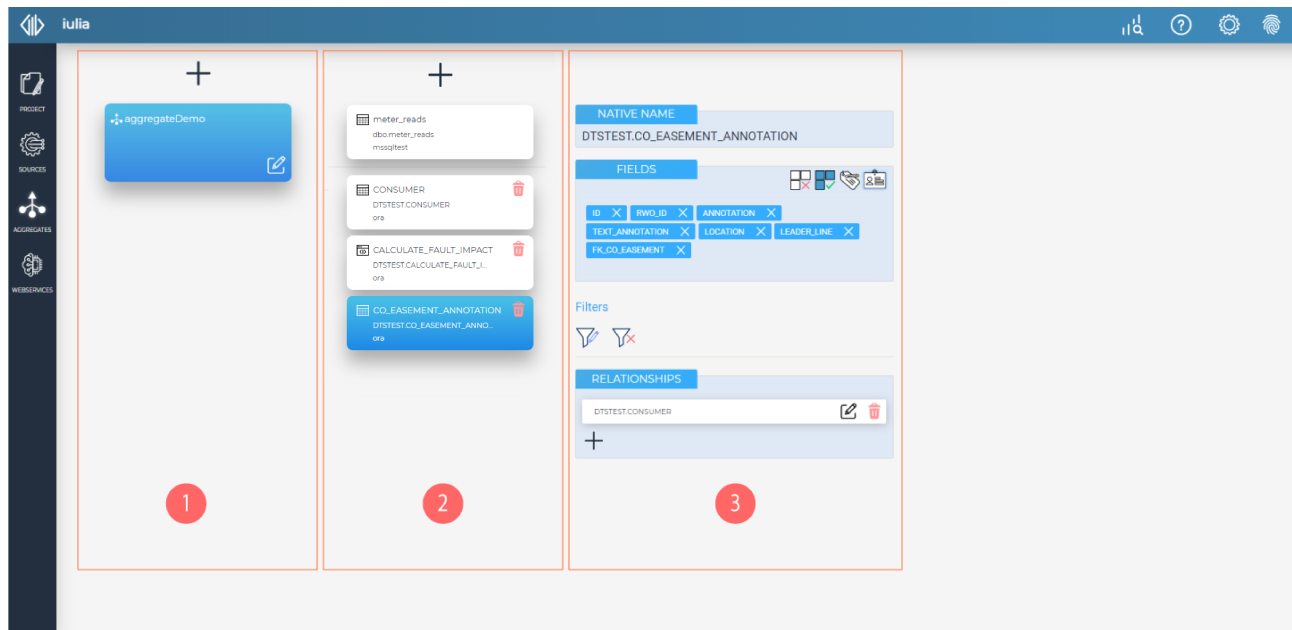
**i** At least one of the two methods (Read or Write) has to be enabled.

If both methods are disabled by the user, the topic will be automatically removed from the project.

### 3.2.4 Aggregates

The Aggregates view shows the Aggregate Connections included in a Project, their available assets and those assets' structure. as well as providing functional interactions with them.

The view is split into 3 main parts:



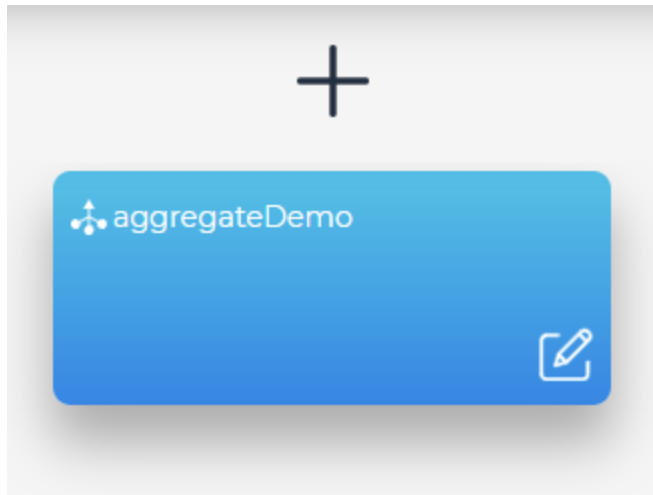
1. [Aggregates Drawer](#)<sup>[60]</sup>
2. [Aggregate Assets](#)<sup>[64]</sup>
3. [Aggregate Details Drawer](#)<sup>[66]</sup>

**i** For technical information regarding the structure, rules and limitations of Aggregates, please see the [Aggregation Technical Guide](#)<sup>[242]</sup>.


### 3.2.4.1 Aggregates Drawer

The **Aggregates Drawer** lists the Aggregates currently defined in the Project as a list of Aggregate Cards.

From here, you can add more Aggregates or interact with the existing ones.





To add a new Aggregate, click on the **Add Button** . This will open the [Create New Aggregate](#) <sup>61</sup> dialog.

To edit or view details for a specific Aggregate Instance, click on the **Edit Button**  on the respective card. This will open the [Aggregate Details](#) <sup>63</sup> dialog.

 **Selecting a Connector Card will update the [Aggregate Assets](#) <sup>64</sup> with the list of assets for that particular Aggregate.**

### 3.2.4.1.1 Create New Aggregate

The Create New Aggregate dialog allows you to create a new Aggregate for your Project.

 CREATE NEW AGGREGATE 

NATIVE NAME

aggTest

CONNECTOR

mssqlconn

SOURCE : SAPCONSUMPTIONDATAS

dbo.SapConsumptionDatas

dbo.meter\_reads

ADD

→

Create new Aggregate dialog

To create a new Aggregate, you must provide a Name and the Main Source Asset.

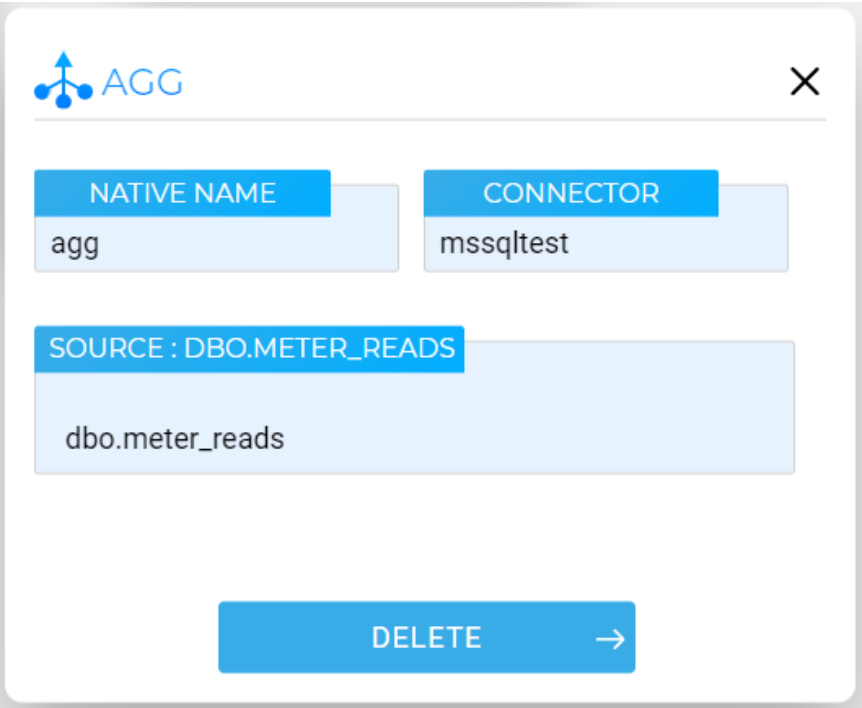
Native Name	The name given to the Aggregate (must be unique within the Project)
Connector	The Connector which contains the Main Source
Source	Allows you to choose a source from the <b>Connector</b> to be the Aggregate's Main Source Asset

To add a new Aggregate, click on the **Add button**.

- i** The sources that appear in the Sources area will depend on the Connector chosen. Only assets that have been included in the Project in the [Sources->Assets Drawer](#)<sup>[49]</sup> will be available.
- i** For more information regarding Aggregate structure and Main Sources, please see the [Aggregation Technical Guide](#)<sup>[242]</sup>

3.2.4.1.2 Aggregate Details

The Aggregate Details dialog shows information about a given Aggregate and allows you to remove it from the Project.



Aggregate details dialog

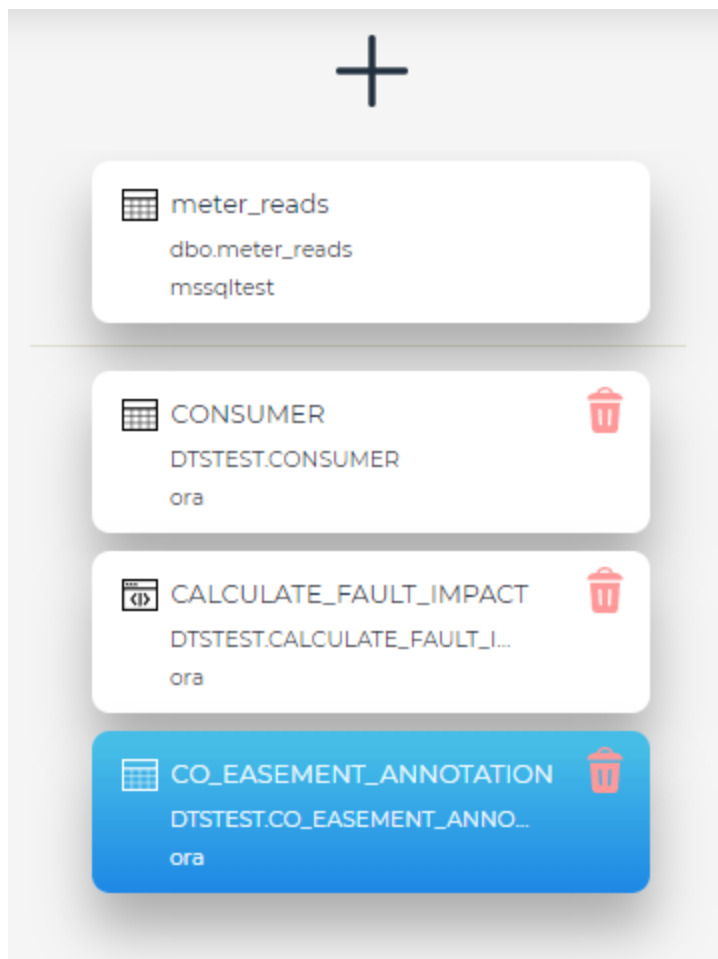
Native Name	The name of the Aggregate (unmodifiable)
Connector	The name of the Connector containing the Main Source(unmodifiable)
Source	The name of the Main Source (unmodifiable)
Delete	Remove the Aggregate from the Project

### 3.2.4.2 Aggregate Assets

The Aggregate Assets Drawer shows the source assets in use by the Aggregate selected in the [Aggregates Drawer](#)<sup>[60]</sup> as a list of cards.


Items in the Asset Drawer can be either **Collections**  (tables, views, etc.) or **Routines**  (methods, procedures, functions, etc.).

The first item (visually separated) will always be the Main Source of the Aggregate.



*Aggregate Assets Drawer*

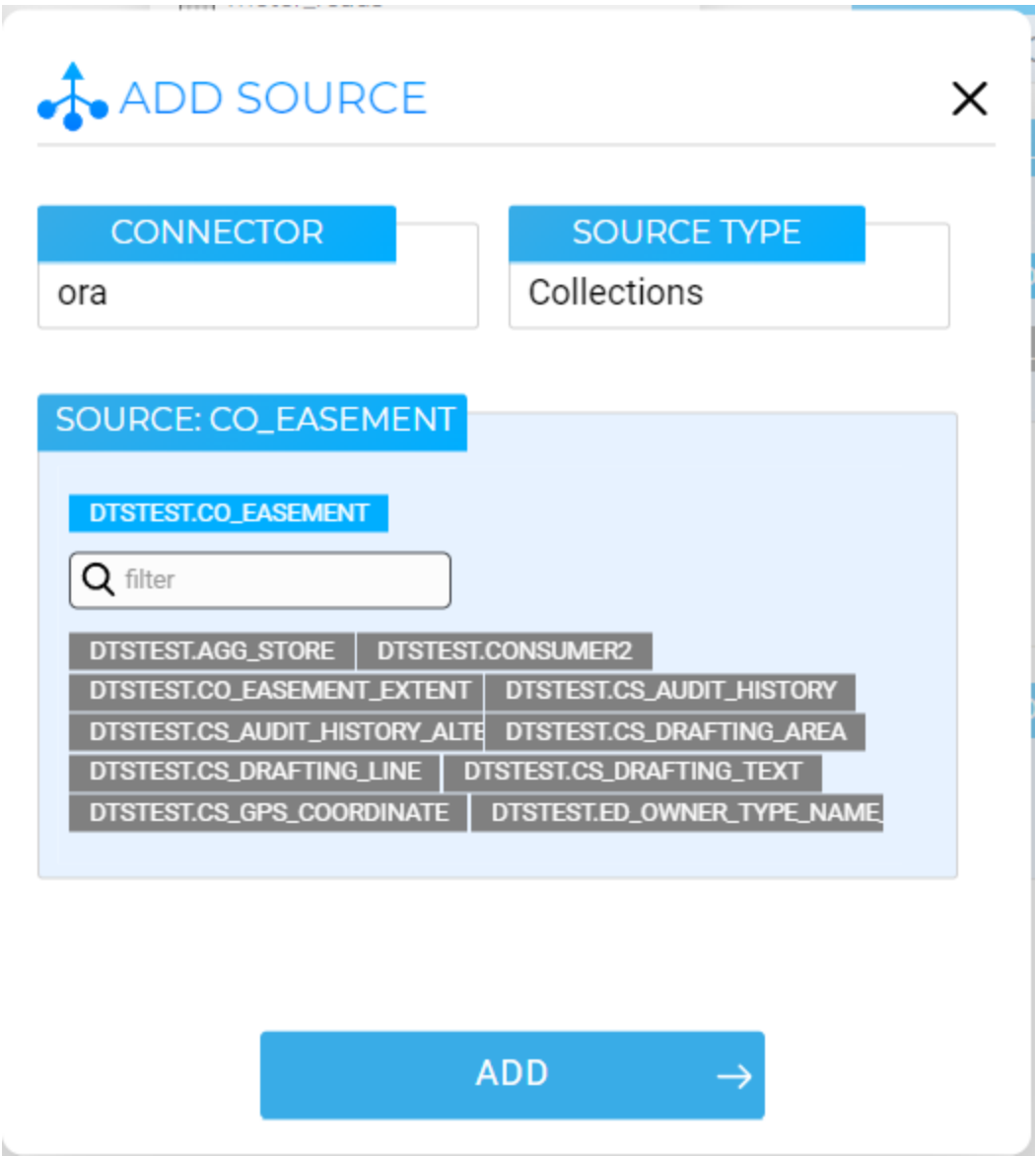
To add a new **Aggregate Source**, click on the **Add Button** . This will open the [Add new Aggregate Source](#)<sup>[65]</sup> dialog.

Any source asset, other than the Main Source, can be removed from the Aggregate by clicking the **Remove Button** . This will also remove all relationships that target the source.

 For more information regarding Aggregate structure, please see the [Aggregation Technical Guide](#) 

3.2.4.2.1 Add New Aggregate Source

The **Add New Aggregate Source** dialog allows you to add a new Source Asset for your Aggregate.



**CONNECTOR**  
ora

**SOURCE TYPE**  
Collections

**SOURCE: CO\_EASEMENT**

**DTSTEST.CO\_EASEMENT**

Q filter

DTSTEST.AGG_STORE	DTSTEST.CONSUMER2
DTSTEST.CO_EASEMENT_EXTENT	DTSTEST.CS_AUDIT_HISTORY
DTSTEST.CS_AUDIT_HISTORY_ALTE	DTSTEST.CS_DRAFTING_AREA
DTSTEST.CS_DRAFTING_LINE	DTSTEST.CS_DRAFTING_TEXT
DTSTEST.CS_GPS_COORDINATE	DTSTEST.ED_OWNER_TYPE_NAME

**ADD** →

Add new Aggregate Source dialog

To add a new Aggregate Source, click on the **Add button**.

<b>Connector</b>	The name of the Connector that contains the Asset
<b>Source Type</b>	Allows you to choose a source type from the drop-down. The options are <b>Collections</b> or <b>Routines</b>
<b>Source</b>	Allows you to choose from assets that match the <b>Connector</b> and <b>Source Type</b>

### 3.2.4.3 Aggregate Asset Details Drawer

The Asset Details Drawer shows details about the Source Asset currently selected in the [Aggregate Assets](#)<sup>64</sup>.

Please see the [Aggregation Technical Guide](#)<sup>242</sup> for an explanation of Aggregate structure, sources and relationships.

NATIVE NAME

DTSTEST.CO\_EASEMENT\_ANNOTATION

FIELDS

ID

RWO\_ID

ANNOTATION

Native ID : ANNOTATION

Custom Name : ANNOTATION

Type : DTS\_GEOMETRY ( MDSYS.SDO\_GEOMETRY )

TEXT\_ANNOTATION

LOCATION

LEADER\_LINE

FK\_CO\_EASEMENT

Filters

RELATIONSHIPS

DTSTEST.CONSUMER

Aggregate details drawer

Native Name

Displays the native name of the selected Source Asset (i.e. its full given name in the source data model)

DTS Product Manual

© 2023 Realworld Systems B.V.

**Fields**

**For Collection Assets:** displays the fields (columns) available in the collection

**For Routine Assets:** displays the outputs of the Routine

**i** For simplicity, the rest of the page will refer to these items as fields, regardless if they are collection fields/columns or routine outputs/results.

**Fields**

**i** A **blue** field indicates that the field is included in the Aggregate.  
A **gray** field indicates that the field is available but not included in the Aggregate.

To edit a field, or view the available information about the field simply click on the field to expand.

To close the field information window click on the **Close**  icon.



**Deselect All**

Excludes all fields from the Aggregate, **except** fields that are involved on the parent side of existing relationships.



**Select All**

Includes all the available fields in the Aggregate.



**Show Custom Names**

Will show the fields' DTS Names in the Fields container's tiles.




**Show Native Identifier**

Will show the fields' Native Identifiers in the Fields container's tiles.

**i** Show Custom names and Show Native Identifiers can be active at the same time, and at least one of them needs to be active.

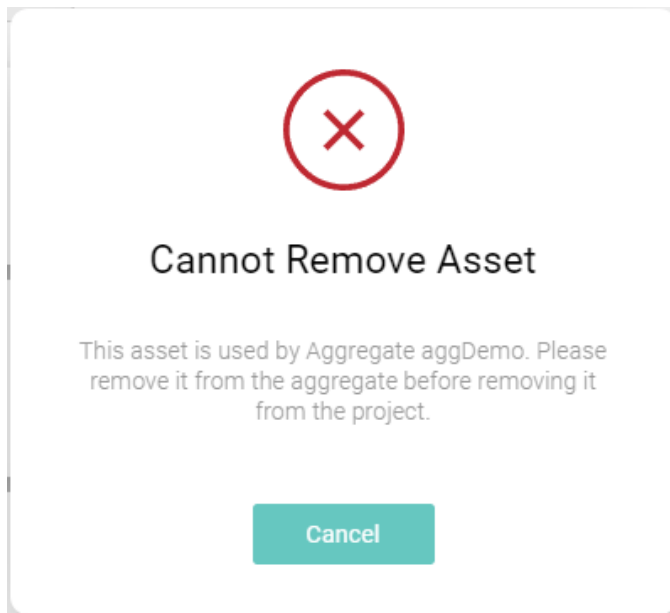
**i** A **blue** background of an input field indicates that the field is read-only and its contents cannot be modified.

A white background of an input field indicated that the value of the field can be modified.

To remove a field from the Aggregate simply click on the **delete**  button. The field will become gray.

To add a field to an Aggregate simply click on the gray field to change its state.

If the field is used in a relationship in the current Aggregate, it cannot be removed and an Alert Dialog will appear on the screen.



*Alert Dialog*

## Filters

This section allows you to set up, edit or remove fundamental filters on the source asset.

For Aggregates, fundamental filters are used to add static/constant terms to source asset queries.



**Add Filter**

Opens the Insert Filter dialog, allowing you to add a complex filter



**Edit Filter**

Opens the Edit Filter dialog, allowing you to edit an existing filter



**Remove Filter**

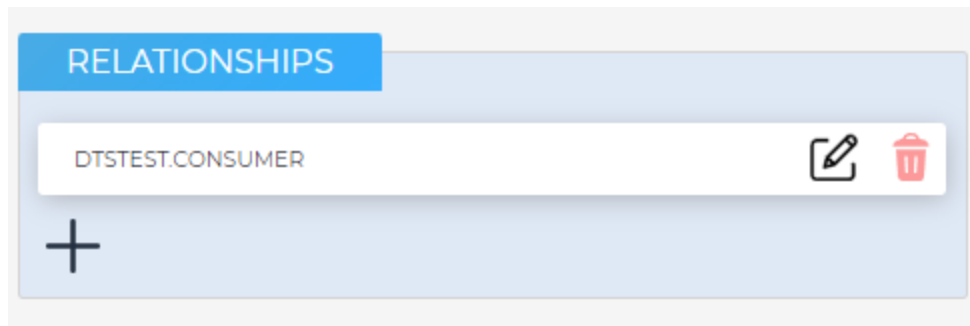
Removes an existing filter



For more information about filters, please check the [Filters](#)  section.


## Relationships

The Relationships sections allows you to Add, Remove and Edit relationships between Aggregate sources.



*Aggregate Relationships Section*

To add a new Relationship, click on the **Add Button** . This will open the Add Relationship dialog.

To edit or view details for a specific Relationship, click on the **Edit Button**  on the respective card. This will open the Edit Relationship dialog.

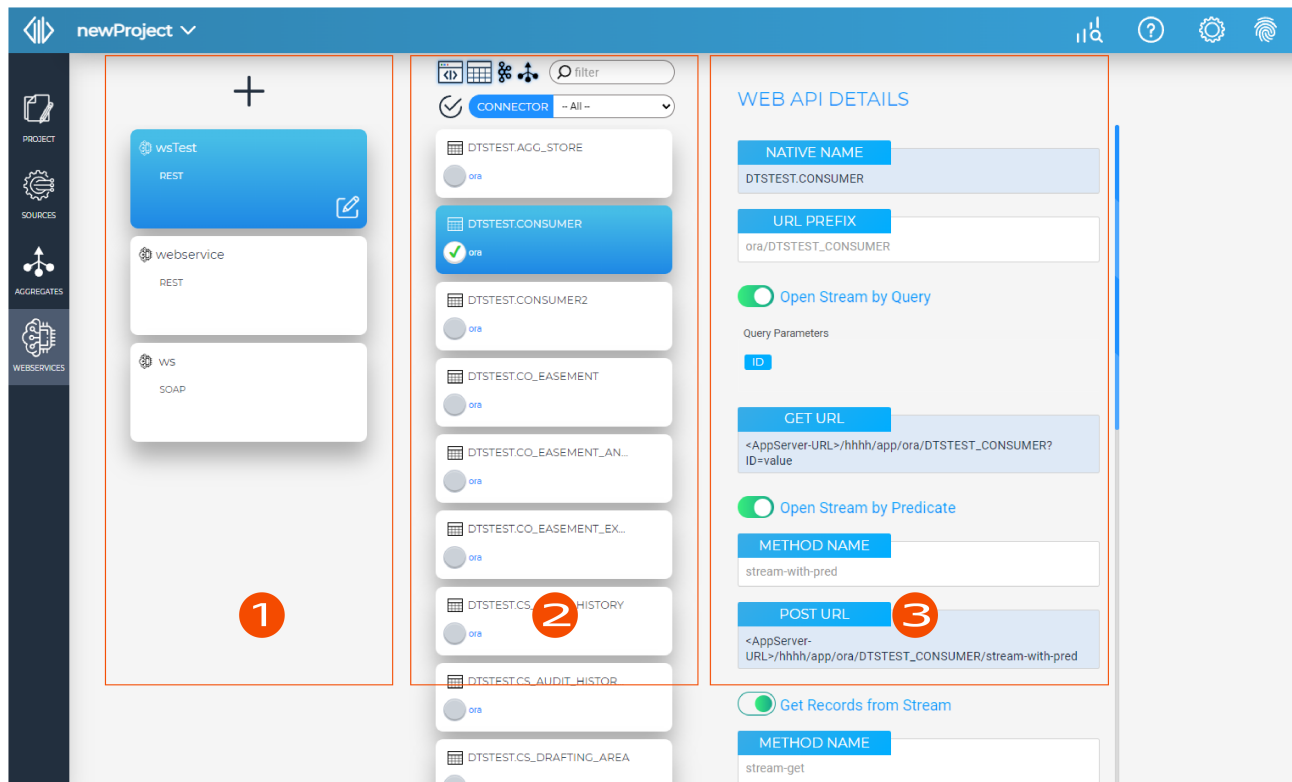
To delete an entry, click the Delete  button.

 For more information about Relationships, please check the [Filters & Relationships](#)  section.

### 3.2.5 Webservices

The Webservices view shows the DTS webservices created for a Project and the assets they expose. From here, you can create new webservices, choose what assets to include and how they will be accessed. This is also where you deploy webservices to an Application Server.

The view is split into 3 main parts:



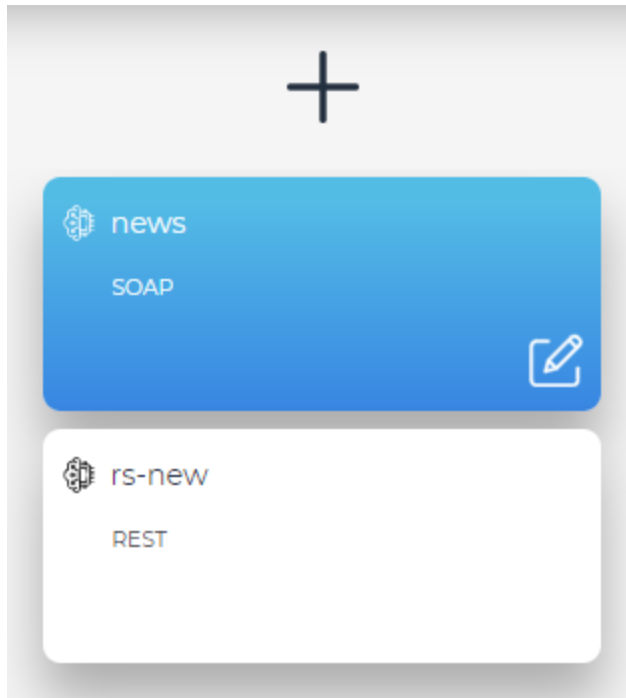
Webservices View

1. [Webservices Drawer](#)<sup>71</sup>
2. [Webservice Assets Drawer](#)<sup>76</sup>
3. [Webservice Asset Details Drawer](#)<sup>78</sup>

**i** For technical information regarding DTS Webservices, please see the [Webservices](#)<sup>186</sup> page of this manual.


### 3.2.5.1 Webservices Drawer

The Webservices Drawer lists all the DTS Webservices currently defined in the Project as a list of cards. From here you can create more Webservices or interact with existing ones.



*Webservices Drawer*

To add a new webservice, click on the **Add button** . This will open the [Create New Webservice](#) <sup>[72]</sup> dialog.

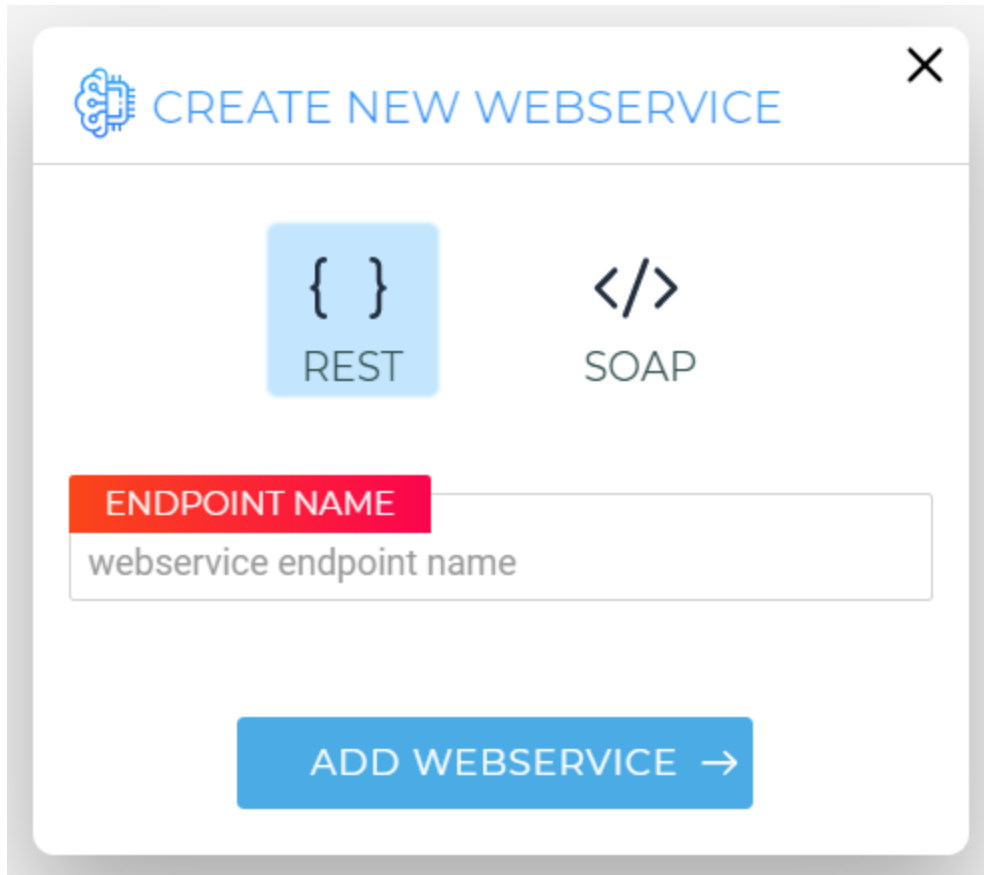
To edit or view details for a specific webservice, click on the **Edit button** . This will open the [Webservice Details](#) <sup>[73]</sup> dialog.

 Selecting a webservice card will open the [Webservice Assets Drawer](#) <sup>[76]</sup>.

 Activating a webservice card triggers an operation that might generate messages, errors or warnings. To read more about this subject go to [Errors & Warnings](#) <sup>[114]</sup>.

### 3.2.5.1.1 Create New Webservice

The Create New Webservice dialog allows you to create new DTS Webservices to expose assets included in the current Project.


A dialog box titled "CREATE NEW WEBSERVICE" with a close button (X) in the top right corner. Inside the dialog, there are two options for web service type: "REST" represented by a blue square with curly braces "{}" and "SOAP" represented by a code icon "</>". Below these options is a text input field with a red label "ENDPOINT NAME" and a placeholder text "webservice endpoint name". At the bottom of the dialog is a blue button labeled "ADD WEBSERVICE →".

*Create New Webservice Dialog*

<b>Webservice type</b>	Select the type for the new webservice (REST/SOAP)
<b>Endpoint Name</b>	Enter the endpoint name for the new webservice
<b>Add webservice</b>	Click the button to add the new webservice. A new webservice card will be added to the <a href="#">Webservices Drawer</a> <sup>[71]</sup>

### 3.2.5.1.2 Webservice Details

The Webservice Details dialog shows basic information about a given webservice and allows you to deploy or delete it.

 **ENDPOINT - NEWS** ✕

ADDRESS ELEMENT

app

BASE URL

<AppServer-URL>/news/app

DEPLOY FOR

DEV

TEST

PROD

DEPLOYMENT TYPE

DOWNLOAD

UPLOAD

DOWNLOAD WAR →

DELETE →

*Webservice Details Dialog - Download*

 **ENDPOINT - NEWS** ✕

ADDRESS ELEMENT

app

BASE URL

<AppServer-URL>/news/app

DEPLOY FOR

DEV

TEST

PROD

DEPLOYMENT TYPE

DOWNLOAD

UPLOAD

DEPLOYER

tomcat


DEPLOY

→

DELETE

→

*Webservice Details Dialog - Upload*

- |   |  |
|---|--|
| <b>Address Element</b>  | Due to particularities of various deployment platforms (Application Servers), an extra element is required in the service's base URL. By default, it is "app", but can be customized here. |
| <b>Base URL</b>   | Shows the base URL the service will be accessible at (view-only)   |
| <b>Deploy For</b>   | The environment this deployment targets  |
| <b>Deployment Type</b>  | How the service should be deployed   |
|  <b>Download</b> | The service will be packaged as a WAR and presented as a browser download on the <b>Download WAR</b> button  |

**Upload**

The service will be automatically deployed the selected **Deployer** upon clicking the **Deploy** button

**Delete**

Delete the Webservice from the Project

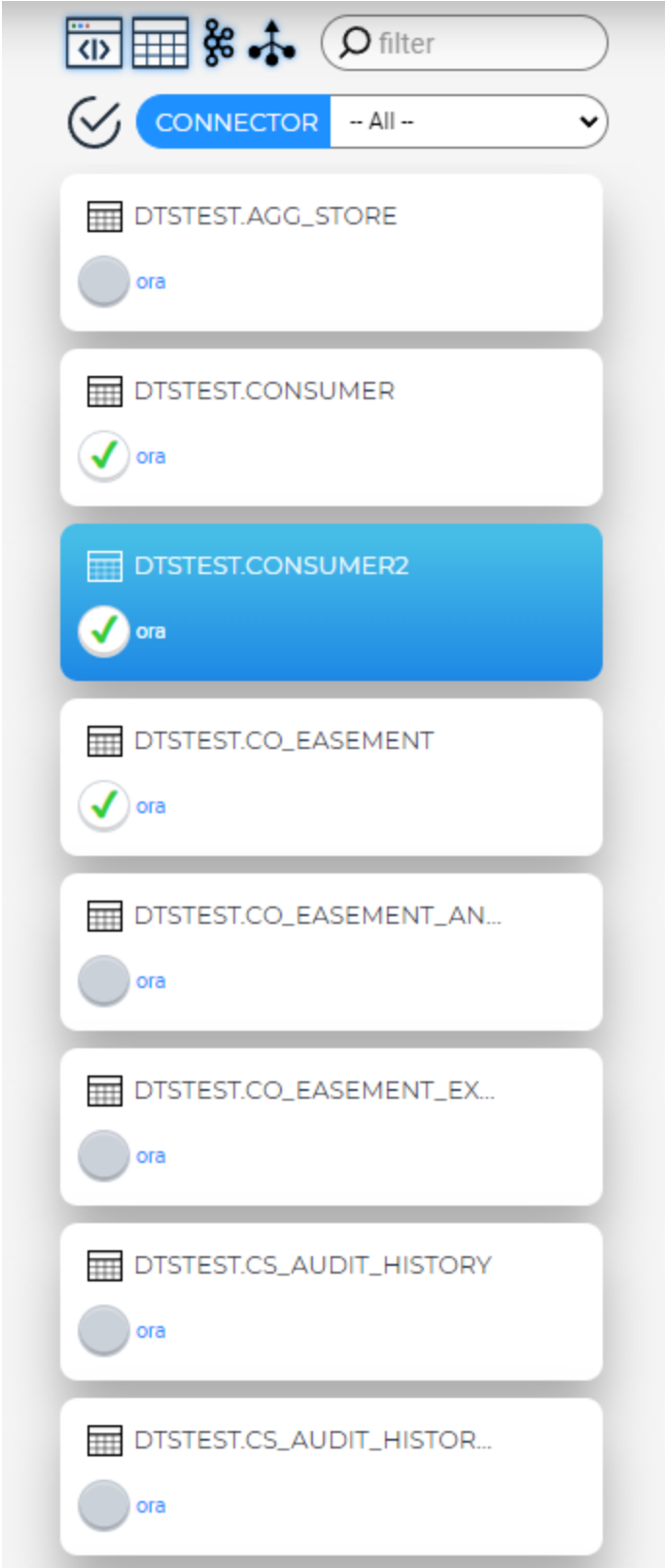


Webservice Deployers are configured using the [Webservice Deployer Dialog](#)

### 3.2.5.2 Webservice Assets Drawer

The Webservice Assets Drawer shows all the assets currently included in the Project as a list of cards and allows you to search and filter them as well as include or exclude them from the Webservice.

Items in the Asset Drawer can be either **Collections**  (tables, views, etc.), **Routines**  (methods, procedures, functions, etc.), **Topics**  or **Aggregates** .



Webservice Assets Drawer



**Show only selected toggle**

Shows only the items currently included in the Webservice



**Collections toggle**

Shows the collections



**Routines Toggle**

Shows the routines



**Topics Toggle**

Shows the topics



**Aggregates Toggle**

Shows the aggregates

**Search/ Filter bar**

Search for a specific item

**Connectors drop-down**

- **-- All --** : Displays all items (for all connectors) - this is the default setting
- **Aggregate**: Displays only aggregates
- **[Connector name entry]**: Displays only items associated with a specific connector

Additionally, the connector is displayed on each item card as a blue text button. Clicking on this button will filter all items by that specific connector.



Selecting a resource card will open the [Webservice Asset Details Drawer](#)<sup>[78]</sup>.





Activating a resource card triggers an operation that might generate messages, errors or warnings. To read more about this subject go to [Errors & Warnings](#)<sup>[114]</sup>.

### 3.2.5.3 Webservice Asset Details Drawer

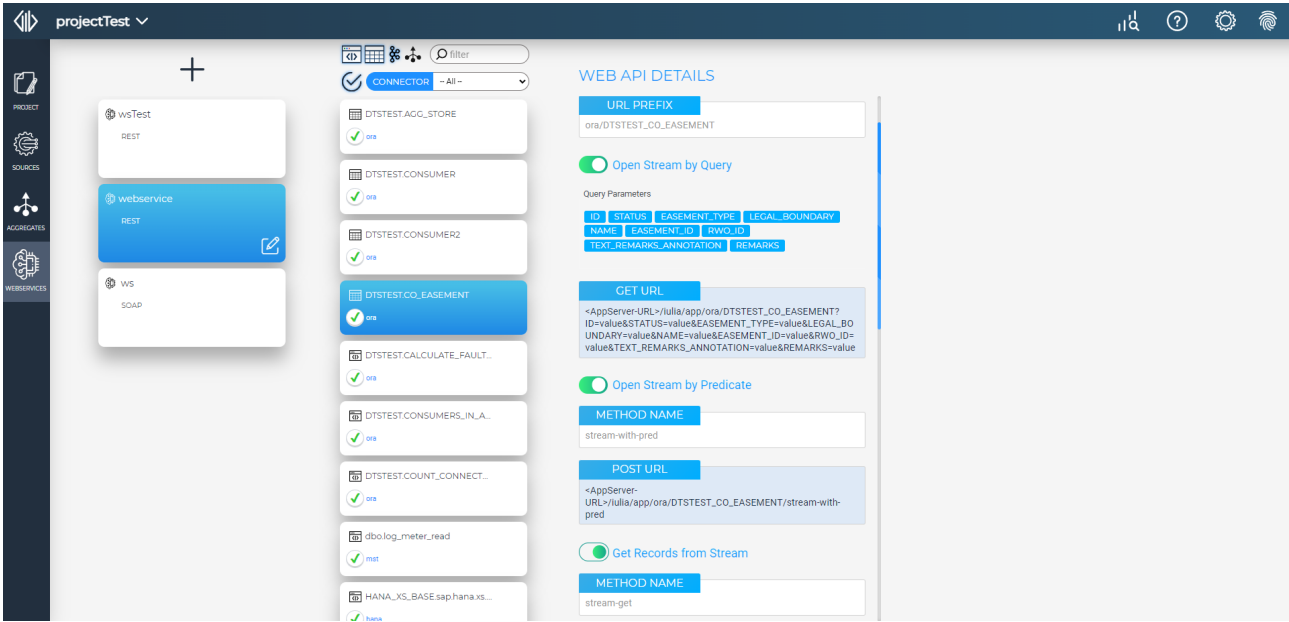
Depending on the selected resource type, the Webservices details drawer has two different presentations:

- [Webservice Stream Operations Drawer](#)<sup>[79]</sup> (for Collection and Aggregate Resources);
- [Webservice Routine Details Drawer](#)<sup>[86]</sup> (for Routines)
- [Webservice Topic Details Drawer](#)<sup>[89]</sup> (for Topics)

### 3.2.5.3.1 Webservice Stream Operations Drawer

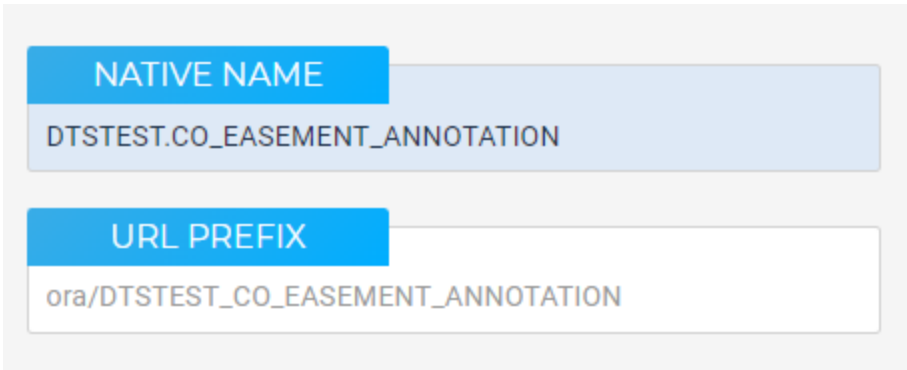
When a Collection  or Aggregate  is selected in the [Webservice Assets Drawer](#)<sup>76</sup>, the Details Drawer will show all the options for configuring Webservice operations that allow clients to interact with streams on that resource.

This allows fine tuning of the interface by which data from the resource is accessed using the Webservice.



Webservices View with Stream Operations Drawer

This section is the master control for all stream methods on this collection.



<b>Native Name</b>	Shows the full native name of the selected Resource (view-only)
<b>URL Prefix</b>	The URL element(s) that will be used to point to WS methods for this collection

**i** The default value of URL Prefix is [CONNECTOR\_NAME]/[COLLECTION\_NAME]

**i** The URL Prefix only identifies the targeted collection within the Webservice. To access a particular Stream Method, more elements must be added to the URL as described in the next paragraphs.

### Open Stream by Query

☒ Open Stream by Query

Query Parameters

ID STATUS EASEMENT\_TYPE LEGAL\_BOUNDARY  
NAME EASEMENT\_ID RWO\_ID  
TEXT\_REMARKS\_ANNOTATION REMARKS

GET URL

```
<AppServer-URL>/iulia/app/ora/DTSTEST_CO_EASEMENT?  
ID=value&STATUS=value&EASEMENT_TYPE=value&LEGAL_BO  
UNDARY=value&NAME=value&EASEMENT_ID=value&RWO_ID=  
value&TEXT_REMARKS_ANNOTATION=value&REMARKS=value
```

This section provides options for the Webservice method that will open a stream on the selected resource using a query based on in-line parameter values provided in the URL (i.e. a REST GET request)

<b>Open Stream by Query</b>	Toggles whether this method will be included in the Webservice
<b>Query Parameters</b>	Allows you to choose which fields you want to make available as query parameters within the GET method URL

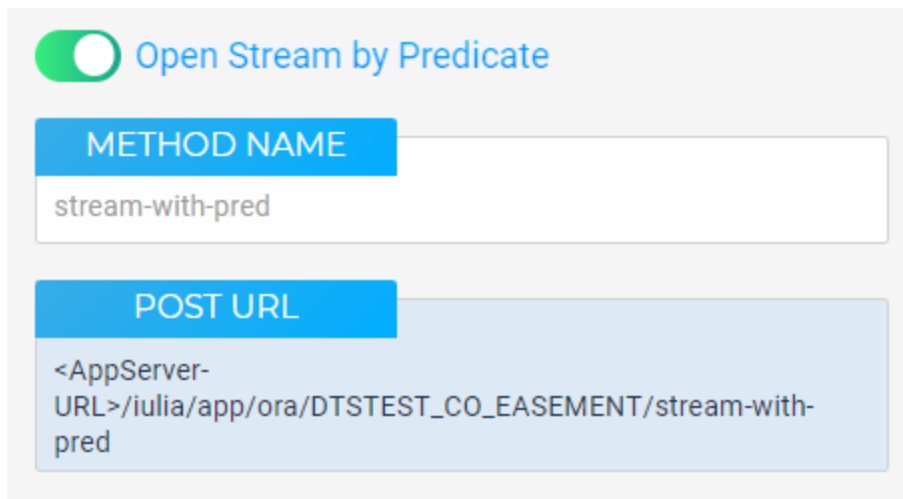
**i** Only fields that are compatible with URL parametrization will be listed (i.e. variants of STRING and INTEGER types)

**GET URL** Shows what the complete URL for calling the method will look like (view-only)

**i** All the parameters in the GET URL (after '?') are optional. Any combination can be used. If none are included, a stream spanning all records provided by the resource will be created.

**i** This method is only available for REST services, as SOAP does not allow URL parameters.

### Open Stream by Predicate



**Open Stream by Predicate**

**METHOD NAME**  
stream-with-pred

**POST URL**  
<AppServer-URL>/iulia/app/ora/DTSTEST\_CO\_EASEMENT/stream-with-pred

This section provides options for the Webservice method that will open a stream on the selected resource using a query based on a [DTS Predicate](#)<sup>[239]</sup> object which will be included in the request body (i.e. a REST POST or SOAP request).

**Open Stream by Predicate** Toggles whether this method will be included in the Webservice

**Method Name** The name you want the method to be accessed with (default "stream-with-pred")

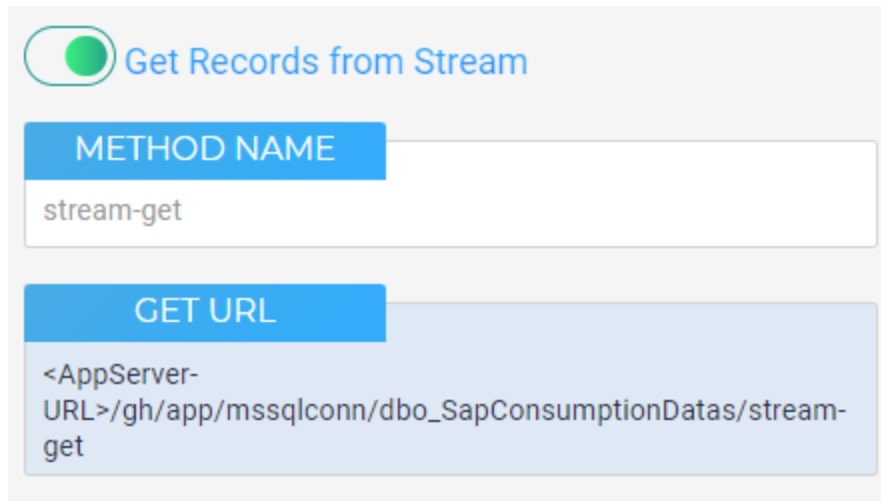
**POST URL** Shows what the complete URL for calling the method will look like (view-only)

**i** The Method Name serves as a URL element in the case of REST POST and as an actual method name for SOAP.

**⚠** When configuring a SOAP method, the Method Name needs to be a valid Java method name.

**i** The POST URL is only significant for REST, as SOAP methods are all accessed using the base endpoint URL.

### Get Records From Stream



☒ Get Records from Stream

**METHOD NAME**

stream-get

**GET URL**

<AppServer-URL>/gh/app/mssqlconn/dbo\_SapConsumptionDatas/stream-get

This section provides options for the Webservice method that will request records from an open stream.

<b>Get Records From Stream</b>	Shows if the method is included or not. This cannot be manually toggled, instead it will be on if any of the Open Stream methods are included and off otherwise.
<b>Method Name</b>	The name you want the method to be accessed with (default "stream-get")
<b>GET URL</b>	Shows what the complete URL for calling the method will look like (view-only)

**i** The Method Name serves as a URL element in the case of REST POST and as an actual method name for SOAP.

**⚠** When configuring a SOAP method, the Method Name needs to be a valid Java method name.

**i** The POST URL is only significant for REST, as SOAP methods are all accessed using the base endpoint URL.

**i** The Method always has the same 2 parameters: stream\_id and dts\_\_size, which are fed inline for REST GET requests and within the body for SOAP requests. The stream\_id parameter is mandatory, while dts\_\_size is considered 1 if missing.

## Get Records With Inline Parameters

☒ Get Records by Query

METHOD NAME

records

Query Parameters

ID

STATUS

EASEMENT\_TYPE

LEGAL\_BOUNDARY

NAME

EASEMENT\_ID

RWO\_ID

TEXT\_REMARKS\_ANNOTATION

REMARKS

GET URL

<AppServer-URL>/iulia/app/ora/DTSTEST\_CO\_EASEMENT/records?ID=value&STATUS=value&EASEMENT\_TYPE=value&LEGAL\_BO  
UNDARY=value&NAME=value&EASEMENT\_ID=value&RWO\_ID= value&TEXT\_REMARKS\_ANNOTATION=value&REMARKS=value

This section provides options for the Webservice method that will execute a one-time query on the resource based on in-line parameter values provided in the URL (i.e. a REST GET request)

**Get Records by Query** Toggles whether this method will be included in the Webservice

**Method Name** The name you want the method to be accessed with (default "records")

**Query Parameters** Allows you to choose which fields you want to make available as query parameters within the GET method URL

**i** Only fields that are compatible with URL parametrization will be listed (i.e. variants of STRING and INTEGER types)

**GET URL** Shows what the complete URL for calling the method will look like (view-only)

**i** All the parameters in the GET URL (after '?') are optional. Any combination can be used. If none are included, the first [dts\_size] records provided by the resource will be returned.

**i** This method is only available for REST services, as SOAP does not allow URL parameters.

**i** This method is equivalent to and a shortcut for opening a stream with the same parameters, making a records request of the desired size and closing the stream.

**i** The `dto__size` parameter is always available (if not used, a single record will be requested).

## Get Records by Predicate

☒ Get Records by Predicate

**METHOD NAME**  
 records-with-pred

**POST URL**  
 <AppServer-URL>/iulia/app/ora/DTSTEST\_CO\_EASEMENT/records-with-pred

This section provides options for the Webservice method that will execute a one-time query on the selected resource based on a [DTS Predicate](#)<sup>[239]</sup> object and a given size which will be included in the request body (i.e. a REST POST or SOAP request).

<b>Get Records by Predicate</b>	Toggles whether this method will be included in the Webservice
<b>Method Name</b>	The name you want the method to be accessed with (default "records-with-pred")
<b>POST URL</b>	Shows what the complete URL for calling the method will look like (view-only)

**i** The Method Name serves as a URL element in the case of REST POST and as an actual method name for SOAP.

**⚠** When configuring a SOAP method, the Method Name needs to be a valid Java method name.

**i** The POST URL is only significant for REST, as SOAP methods are all accessed using the base endpoint URL.

**i** This method is equivalent to and a shortcut for opening a stream with the same Predicate, making a records request of the desired size and closing the stream.

**i** The size parameter is considered to be 1 if missing.

## Get Record by Key

**Get Record by Key**

Key Field

**ID**

STATUS EASEMENT\_TYPE LEGAL\_BOUNDARY NAME  
EASEMENT\_ID RWO\_ID TEXT\_REMARKS\_ANNOTATION  
REMARKS

**GET URL**

<AppServer-URL>/hhhh/app/ora/DTSTEST\_CO\_EASEMENT/{ID}

This section provides options for the Webservice method that will execute a one-time query on the selected resource and return the first record found. It should be configured to use a unique field.

**Get Record by Key** Toggles whether this method will be included in the Webservice

**Key Field** The field that will be used as key for finding records

**i** Only fields that are compatible with URL parametrization will be listed (i.e. variants of STRING and INTEGER types)

**GET URL** Shows what the complete URL for calling the method will look like (view-only)


**i** One and only one field can be selected. If the resource has a declared primary key formed of a single field, it will be pre-selected.

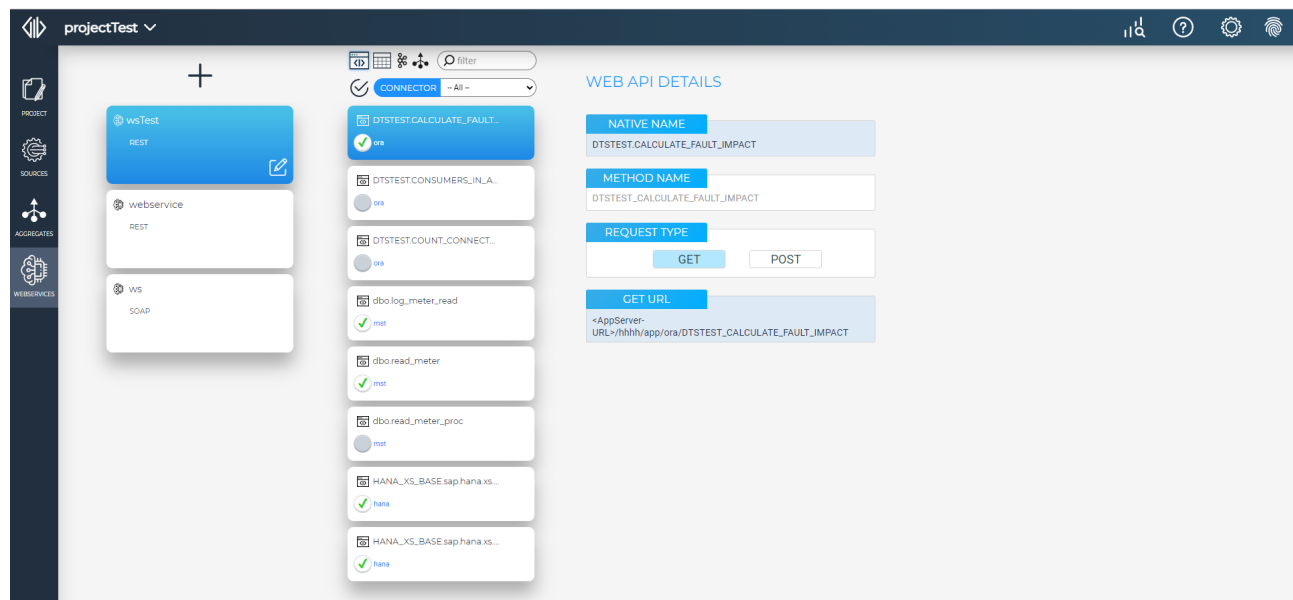
**i** This method is only available for REST services, as SOAP does not allow URL parameters.

**i** This method is equivalent to and a shortcut for opening a stream with the key parameter, making a one record request and closing the stream.

 To read more about the DTS URL format, see [Webservice Access](#)  195

### 3.2.5.3.2 Webservice Routine Details Drawer

When a Routine  is selected in the Webservice Assets Drawer, the Details Drawer will show the options for configuring the Webservice Method that calls the Routine.



*Webservices View with Method Details Drawer*

The following details are available for every routine:

### WEB API DETAILS

NATIVE NAME

DTSTEST.CALCULATE\_FAULT\_IMPACT

METHOD NAME

DTSTEST\_CALCULATE\_FAULT\_IMPACT

REQUEST TYPE

GET

POST

GET URL

<AppServer-URL>/hhhh/app/ora/DTSTEST\_CALCULATE\_FAULT\_IMPACT

<b>Native Name</b>	Shows the full native name of the selected Routine (view-only)
<b>Operation Name</b>	The name you want the method to be accessed with (by default, it will be the a URL and Java friendly adaptation of the Native Name)
<b>Request Type</b>	Toggles the request Type ( GET or POST)
<b>POST URL</b>	Shows what the complete URL for calling the method will look like (view-only)

The screenshot shows a configuration panel with two main sections. The top section, titled 'REQUEST TYPE', contains two buttons: 'GET' and 'POST'. The 'POST' button is highlighted in blue. The bottom section, titled 'QUERY PARAMS', contains a list of parameters. The parameter 'result' is highlighted in blue.

**Query Params** Allows you to choose which fields you want to make available as query parameters

**i** The Method Name serves as a URL element in the case of REST POST and as an actual method name for SOAP.

**i** GET is only available when all the input parameters of the routine can be codified as URL Query Parameters. By default, GET is set wherever possible.

**!** When configuring a SOAP method, the Method Name needs to be a valid Java method name.


**i** The POST URL is only significant for REST, as SOAP methods are all accessed using the base endpoint URL.

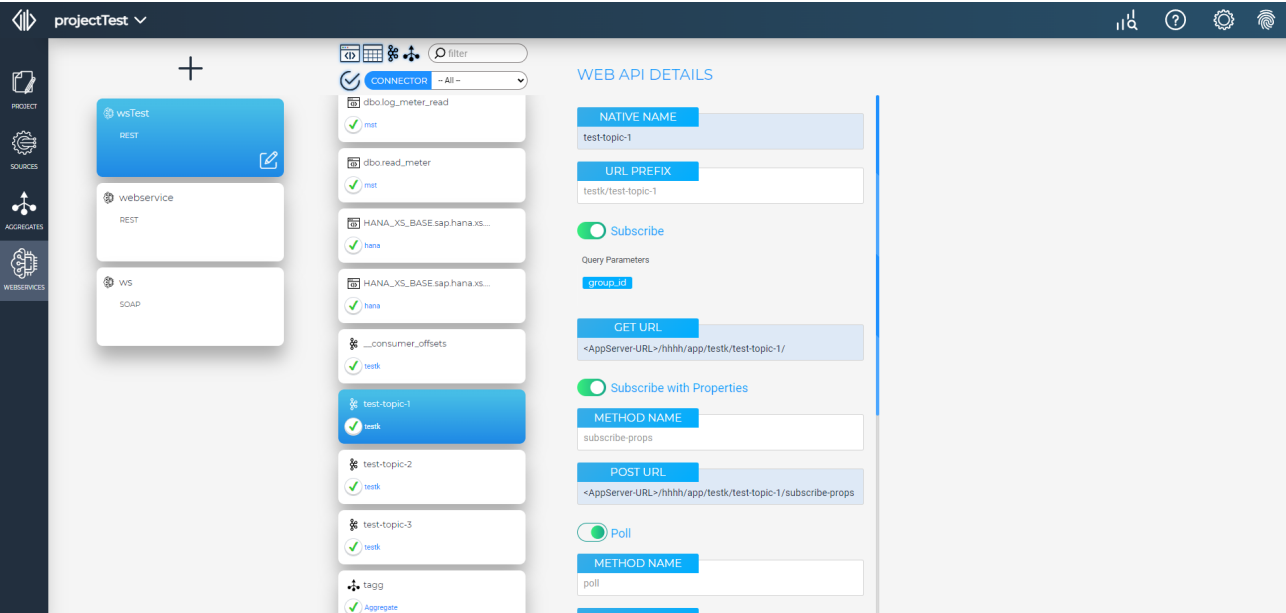
If the selected routine is "streamy" (i.e. returns a record stream), a method to get records from the resulting stream is necessary, so one more panel will be available:

The screenshot shows a configuration panel with a toggle switch labeled 'Get Records from Stream'. The toggle is turned on. Below the toggle is a text input field labeled 'METHOD NAME' containing the text 'streamGet\_ora\_DTSTEST\_TVF\_TABLE\_FUNC'.

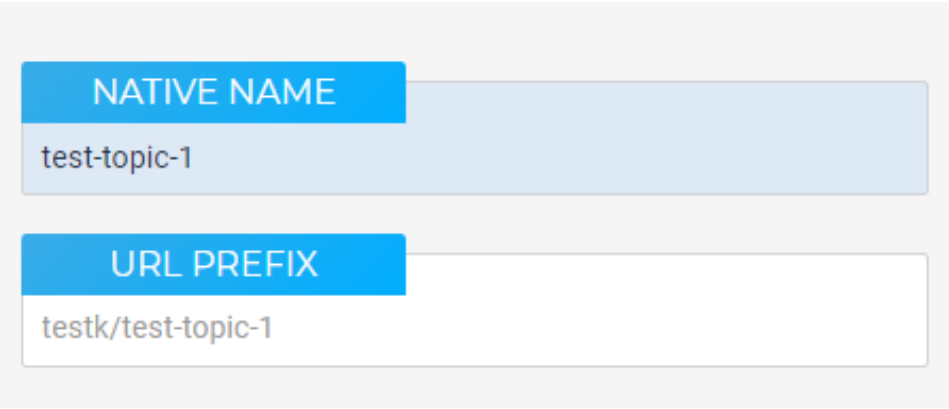
Here, the SOAP Method Name / REST URL element of that method can be customized.

3.2.5.3.3 Webservice Topic Details Drawer

When a Topic  is selected in the Webservice Assets Drawer, the Details Drawer will show the options for configuring the operations available for interacting with the Topic.




Webservices- Topic Details Drawer



- Native Name**
- Shows the full native name of the selected Resource (view-only)
- URL Prefix**
- The URL element(s) that will be used to point to WS methods for this topic

Subscribe

 Subscribe

Query Parameters

group\_id

GET URL


<AppServer-URL>/hhhh/app/testk/test-topic-1/


This section provides options for the Webservice method that will subscribe to the selected topic using the in-line parameter values provided in the URL (i.e. a REST GET request)

- Subscribe

Toggles whether this method will be included in the Webservice
- Query Parameters


A non-modifiable list of available query parameters.

 At present, only the group\_id parameter is available for use in this operation. To specify other parameters, please use **Subscribe with Properties**.

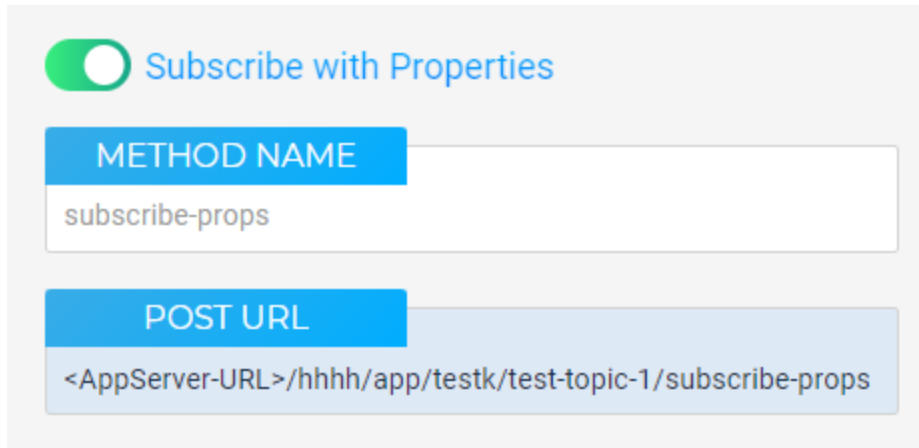
 If the group\_id is omitted, DTS will generate a temporary one for this subscribe action.

GET URL

Shows what the complete URL for calling the method will look like (view-only)

 This method is only available for REST services, as SOAP does not allow URL parameters.
- Subscribe with Properties
- DTS Product Manual

© 2023 Realworld Systems B.V.



This section provides options for the Webservice method that will subscribe to the selected topic using specific properties which will be included in the request body (i.e. a REST POST or SOAP request).

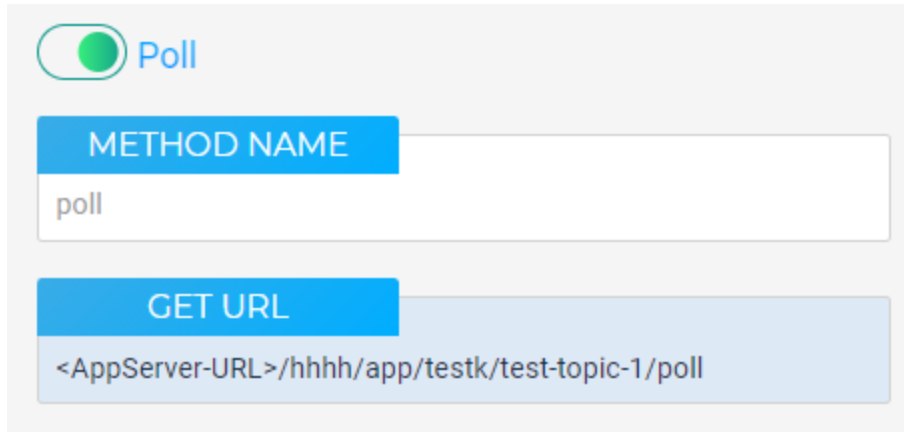
<b>Subscribe with Properties</b>	Toggles whether this method will be included in the Webservice
<b>Method Name</b>	The name you want the method to be accessed with (default "subscribe-props")
<b>POST URL</b>	Shows what the complete URL for calling the method will look like (view-only)

 The Method Name serves as a URL element in the case of REST POST and as an actual method name for SOAP.

 When configuring a SOAP method, the Method Name needs to be a valid Java method name.

 The POST URL is only significant for REST, as SOAP methods are all accessed using the base endpoint URL.

## Poll



☒ Poll

**METHOD NAME**

poll

**GET URL**

<AppServer-URL>/hhhh/app/testk/test-topic-1/poll

This section provides options for the Webservice method that will request records from a given topic.

<b>Poll</b>	Shows if the method is included or not. This cannot be manually toggled, instead it will be on if any of the Subscribe methods are included and off otherwise.
<b>Method Name</b>	The name you want the method to be accessed with (default "poll")
<b>GET URL</b>	Shows what the complete URL for calling the method will look like (view-only)

**i** The Method Name serves as a URL element in the case of REST POST and as an actual method name for SOAP.

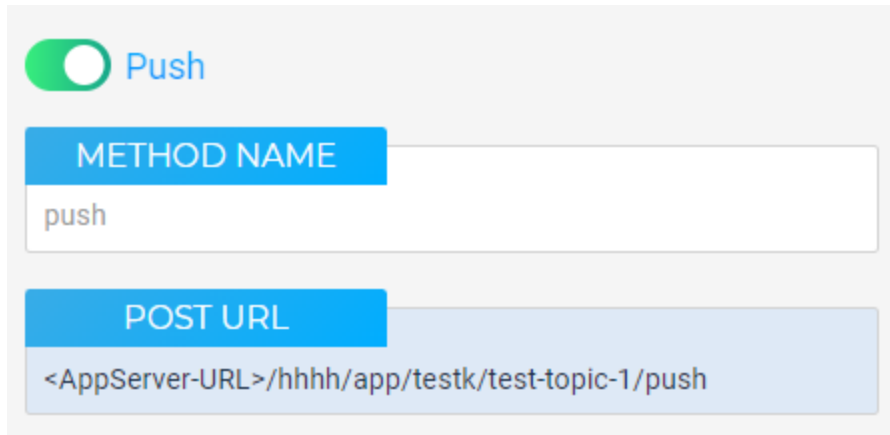
**⚠** When configuring a SOAP method, the Method Name needs to be a valid Java method name.

**i** The POST URL is only significant for REST, as SOAP methods are all accessed using the base endpoint URL.

**i** The Method always has the same 2 parameters: `stream_id` and `timeout`, which are fed inline for REST GET requests and within the body for SOAP requests.

The `stream_id` parameter is mandatory, while `timeout` will use the default value (configured in the connector) if missing.

## Push



This section provides options for the Webservice method that will read a message from the selected Topic.

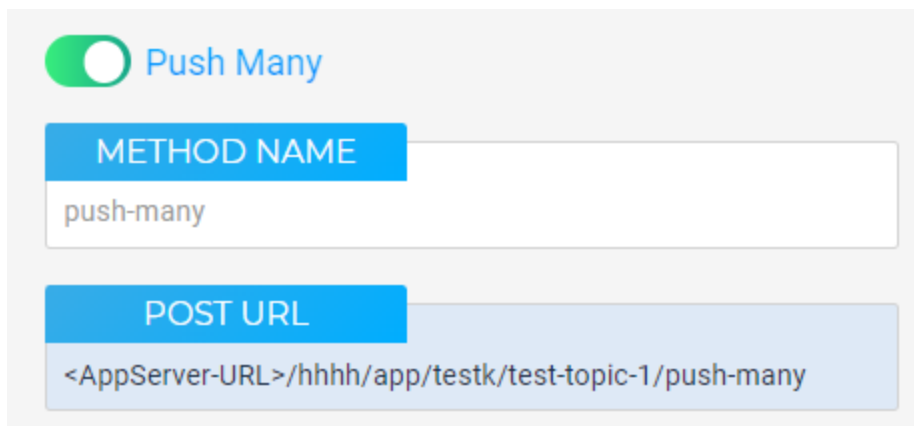
<b>Push</b>	Toggles whether this method will be included in the Webservice
<b>Method Name</b>	The name you want the method to be accessed with (default "push")
<b>POST URL</b>	Shows what the complete URL for calling the method will look like (view-only)

 The Method Name serves as a URL element in the case of REST POST and as an actual method name for SOAP.

 When configuring a SOAP method, the Method Name needs to be a valid Java method name.

 The POST URL is only significant for REST, as SOAP methods are all accessed using the base endpoint URL.

## Push Many



This section provides options for the Webservice method that will read multiple messages from the selected Topic.

<b>Push Many</b>	Toggles whether this method will be included in the Webservice
<b>Method Name</b>	The name you want the method to be accessed with (default "push-many")
<b>POST URL</b>	Shows what the complete URL for calling the method will look like (view-only)

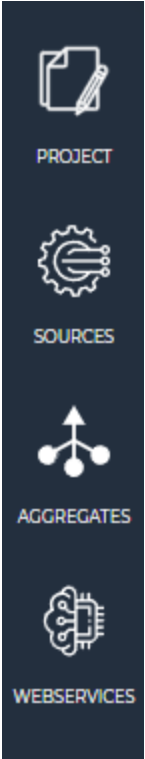
 The Method Name serves as a URL element in the case of REST POST and as an actual method name for SOAP.

 When configuring a SOAP method, the Method Name needs to be a valid Java method name.

 The POST URL is only significant for REST, as SOAP methods are all accessed using the base endpoint URL.

### 3.2.6 Left-Side Menu Toolbar

The Left-Side Menu Toolbar is available once a Project has been selected in the [Workspace](#)<sup>[27]</sup>. It allows navigation through the sections of a Project.



<b>Project</b>	Takes you to the <a href="#">Project Information</a> <sup>[30]</sup> view
<b>Sources</b>	Displays the <a href="#">Sources</a> <sup>[41]</sup> view for the selected project
<b>Aggregates</b>	Displays the <a href="#">Aggregates</a> <sup>[59]</sup> view for the selected project
<b>Webservices</b>	Displays the <a href="#">Webservices</a> <sup>[70]</sup> view for the selected project


### 3.3 Top Menu Toolbar





The Top menu toolbar is always situated in the top part of the DTS window.



Top menu toolbar

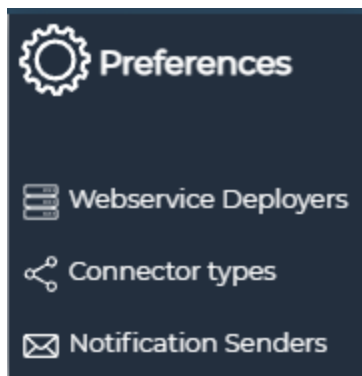
It contains the following buttons and context menus:

 <b>Home</b>	Takes you back to the <a href="#">Workspace</a> <sup>[27]</sup> page
---	--




 <b>Documentation</b>	Opens the Help Manual for the DTS application
 <b>Preferences</b>	Opens the <a href="#">System Preferences Menu</a> <sup>[96]</sup> (system-wide settings)
 <b>User</b>	Opens the <a href="#">User Menu</a> <sup>[111]</sup> (user settings)
 <b>Active Resources</b>	Opens the <a href="#">Active Resources</a> <sup>[40]</sup> list (status of your published projects)

### 3.3.1 Preferences Menu

This Menu presents the DTS System Preferences.




The available actions are:

 <b>Webservice Deployers</b>	Opens a separate <a href="#">Webservice Deployers</a> <sup>[97]</sup> dialog
 <b>Connector Types</b>	Opens a separate <a href="#">Connector Types</a> <sup>[104]</sup> dialog
 <b>Notification Senders</b>	Opens a separate <a href="#">Notification Senders</a> <sup>[109]</sup> dialog

3.3.1.1 Webservice Deployers

The Webservice Deployers dialog allows you to create a new deployment entity within DTS, which can then be used to automatically deploy Webservices from their [Details Dialog](#)<sup>73</sup>.

 **WEBSERVICE DEPLOYERS**

Name

Type


App Server


+

deployTest

TomcatHTTP


hostname1






localTest

Local







scpTest


SCP

hostname2





Add a new Webservice Deployer

To add a new Webservice Deployer, begin by clicking the  **Add button** . This will reveal the Add new Webservice Deployer section.

**WEBSERVICE DEPLOYERS**

Name	Type	App Server
<div><div>webservice deployer name</div><div>-- Select Deployer Type --</div><div>appServer hostname</div><div></div></div>		
<div>Port</div> <div>appServer host port</div>		<div>Copy path</div> <div>Deployment path</div>
<div>Host username</div> <div>Host username</div>		<div>Admin username</div> <div>appServer username</div>
<div>Host password</div> <div>Host password</div>		<div>Admin password</div> <div>appServer password</div> <div></div>
<div>deployTest</div>	<div>TomcatHTTP</div>	<div>hostname1</div> <div></div> <div></div>
<div>localTest</div>	<div>Local</div>	<div></div> <div></div> <div></div>
<div>scpTest</div>	<div>SCP</div>	<div>hostname2</div> <div></div> <div></div>

Add new Webservice Deployer section

Select the **Webservice Deployer type** from the drop-down and fill in the required fields.

The dialog contains the following fields:

- [Webservice Deployer] Type\***

The type of deployment that should be executed when this deployer is used.
- [Webservice Deployer] Name\***

The name given to the deployer (this is how it will be known within DTS). The name must be unique.
- Hostname\***

The hostname or IP address of the machine running the application server
- Port**

The port used to access the administrative functions of the application server
- Host Username**

The username used for logging onto the application server host machine (using SSH)
- Host Password**

The password for the Host Username

<b>Copy path</b>	The absolute path on the application server where the webservice will be copied
<b>Admin Username</b>	The username used for logging into the application server software as an administrator
<b>Admin Password</b>	The password for App Server Username

To save an entry, click the Save  button.

 The save button will only become available when all the required fields are filled.

 Please note that the Webservice Deployer name must be unique.

To delete an entry, click the Delete  button.

 \*These parameters are always required

 The fields that are required/available depend on the Deployment Type selected in the Deployer Type field, as does the interpretation of certain fields.

## Deployer Types

### SCP

SCP deployment will attempt to copy the WAR archive containing the Webservice to the application server machine via SCP. This requires the application server to feature automatic deployment from a watched directory.

It requires the following parameters:

<b>Hostname</b>	The hostname or IP address of the machine running the application server
<b>Host Username</b>	The username used for logging onto the application server host machine (using SSH)
<b>Host Password</b>	The password for the Host Username
<b>Deployment path</b>	The directory the application server watches for deployed service changes

 If the application server is configured to only pick up changes to the watched folder on startup, a restart will be required.

## Samba

Samba deployment will attempt to copy the WAR archive containing the Webservice to the application server machine via Samba. This requires the application server to feature automatic deployment from a watched directory.

It requires the following parameters:

<b>Hostname</b>	The hostname or IP address of the machine running the application server
<b>Host Username</b>	The username used for logging onto the application server host machine (Windows Login)
<b>Host Password</b>	The password for the Host Username
<b>Deployment path</b>	The directory the application server watches for deployed service changes

 If the application server is configured to only pick up changes to the watched folder on startup, a restart will be required.

## TomcatHTTP

TomcatHTTP deployment uses a two-stage method of deployment to a Tomcat application server. In the first stage, it will copy the WAR file to a given path on the application server machine and in the second stage it will issue an HTTP request to Tomcat to deploy the Webservice from that local path.

It requires the following parameters:

<b>Hostname</b>	The hostname or IP address of the machine running the application server
<b>Port</b>	The port used to access Tomcat's administrative functions (by default 9990)
<b>Host Username</b>	The username used for logging onto the application server host machine (using SSH)
<b>Host Password</b>	The password for the Host Username
<b>Deployment path</b>	The absolute path on the application server where the webservice will be copied (can be any path that both Host Username and the Tomcat OS user can access and Host Username can write to)
<b>App Server Username</b>	A username with enough administrative privileges on the Tomcat system to deploy webservices
<b>App Server Password</b>	The password for App Server Username

**i** This deployment method is useful when using a Tomcat server that doesn't watch a directory or when connecting via SSH/Samba with a user that can write to the watched directory is not possible.

## JBossCLI

JBossCLI deployment uses the remote console tool provided by JBoss/Wildfly application servers to deploy services.

It requires the following parameters:

<b>Hostname</b>	The hostname or IP address of the machine running the application server
<b>Port</b>	The port used to access the JBoss/Wildfly administrative functions (by default 9990)
<b>App Server Username</b>	A username with enough administrative privileges on the JBoss/Wildfly system to deploy webservice
<b>App Server Password</b>	The password for App Server Username
<b>CLI Executable Path</b>	The absolute path to a locally available jboss-cli.jar

**i** This deployment method is useful when using a JBoss/Wildfly server that doesn't watch a directory or when connecting via SSH/Samba with a user that can write to the watched directory is not possible.

## Weblogic

Weblogic deployment uses a two-stage method of deployment to an Oracle Weblogic application server. In the first stage, it will copy the WAR file to a given path on the application server machine and in the second stage it will use SSH to log into the same machine and use the Weblogic administration CLI on the server to deploy the webservice from the temporary path.

<b>Hostname</b>	The hostname or IP address of the machine running the application server
<b>Host Username</b>	The username used for logging onto the application server host machine (using SSH)
<b>Host Password</b>	The password for the Host Username
<b>Deployment path</b>	The absolute path on the application server where the webservice will be copied (can be any path that both Host Username and the Weblogic OS user can access and Host Username can write to)
<b>App Server Username</b>	A username with enough administrative privileges on the Weblogic system to deploy webservice

<b>App Server Password</b>	The password for App Server Username
<b>CLI Executable Path</b>	The absolute path to a the Weblogic CLI tool on the app server machine

**i** This deployment method is useful when using a Weblogic server that doesn't watch a directory or when connecting via SSH/Samba with a user that can write to the watched directory is not possible.

## Websphere

Websphere deployment uses a two-stage method of deployment to an IBM Websphere application server. In the first stage, it will copy the WAR file to a given path on the application server machine and in the second stage it will use SSH to log into the same machine and use the Websphere administration CLI on the server to deploy the webservice from the temporary path.

<b>Hostname</b>	The hostname or IP address of the machine running the application server
<b>Host Username</b>	The username used for logging onto the application server host machine (using SSH)
<b>Host Password</b>	The password for the Host Username
<b>Deployment path</b>	The absolute path on the application server where the webservice will be copied (can be any path that both Host Username and the Websphere OS user can access and Host Username can write to)
<b>App Server Username</b>	A username with enough administrative privileges on the Websphere system to deploy webservices
<b>App Server Password</b>	The password for App Server Username
<b>CLI Executable Path</b>	The absolute path to a the Websphere CLI tool on the app server machine

**i** This deployment method is useful when using a Websphere server that doesn't watch a directory or when connecting via SSH/Samba with a user that can write to the watched directory is not possible.

**i** Please see [Webservices->Integration->Application Server](#) <sup>199</sup> for technical details and particularities regarding different Application Servers.

## Edit a Webservice Deployer



### WEBSERVICE DEPLOYERS

Name	Type	App Server		
+				
deployTest	TomcatHTTP	hostname1	⬆	🗑
Port	8080	Copy path	/tmp/path	
Host username	user	Admin username	admin	
Host password	*****	Admin password	*****	💾
localTest	Local		⬇	🗑
scpTest	SCP	hostname2	⬇	🗑

*Edit Webservice Deployer*

To edit an existing Webservice Deployer click on the respective row to view its details.


If a field has been modified, the corresponding label will turn green.

**⚠ Please note that the Webservice Deployer type cannot be modified.**

**⚠ Please note that the Webservice Deployer name must be unique.**

If some of the fields are empty or have incorrect values, their respective labels will turn red and the save button will be disabled.

The fields that contain the initial value will have a blue label.

To save your changes, click the Save  button.

**i The save button will only become available when all the required fields are filled.**

To delete an entry, click the Delete  button.

deployTest		TomcatHTTP			
Port	8080	Copy path	/tmp		
Host username	user	Admin username	admin		
Host password	*****	Admin password	*****		

*disabled save button*

### 3.3.1.2 Connector Types

The Connector Types Dialog lets you manage the Connector Types available in your Projects. These connector types must be based on an existing Category (either one of the [built-in connectors](#)<sup>118</sup>, or a registered custom one).

×

Category	Variety	Used By
+		
<div><div>Oracle</div><div>:</div><div>v1</div><div>⌵</div></div> <div><div>ENVIRONMENT →</div><div>demo / ora demo / ora2</div></div> <div><div>VOLUMES →</div></div>		
SAPHana	v1	demo/hana ⌵
WebService	v1	demo/weather ⌵
Smallworld	PNi4	⌵

Connector Types dialog

The existing elements in the **Connector Types list** are read-only - their fields cannot be modified.

The **Used By section** shows the list of Connectors of each specific type defined in the environment in the form [Project Name]/[Connector Name].

The **⌵ expand symbol** will be present for each item on the list.

To expand this list, click on its respective row/tile. The expanded section contains buttons for opening the **Environment** and **Volumes** editors for the selected connector type, as well as a full expanded list of the project/connectors that are currently using it.

To delete an entry, click the Delete button.

**Please note that you can only delete connector types that are not currently in use.**

Smallworld

:

PNI4

⬆

🗑

ENVIRONMENT →

⚠ This connector type is not currently in use

VOLUMES →

connector type not in use

Add new Connector Type

To add a new **Connector Type** click on the **+ Add button**.  
Select a **Category** from the drop-down , then enter a **Variety**. The Variety is used to differentiate between Connector Types of the same Category. It can be anything, but must be unique for the Category.

-- Select Category --

:

connector variety

🗑

ENVIRONMENT

💾

VOLUMES

Add New Connector Type

- Category \*

The base Category for the Connector Type.
- Variety \*

The name you wish to call this particular variety of the [Category] connector type
- i

 Connector Types defined here will henceforth be know throughout DTS as [Category]:[Variety]
- Environment & Volumes

These buttons are disabled when creating a new Connector Type, but become accessible once the entry is saved.
- i

 Connector Types will inherit all the parameters from their Category as well

To save an entry, click the Save  button.

 The save button will only become available when all the required fields are filled.


 \* These fields are required

 Connector Types are required for creating Connectors within projects. A Connector cannot directly use a Connector Category.

## Edit a Connector Type

The only editable properties for a Connector Type are its lists of **Environment Variables** and **Volume Mappings**, which are accessible by clicking the **Environment** and **Volumes** buttons respectively.




✕


ENVIRONMENT



---

Variable Name
Value

+


DTS_WS_PRODUCER_WSDL2.JAVA_PAT1	=	/usr/local/wss/producer/tools/apache-c	
DTS_WS_PRODUCER_WADL2.JAVA_PAT1	=	/usr/local/wss/producer/tools/apache-c	
DTS_WS_PRODUCER_OPENAPI_GENERA	=	/usr/local/wss/producer/tools/openapi-	
DTS_WS_PRODUCER_OUTPUT_DIR	=	/usr/local/wss/producer/out	

SAVE →


The **Environment Variables Editor** allows you the **Edit**,  **Add** and  **Delete** System Environment Variables that will be set in all containers running this Connector Type. They present

as Key-Value Pairs where the key is the name of the variable and the value is the value assigned to the variable.


Environment variables are generally used for setting useful file paths or parameters for the Producer running in the container.

✕

Host Path	Container Path
+	
/d/DTS/dist/2022_1/wss/producer	-> /usr/local/wss/producer



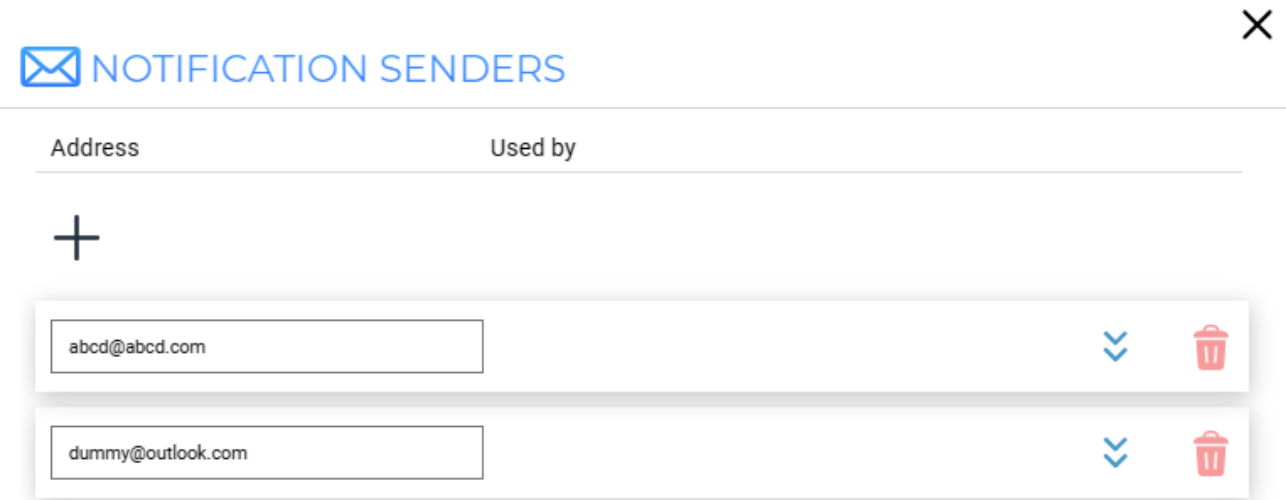
SAVE →

The Volume Mappings Editor allows you to **Edit**, **+** **Add** and  **Delete** Volume Mappings between the host machine and the containers running this Connector Type. They present as Key-Value Pairs where the key is a path on the Host Machine and the value is the mapped path inside Producer Containers.

Volume Mappings are generally used to provide access to files on the host to the Producer running in the container.

3.3.1.3 Notification Senders

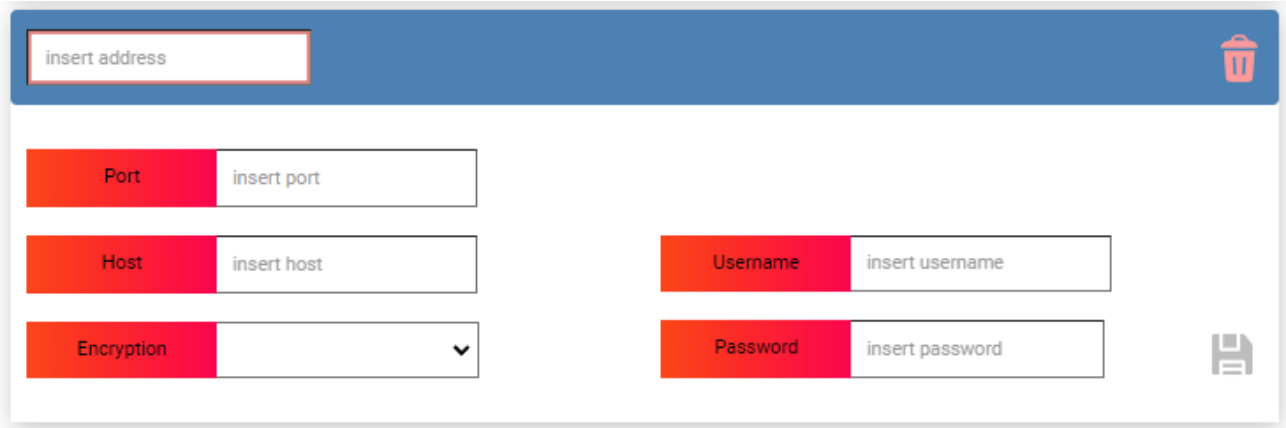
The Notification Senders dialog allows you to configure email accounts within DTS, which can then be used to send [Project Notifications](#)<sup>[32]</sup>.



Notification Senders Dialog

Add a new Notification Sender

To add a new Notification Sender, begin by clicking the **+** **Add button** . This will reveal the Add new Notification Sender section.



Add new Notification Sender

The dialog contains the following fields:

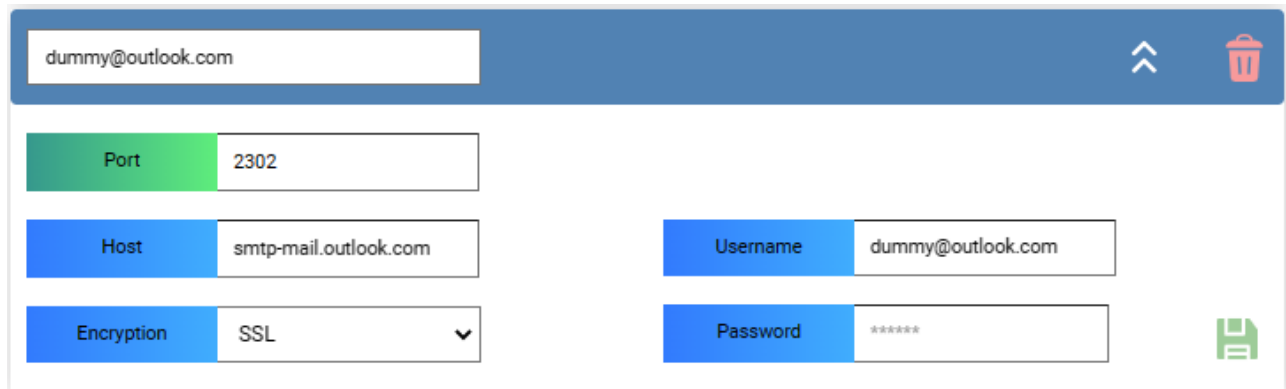
<b>[Notification Sender] Address</b>	The email address of the account (this will double as the identifier for the sender within DTS).
<b>Host</b>	The SMTP server hostname.
<b>Port</b>	The SMTP server port.
<b>Encryption</b>	The type of encryptions to be used (none, SSL, STARTTLS)
<b>Username</b>	The SMTP username.
<b>Password</b>	The SMTP password.

To save an entry, click the Save  button.

 **The save button will only become available when all the fields are filled.**

 **Only one sender can be configured for any given email address.**

## Edit a Notification Sender




To edit an existing Notification Sender click on the respective row to view its details.

If a field has been modified, the corresponding label will turn green.

If some of the fields are empty or have incorrect values, their respective labels will turn red and the save button will be disabled.

The fields that contain the initial value will have a blue label.

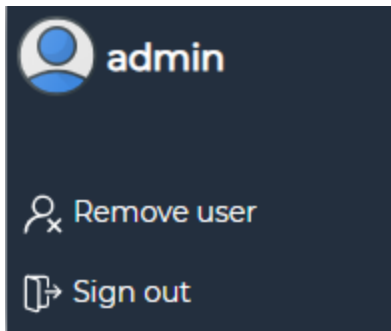
To save your changes, click the Save  button.

 **The save button will only become available when all the required fields are filled.**

To delete an entry, click the Delete  button.

### 3.3.2 User Menu

This menu presents the current user and its available actions.



Removes a user account from the database



Click on this button to sign out of the DTS application and go back to the [Login](#) <sup>261</sup> page

## 3.4 Filters & Relationships

---

For DTS, Filters and Relationships are manifestations of the same underlying objects - [Predicates](#) <sup>239</sup>. However, differences in how they are used warrant treating them separately.

### Filters

Filters are used to define complex query clauses on DTS resources, generally used to limit which records from those resources DTS will provide to its consumers.



INSERT FILTER



Operation: 
 Negated: ☐

Operation: 
 Negated: ☒

Field: 
 Value:

Operation: 
 Negated: ☐

Field: 
 Value:

ACCEPT

*Filter dialog*

<b>Operation</b>	Allows you to select a specific operation from the drop-down
<b>Negated</b>	Allows you to set the value of the Negated attribute
<b>Field</b>	Allows you to select a specific field from the drop-down, on which to apply a filtering operation.
<b>Value</b>	Allows you to specify a constant value for the selected field to be checked against.

**i** The available operations are: AND, OR, EQUALS, LIKE, GREATER, LOWER, GREATER or EQUAL, LOWER or EQUAL.

**i** To save your changes, click on the Accept button.

Operations can be compounded using the **AND** and **OR** operators to virtually no limit of complexity.


The Negated checkbox controls whether the particular clause in the filter will be negated, for example:

- EQUALS + Negated = NOT EQUALS
- GREATER + Negated = LOWER or EQUAL
- OR + Negated = NOR

**i** Please see the [Predicates Technical Guide](#) <sup>238</sup> for more information regarding filter composition.

Relationships

The same mechanism that is used for defining Filters is used to define Aggregate Relationships, with a few notable differences.

 RELATIONSHIP

PARENT

DTSTEST.CONSUMER

Operation: EQUALS

Negated: ☐

Child Field: RW0\_ID

Parent Field: ID

ACCEPT

Aggregate relationship

Firstly, the Relationship Editor provides a choice of **Parent**. This sets the Aggregate source to target with the relationship.


Secondly, since for a relationship we need to compare two attributes from resource records, rather than an attribute to a constant value, the **Value** box from Filter is replaced with a **Parent Field** dropdown which presents the attributes available in the **Parent** resource.

Lastly, since relationships are used to simulate Foreign Keys or to feed routine inputs, the **Operation** can only be **EQUALS**.


Operation	For relationships, this can only be <b>EQUALS</b>
Negated	Allows you to set the value of the Negated attribute (unused).
Child Field	Allows you to select a specific child field from the drop-down.  Child fields are the Query Parameters of the Aggregate Source that the relationship is defined on: fields for Collections and arguments for Routines.
Parent Field	Allows you to specify a Parent Field to be fed into the Child Field.

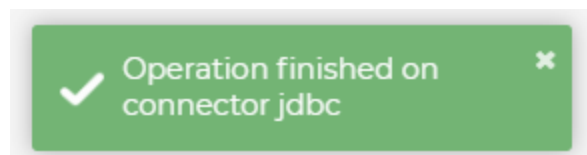
Parent fields are Attributes of the selected Parent resource: fields for Collections and outputs for Routines.

 Please see the [Aggregation Technical Guide](#) <sup>242</sup> for more information on Relationships.

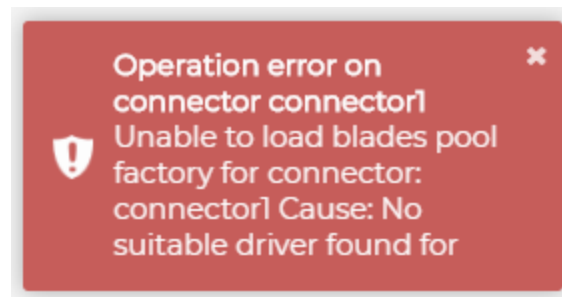
 To save your changes, click on the Accept button.

## 3.5 Errors & Warnings

Every time an action that requires loading time is performed (such as opening a connector or loading a collection) an **Animated Loader**  will appear at the bottom of the selected card to indicate the operation is still in progress. After the operation is finished, the DTS UI will display, depending on the operation's success, a message, a warning or an error on the bottom right hand corner of the screen.



*Successful operation message example*

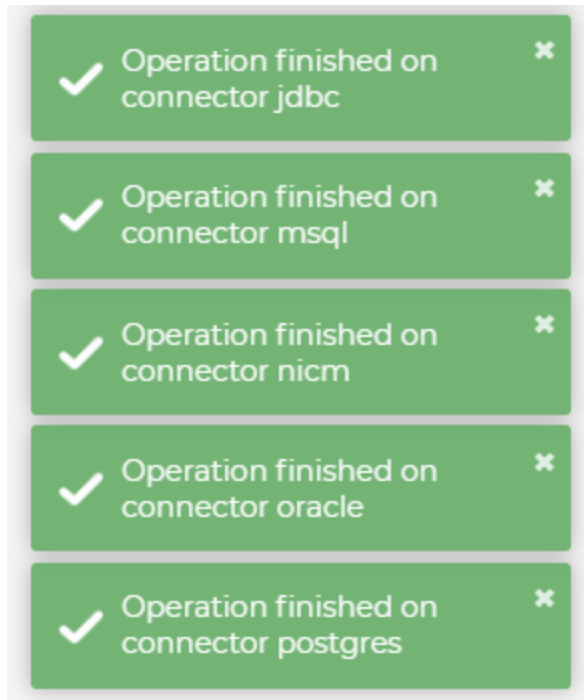


*Error message example*




*Info message example*

 The messages will disappear after a few seconds. If you have multiple operations running at the same time, the messages will stack.



*Message stack*

In case of an error, the corresponding card that triggered the operation will be highlighted in red. The

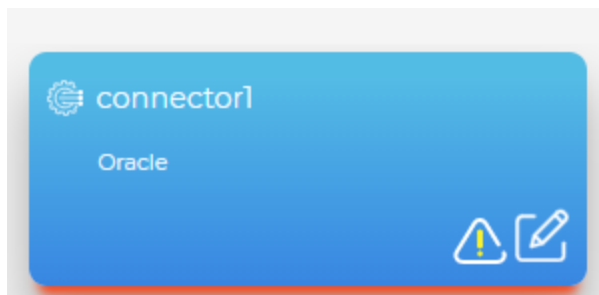
**Error Button**  will also appear on the card .

To view the error message again, click on the error button. The message will appear again on the bottom right hand corner of the screen.

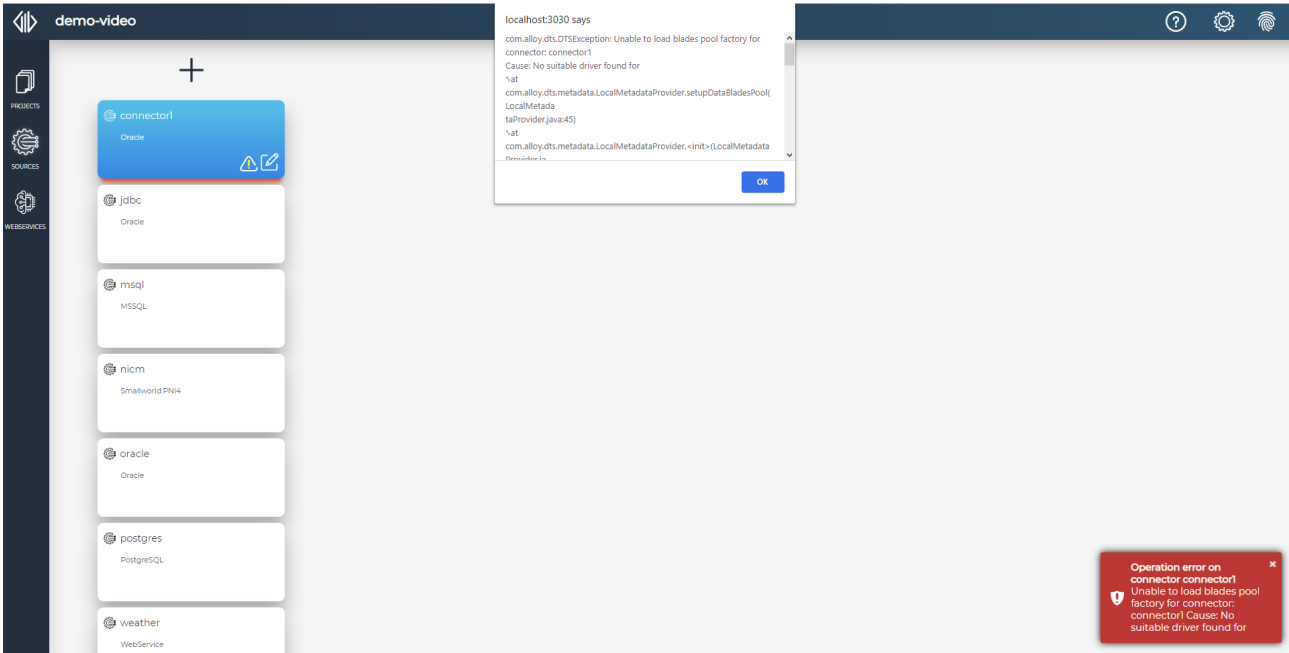
To view more specific details about an error , simply click on the error message on the bottom right hand corner of the screen.

An alert message will appear at the top of the screen containing detailed information.

To close the alert, click OK.



*Card with error*



Error details alert



# Connectors

Details about all the datasource connectors DTS provides

## 4 Connectors

DTS connectors are software modules that implement the [DTS communications API](#)<sup>[256]</sup> in order to provide access to data and functionality from a particular data source.

Connectors are generally stand-alone modules designed to run inside a container (e.g. Docker), connect to the [DTS Controller](#)<sup>[210]</sup> and respond to requests in accordance to the DTS communications API.

The types of requests connectors can respond to depend on the mode in which they are running. This can be **Metadata Mode** for responding to requests to *describe* data structures and functionality endpoints or **Data Mode** for responding to data *streaming* and remote *execution* requests.

This section explores the particularities of each pre-built data source connector DTS provides.

Certain terms are used in the overview of each connector:

- **Development Platform:** they fundamental runtime environment of the connector;
- **Deployment Paradigm:** the way each connector is encapsulated for deployment and/or instantiated;
- **Metadata Mode Paradigm:** how each connector is instantiated for use in Metadata Mode: **Local** (within the GUI Controller container) or **Remote** (in a separate container);
- **DTS\_PRODUCER\_CATEGORY:** the exact name by which DTS knows the connector type category.

Currently available pre-built connectors are:





- [MS SQL Server](#)<sup>[127]</sup>
- [Oracle](#)<sup>[139]</sup>
- [PostgreSQL](#)<sup>[146]</sup>
- [SAP Hana](#)<sup>[153]</sup>
- [MySQL](#)<sup>[133]</sup>
- [Smallworld](#)<sup>[159]</sup>
- [Web Service](#)<sup>[181]</sup>

### 4.1 Apache Kafka

---

The Apache Kafka Connector allows DTS and its clients to interact with Topics in Apache Kafka.

Development Platform	Java 8
Deployment Paradigm	Docker Container
Metadata Mode Paradigm	Local
DTS_PRODUCER_CATEGORY	Kafka

- [Connection Parameters](#)  119
- [Types](#)  119
- [Topics](#)  120
- [Limitations](#)  120

#### 4.1.1 Connection Parameters

##### Properties


A list of properties that will be used for all Kafka Consumers and Producers on this connector. Properties need to be in the standard form:

```
<property.name>=<property.value>
```

e.g.:

```
bootstrap.servers=172.16.10.225
```

 **bootstrap.servers** is the only mandatory property since it defines where to find the Kafka instance

 Any properties defined here will be inherited by all Topics in the connector and all Subscriptions to those Topics. A property with the same name set on a Topic or on Subscribe will override the properties set here.

#### 4.1.2 Types

Kafka type mappings are not fully automatic, as the system does not provide the necessary metadata.

Instead, the user needs to decide how the keys and message bodies of Kafka records will be (de)serialized by DTS. This will also reflect in the data types provided to the consumers.

At this time, DTS support two types of (de)serialization for Kafka objects.

Kafka Deserializer	DTS Type
org.apache.kafka.common.serialization.StringDeserializer	STRING
org.apache.kafka.common.serialization.ByteArrayDeserializer	BYTE VECTOR

#### 4.1.3 Topics

The DTS Kafka Connector supports operations on all Topics accessible with the given connection parameters.

##### DTS Naming

**DTS Name** <topic\_name>

**DTS Key** <topic\_name>

##### Keys and Messages

DTS can serialize and deserialize any keys and messages from any topic with the as outlined in [Types](#)<sup>119</sup>.

##### Metadata

The connector is also capable of extracting metadata regarding available Topics:

- Partition IDs

#### 4.1.4 Limitations

- Avro or any other kind of custom structured (de)serialization is not yet supported.
- Static filters on keys and messages are not yet available.

## 4.2 MariaDB

The MariaDB Connector allows DTS and its clients to access data and functionality in MariaDB databases.

<b>Development Platform</b>	Java 8
<b>Deployment Paradigm</b>	Docker Container
<b>Metadata Mode Paradigm</b>	Local
<b>DTS_PRODUCER_CATEGORY</b>	MariaDB

- [Connection Parameters](#) <sup>121</sup>
- [Types](#) <sup>122</sup>
- [Geometry](#) <sup>123</sup>
- [Tables and Views](#) <sup>125</sup>
- [Routine Calls](#) <sup>125</sup>
- [Limitations](#) <sup>126</sup>

### 4.2.1 Connection Parameters

<b>Connection String</b>	The connection string JDBC will use to connect to the DB, of the following form:  <code>jdbc:mariadb://&lt;hostname&gt;:&lt;port&gt;/&lt;db&gt;</code>  e.g.: <code>jdbc:mariadb://172.16.10.231:3307/test</code>
<b>Username</b>	Username for the user through which DTS will connect to the DB.
<b>Password</b>	Password for the user.
<b>Schemas</b>	The names of the schemas/databases DTS should access, separated by commas.

#### Schemas are optional

If the Schemas field is left empty, DTS will access all the databases in the instance that the specified user can.

## 4.2.2 Types

MariaDB Type	DTS Type
bigint	INT64
int8	INT64
int	INT32
integer	INT32
mediumint	INT32
int4	INT32
int3	INT32
smallint	INT16
int2	INT16
tinyint	UNSIGNED_INT8
int1	UNSIGNED_INT8
numeric	DECIMAL
decimal	DECIMAL
dec x	DECIMAL
fixed	DECIMAL
bit	BOOLEAN
boolean	BOOLEAN
float	FLOAT
real	FLOAT
double	DOUBLE
double precision	DOUBLE
date	DATE
year	DATE
datetime	DATE_TIME
timestamp	DATE_TIME
time	TIME
char	STRING
varchar	STRING
enum	STRING
set	STRING
uuid	STRING
inet4	STRING
inet6	STRING

binary	BINARY
varbinary	BINARY
char byte	BINARY
geometry	DTS_GEOMETRY
point	DTS_POINT_GEOMETRY
linestring	DTS_LINE_GEOMETRY
polygon	DTS_AREA_GEOMETRY
multipoint	DTS_MULTIPPOINT_GEOMETRY
multilinestring	DTS_MULTILINE_GEOMETRY
multipolygon	DTS_MULTIAREA_GEOMETRY
geometrycollection	DTS_GEOMETRY

**Currently Unsupported Native Types:**

tinyblob, blob, mediumblob, longblob, tinytext, text, mediumtext, longtext, json, long, long varchar

**See also**

[DTS Types](#) <sup>[218]</sup>

### 4.2.3 Geometry

MariaDB uses the ST Geometry format with following distinct types:

- geometry
- point
- linestring
- polygon
- multipoint
- multilinestring
- multipolygon
- geometrycollection

All types are functionally equivalent, the only distinction is what ST subtype each allows. While *geometry* allows any ST subtype to be stored, the others only allow the homonym subtype (i.e. only LINESTRING geometries are allowed in a *linestring* column or variable).

## Geometry Types

Here is how MariaDB (ST) geometry subtypes map to GeoJson geometry types.

ST Subtype / MariaDB Type	Interpretation	GeoJson Geometry Type
POINT / point or geometry	Single point	Point
MULTIPOINT / multipoint or geometry	Multiple points	MultiPoint
LINESTRING / linestring or geometry	Sequence of straight lines without gaps	LineString
CIRCULARSTRING / geometry	Sequence of circle arcs without gaps	LineString + DTS corrections
MULTILINESTRING / multilinestring or geometry	Multiple LINESTRINGs (not necessarily connected)	MultiLineString
COMPOUNDCURVE / geometry	Sequence of connected LINESTRINGs and CIRCULARLINESTRINGs	MultiLineString + DTS corrections
POLYGON / polygon or geometry	Polygon (with or without holes)	Polygon
CURVEPOLYGON / geometry	Polygon whose boundaries can contain circle arcs	Polygon + DTS corrections
MULTIPOLYGON / multipolygon or geometry	Multiple POLYGONs	MultiPolygon (+ DTS corrections)
GEOMETRYCOLLECTION / geometrycollection or geometry	Collection of any of the above	GeometryCollection (+ DTS corrections)

Z and M modifiers (e.g. POINT Z) are supported for all geometry types (except GEOMETRYCOLLECTION where they are not applicable), but the M values are always ignored and processing geometries with Z values has certain [limitations](#) <sup>126</sup>.

**i Further reading:**

- [MariaDB Geographic and Geometric Features](#)
- [DTS Geometry Overview](#) <sup>222</sup>

#### 4.2.4 Tables and Views

The DTS MariaDB Connector fully supports all tables and views in MariaDB and does not differentiate between the two.

##### DTS Naming

**DTS Name**                    <table\_or\_view\_name>

**DTS Key**                    <schema\_name>.<table\_or\_view\_name>

##### Metadata

The connector is also capable of extracting metadata regarding the table/view columns:

- DATA\_TYPE
- CHARACTER\_MAXIMUM\_LENGTH
- NUMERIC\_PRECISION
- DATETIME\_PRECISION
- NUMERIC\_SCALE
- IS\_NULLABLE
- ORDINAL\_POSITION
- CONSTRAINT -> PRIMARY\_KEY

Standard functionality only makes use of DATA\_TYPE, ORDINAL\_POSITION and PRIMARY\_KEY, but the other items are available for extension in DTS descriptor objects.

#### 4.2.5 Routine Calls

The DTS MariaDB Connector supports remote calls for any MariaDB stored procedures and functions.

##### DTS Naming

<b>DTS Name</b>	<proc_or_func_name>
<b>DTS Key</b>	<schema_name>.<proc_or_func_name>

## Arguments and Results

Functions and procedures with any type of arguments and results are supported, as long as the types themselves are supported. Please see the [list of supported types](#)<sup>[122]</sup> for more information.

## Metadata

The connector is also capable of extracting metadata regarding the arguments and results of a procedure or function:

- DATA\_TYPE
- CHARACTER\_MAXIMUM\_LENGTH
- NUMERIC\_PRECISION
- DATETIME\_PRECISION
- NUMERIC\_SCALE
- ORDINAL\_POSITION
- PARAMETER\_MODE

Standard functionality only makes use of DATA\_TYPE, ORDINAL\_POSITION and PARAMETER\_MODE, but the other items are available for extension in DTS descriptor objects.

### 4.2.6 Limitations

- DB Synonyms are not currently supported.
- The following native data types are not currently supported:
  - **blob, tinyblob, mediumblob, longblob**
  - **text, tinytext, mediumtext, longtext**
  - **long, long varchar**
  - **json**
- M (measure) values on "POINT M" or "POINT ZM" geometries are ignored by the connector and are not passed through.

- General DTS limitations regarding Z-valued geometries apply. Please see [Known Limitations](#)<sup>127</sup> for more details.

## 4.3 MS SQL Server

The MS SQL Server Connector allows DTS and its clients to access data and functionality in Microsoft SQL Server databases.

<b>Development Platform</b>	Java 8
<b>Deployment Paradigm</b>	Docker Container
<b>Metadata Mode Paradigm</b>	Local
<b>DTS_PRODUCER_CATEGORY</b>	MSSQL

- [Connection Parameters](#)<sup>127</sup>
- [Types](#)<sup>128</sup>
- [Geometry](#)<sup>129</sup>
- [Tables and Views](#)<sup>131</sup>
- [Function/Procedure Calls](#)<sup>131</sup>
- [Limitations](#)<sup>133</sup>

### 4.3.1 Connection Parameters

<b>Connection String</b>	<p>The connection string JDBC will use to connect to the DB, of the following form:</p> <pre>jdbc:sqlserver://&lt;hostname&gt;:&lt;port&gt;;databaseName=&lt;d b_name&gt;</pre> <p>e.g.:</p> <pre>jdbc:sqlserver://172.16.10.231:1433;databaseName=dt</pre>
<b>Username</b>	Username for the user through which DTS will connect to the DB.

**Password**

Password for the user.

**Schemas**

The names of the schemas DTS should access, separated by commas.

**Schemas are optional**

If the Schemas field is left empty, DTS will access all the schemas in the database that the specified user can.

**4.3.2 Types**

MSSQL Type	DTS Type
bigint	INT64
int	INT32
smallint	INT16
tinyint	UNSIGNED_INT8
numeric	DECIMAL
decimal	DECIMAL
money	DECIMAL
smallmoney	DECIMAL
bit	BOOLEAN
float	DOUBLE
real	DOUBLE
date	DATE
datetime	DATE_TIME
datetime2	DATE_TIME
smalldatetime	DATE_TIME
datetimeoffset	DATE_TIME
time	TIME
char	STRING
nchar	STRING
varchar	STRING
nvarchar	STRING
text	STRING
ntext	STRING
binary	BINARY
varbinary	BINARY
image	BINARY

uniqueidentifier	STRING
xml	STRING
geometry	DTS_GEOMETRY
geography	DTS_GEOGRAPHY

**User Types** are fully supported and mapped as their underlying native type.

**Table Types** are also fully supported and mapped as **DTS objects** which reflect the table type's column structure. When they are used as arguments, they take the form of arrays with the respective row element type.



**Currently Unsupported Native Types:**

cursor, rowversion, hierarchyid, sqlvariant



**See also**

[DTS Types](#) <sup>218</sup>

### 4.3.3 Geometry

There are two categories of objects that are interpreted as geometries in MS SQL Server:

Geometry	Geography
Compliant with the ST_GEOMETRY standard	Compliant with the ST_GEOMETRY standard
Accessible as WKT	Accessible as WKT
Strictly euclidean	Strictly geodetic
Coordinates represent (x, y) or (x, y, z)	Coordinates represent (lon, lat) or (lon, lat, elv)
If specified, coordinate system must be in length units	If specified, coordinate system must be in degrees

The DTS MS SQL Connector transforms both categories to [DTSGeoJson](#) <sup>222</sup> for transfer, but keeps track of the them as different [internal types](#) <sup>218</sup> (DTS\_GEOMETRY, DTS\_GEOGRAPHY) in order to correctly interact with MS SQL Server and other data sources which similarly differentiate spatial data.

## Geometry Types

Here is how MS SQL geometry and geography subtypes map to GeoJson geometry types.

ST Subtype	Interpretation	GeoJson Geometry Type
POINT	Single point	Point
MULTIPOINT	Multiple points	MultiPoint
LINESTRING	Sequence of straight lines without gaps	LineString
CIRCULARSTRING	Sequence of circle arcs without gaps	LineString + DTS corrections
MULTILINESTRING	Multiple LINESTRINGs (not necessarily connected)	MultiLineString
COMPOUNDCURVE	Sequence of connected LINESTRINGs and CIRCULARLINESTRINGs	MultiLineString + DTS corrections
POLYGON	Polygon (with or without holes)	Polygon
CURVEPOLYGON	Polygon whose boundaries can contain circle arcs	Polygon + DTS corrections
MULTIPOLYGON	Multiple POLYGONs	MultiPolygon (+ DTS corrections)
GEOMETRYCOLLECTION	Collection of any of the above	GeometryCollection (+ DTS corrections)

Z and M modifiers (e.g. POINT Z) are supported for all geometry types (except GEOMETRYCOLLECTION where they are not applicable), but the M values are always ignored and processing geometries with Z values has certain [limitations](#)<sup>133</sup>.

### Further reading:

- [MSSQL Spatial Data Types Overview](#)
- [DTS Geometry Overview](#)<sup>222</sup>

#### 4.3.4 Tables and Views

The DTS MS SQL Server Connector fully supports all tables and views in MS SQL Server and does not differentiate between the two.

##### DTS Naming

**DTS Name** <table\_or\_view\_name>


**DTS Key** <schema\_name>.<table\_or\_view\_name>

##### Metadata

The connector is also capable of extracting metadata regarding the table/view columns:

- DATA\_TYPE
- CHARACTER\_OCTET\_LENGTH
- CHARACTER\_MAXIMUM\_LENGTH
- NUMERIC\_PRECISION
- DATETIME\_PRECISION
- NUMERIC\_SCALE
- IS\_NULLABLE
- ORDINAL\_POSITION
- CONSTRAINT -> PRIMARY\_KEY

Standard functionality only makes use of DATA\_TYPE, ORDINAL\_POSITION and PRIMARY\_KEY, but the other items are available for extension in DTS descriptor objects.

 Structured types (table types) are not supported as column data types by MS SQL Server, only native and user types are. Please see [Types](#)<sup>[128]</sup> for information of how the connector handles these.



#### 4.3.5 Routine Calls

The DTS MS SQL Connector supports remote calls for any MS SQL Server stored procedures and functions, scalar-valued as well as table-valued.

## DTS Naming

<b>DTS Name</b>	<proc_or_func_name>
<b>DTS Key</b>	<schema_name>.<proc_or_func_name>

## Arguments and Results

- **Table type arguments** to functions and procedures, as well as results of table-valued functions are fully supported for remote calling
    - The table arguments are always passed through DTS as arrays of custom objects, each object representing a row in the input table
    - The result of a table-valued function is always a DTS stream
-  See [Streams](#)  220
- MS SQL does not have a way of representing a **single custom object**, other than a table with a single row, so that extends to DTS, where they will become arrays with a single element.
  - In MS SQL Server, **procedure output arguments** always serve as inputs as well. As such, they will appear both in the list of arguments and the list of results for a given remote call and be treated as two distinct items, one going in, one coming out.

For argument and result native and user type mappings, please see [Types](#)  128.

## Metadata

The connector is also capable of extracting metadata regarding the arguments and results of a procedure or function:

- DATA\_TYPE
- CHARACTER\_OCTET\_LENGTH
- CHARACTER\_MAXIMUM\_LENGTH
- NUMERIC\_PRECISION
- DATETIME\_PRECISION
- NUMERIC\_SCALE
- ORDINAL\_POSITION
- PARAMETER\_MODE

- `USER_DEFINED_TYPE`

Standard functionality only makes use of `DATA_TYPE`, `ORDINAL_POSITION`, `PARAMETER_MODE` and `USER_DEFINED_TYPE`, but the other items are available for extension in DTS descriptor objects.

#### 4.3.6 Limitations

- DB Synonyms are not currently supported.
- The following native data types are not currently supported:
  - `cursor`
  - `rowversion`
  - `hierarchyid`
  - `sqlvariant`
- Any user types that extend these native types are also not supported.
- Any table types that contain columns which use these native types are not supported.
- M (measure) values on "POINT M" or "POINT ZM" geometries are ignored by the connector and are not passed through.
- General DTS limitations regarding Z-valued geometries apply. Please see [Known Limitations](#)<sup>322</sup> for more details.

## 4.4 MySQL

The MySQL Connector allows DTS and its clients to access data and functionality in MySQL databases.

<b>Development Platform</b>	Java 8
<b>Deployment Paradigm</b>	Docker Container
<b>Metadata Mode Paradigm</b>	Local
<b>DTS_PRODUCER_CATEGORY</b>	MySQL

- [Connection Parameters](#)<sup>134</sup>

- [Types](#) 134
- [Geometry](#) 135
- [Tables and Views](#) 137
- [Routine Calls](#) 138
- [Limitations](#) 139

#### 4.4.1 Connection Parameters

##### Connection String

The connection string JDBC will use to connect to the DB, of the following form:

```
jdbc:mysql://<hostname>:<port>
```

e.g.:

```
jdbc:mysql://172.16.10.231:3306
```

##### Username

Username for the user through which DTS will connect to the DB.

##### Password

Password for the user.

##### Schemas

The names of the schemas/databases DTS should access, separated by commas.



##### Schemas are optional

If the Schemas field is left empty, DTS will access all the databases in the instance that the specified user can.

#### 4.4.2 Types

MySQL Type	DTS Type
bigint	INT64
int	INT32
integer	INT32
mediumint	INT32
smallint	INT16
tinyint	UNSIGNED_INT8
numeric	DECIMAL
decimal	DECIMAL
dec	DECIMAL

fixed	DECIMAL
bit	BOOLEAN
float	FLOAT
real	FLOAT
double	DOUBLE
double precision	DOUBLE
date	DATE
year	DATE
datetime	DATE_TIME
timestamp	DATE_TIME
time	TIME
char	STRING
varchar	STRING
enum	STRING
set	STRING
binary	BINARY
varbinary	BINARY
geometry	DTS_GEOMETRY
point	DTS_POINT_GEOMETRY
linestring	DTS_LINE_GEOMETRY
polygon	DTS_AREA_GEOMETRY
multipoint	DTS_MULTIPPOINT_GEOMETRY
multilinestring	DTS_MULTILINE_GEOMETRY
multipolygon	DTS_MULTIAREA_GEOMETRY
geometrycollection	DTS_GEOMETRY



#### Currently Unsupported Native Types:

tinyblob, blob, mediumblob, longblob, tinytext, text, mediumtext, longtext, json



#### See also

[DTS Types](#) <sup>218</sup>

### 4.4.3 Geometry

MySQL uses the ST Geometry format with following distinct types:

- geometry
- point

- linestring
- polygon
- multipoint
- multilinestring
- multipolygon
- geometrycollection

All types are functionally equivalent, the only distinction is what ST subtype each allows. While *geometry* allows any ST subtype to be stored, the others only allow the homonym subtype (i.e. only LINESTRING geometries are allowed in a *linestring* column or variable).

## Geometry Types

Here is how MySQL (ST) geometry subtypes map to GeoJson geometry types.

ST Subtype / MySQL Type	Interpretation	GeoJson Geometry Type
POINT / point or geometry	Single point	Point
MULTIPOINT / multipoint or geometry	Multiple points	MultiPoint
LINESTRING / linestring or geometry	Sequence of straight lines without gaps	LineString
CIRCULARSTRING / geometry	Sequence of circle arcs without gaps	LineString + DTS corrections
MULTILINESTRING / multilinestring or geometry	Multiple LINESTRINGs (not necessarily connected)	MultiLineString
COMPOUNDCURVE / geometry	Sequence of connected LINESTRINGs and CIRCULARLINESTRINGs	MultiLineString + DTS corrections
POLYGON / polygon or geometry	Polygon (with or without holes)	Polygon
CURVEPOLYGON / geometry	Polygon whose boundaries can contain circle arcs	Polygon + DTS corrections
MULTIPOLYGON / multipolygon or geometry	Multiple POLYGONs	MultiPolygon (+ DTS corrections)

GEOMETRYCOLLECTION / geometrycollection or geometry	Collection of any of the above	GeometryCollection (+ DTS corrections)
---	--------------------------------	--

Z and M modifiers (e.g. POINT Z) are supported for all geometry types (except GEOMETRYCOLLECTION where they are not applicable), but the M values are always ignored and processing geometries with Z values has certain [limitations](#) <sup>139</sup>.

**i Further reading:**

- [MySQL Spatial Reference](#)
- [DTS Geometry Overview](#) <sup>222</sup>

#### 4.4.4 Tables and Views

The DTS MySQL Connector fully supports all tables and views in MySQL and does not differentiate between the two.

### DTS Naming

**DTS Name** <table\_or\_view\_name>

**DTS Key** <schema\_name>.<table\_or\_view\_name>

### Metadata

The connector is also capable of extracting metadata regarding the table/view columns:

- DATA\_TYPE
- CHARACTER\_MAXIMUM\_LENGTH
- NUMERIC\_PRECISION
- DATETIME\_PRECISION
- NUMERIC\_SCALE
- IS\_NULLABLE
- ORDINAL\_POSITION
- CONSTRAINT -> PRIMARY\_KEY

Standard functionality only makes use of DATA\_TYPE, ORDINAL\_POSITION and PRIMARY\_KEY, but the other items are available for extension in DTS descriptor objects.

#### 4.4.5 Routine Calls

The DTS MySQL Connector supports remote calls for any MySQL stored procedures and functions.

##### DTS Naming

**DTS Name** <proc\_or\_func\_name>

**DTS Key** <schema\_name>.<proc\_or\_func\_name>

##### Arguments and Results

Functions and procedures with any type of arguments and results are supported, as long as the types themselves are supported. Please see the [list of supported types](#)<sup>134</sup> for more information.

##### Metadata

The connector is also capable of extracting metadata regarding the arguments and results of a procedure or function:

- DATA\_TYPE
- CHARACTER\_MAXIMUM\_LENGTH
- NUMERIC\_PRECISION
- DATETIME\_PRECISION
- NUMERIC\_SCALE
- ORDINAL\_POSITION
- PARAMETER\_MODE

Standard functionality only makes use of DATA\_TYPE, ORDINAL\_POSITION and PARAMETER\_MODE, but the other items are available for extension in DTS descriptor objects.

#### 4.4.6 Limitations

- DB Synonyms are not currently supported.
- The following native data types are not currently supported:
  - **blob, tinyblob, mediumblob, longblob**
  - **text, tinytext, mediumtext, longtext**
  - **json**
- M (measure) values on "POINT M" or "POINT ZM" geometries are ignored by the connector and are not passed through.
- General DTS limitations regarding Z-valued geometries apply. Please see [Known Limitations](#)<sup>322</sup> for more details.

## 4.5 Oracle

---

The Oracle Connector allows DTS and its clients to access data and functionality in Oracle Databases.

<b>Development Platform</b>	Java 8
<b>Deployment Paradigm</b>	Docker Container
<b>Metadata Mode Paradigm</b>	Local
<b>DTS_PRODUCER_CATEGORY</b>	Oracle

- [Connection Parameters](#)<sup>140</sup>
- [Types](#)<sup>140</sup>
- [Geometry](#)<sup>141</sup>
- [Tables and Views](#)<sup>143</sup>
- [Function/Procedure Calls](#)<sup>144</sup>
- [Limitations](#)<sup>145</sup>

### 4.5.1 Connection Parameters

**Connection String**

The connection string JDBC will use to connect to the DB, of the following form:

```
jdbc:oracle:thin:@<hostname>:<port>:<sid>
```

e.g.:

```
jdbc:oracle:thin:@172.16.10.212:1521:orcl
```

**Username**

Username for the user through which DTS will connect to the DB.

**Password**

Password for the user.

**Schemas**

The names of the schemas DTS should access, separated by commas.

**Schemas are optional**

If the Schemas field is left empty, DTS will only access the given user's own schema.

### 4.5.2 Types

Oracle Type	DTS Type
BOOLEAN	BOOLEAN
CHAR	STRING
NCHAR	STRING
VARCHAR	STRING
VARCHAR2	STRING
NVARCHAR2	STRING
ROWID	STRING
UROWID	STRING
MLSLABEL	STRING
RAW	BINARY
INTEGER	INT32
INT	INT32
SMALLINT	INT32
NUMBER	DECIMAL
NUMERIC	DECIMAL
DECIMAL	DECIMAL

FLOAT	DOUBLE
REAL	FLOAT
DOUBLE PRECISION	DOUBLE
DATE	DATE
TIMESTAMP	DATE_TIME
TIMESTAMP WITH TIMEZONE	DATE_TIME
TIMESTAMP WITH LOCAL TIMEZONE	DATE_TIME
INTERVAL YEAR TO MONTH	STRING
INTERVAL DAY TO SECOND	STRING
TIME	TIME
MDSYS.SDO_GEOMETRY	DTS_GEOMETRY

The DTS Oracle Connector also supports the following categories of User-Defined Types:

- **Object Types:** any Oracle object type is supported as long as its individual field types are also supported;
- **Collection Types:** VARRAYs of native and Object Types are fully supported.
- **Type Inheritance** is fully supported;
- **Aliases** are fully supported for type naming;

 **Currently Unsupported Native Types:**  
BLOB, CLOB, NCLOB, LONG RAW, BFILE, CFILE, XMLTYPE

 **See also**  
[DTS Types](#) <sup>[218]</sup>

### 4.5.3 Geometry

The DTS Oracle Connector can exchange geometry data with an Oracle Database in the standard Oracle Spatial **MDSYS.SDO\_GEOMETRY** format.

The connector transforms SDO\_GEOMETRY objects to and from [DTSGeoJson](#) <sup>[222]</sup> for propagation through DTS and maps such columns, fields and arguments with the [internal type](#) <sup>[218]</sup> of DTS\_GEOMETRY.

## Geometry Types

SDO\_GEOMETRY objects come in a few different types, codified by the GTYPE field. Here is how they are mapped by the connector:

SDO GTYPE	Interpretation	GeoJson Geometry Type
DL01	Single point	Point
DL02	Sequence of lines (straight or curved) without gaps	LineString (+ DTS corrections)
DL03	Polygon (with or without holes) (can have curved edges)	Polygon (+ DTS corrections)
DL04	Collection of various geometries	GeometryCollection (+ DTS corrections)
DL05	Multiple points	MultiPoint
DL06	Multiple lines, not necessarily connected	MultiLineString (+ DTS corrections)
DL07	Multiple polygons	MultiPolygon (+ DTS corrections)

 Solid (DL08) and multisolid (DL09) geometries are not supported at this time

 The D and L values in the GTYPE represent the Dimensionality and the Linear referencing measure respectively.

The connector accepts D values of 2 and 3 (with certain [limitations](#)<sup>[145]</sup>) and ignores the L values.

## Sector Types

Corrections will be added to geometries which need to be approximated for standard GeoJson representation. These are the geometries which contain sectors that are not sequences of straight lines.

The SDO\_GEOMETRY format encodes the various sectors that form a geometry in the SDO\_ELEM\_INFO field as triplets of the form **[startIndex, elementType, interpretation]**. Here is how these translate to DTS correction line types:

SDO_ELEM_INFO	Interpretation	Correction Line Type
i, 1, 1	Point coordinate	none
i, 1, 0	Point orientation	none*
i, 1, n>1	Point cluster with n points	none
i, 2, 1	Straight line string	none (LINE_STRING**)

i, 2, 2	Circular arc string	CP_ARCS
i, 2, 3	NURBS	NURBS
i, 1003/2003, 1	Polygon outer/inner boundary made of straight lines	none (LINE_STRING**)
i, 1003/2003, 2	Polygon outer/inner boundary made of circular arcs	CP_ARCS
i, 1003/2003, 3	Polygon outer/inner boundary which is a perfect rectangle	RECTANGLE
i, 1003/2003, 4	Polygon outer/inner boundary which is a perfect circle	CIRCLE
i, 4, n>1	Compound line string made of n sectors (can contain straight lines, arcs and NURBS)	combination of CP_ARCS, NURBS (and LINE_STRING**)
i, 1005/2005, n>1	Compound polygon made of n sectors (can contain straight lines, arcs and NURBS)	combination of CP_ARCS, NURBS (and LINE_STRING**)

 Surface and Solid elements (ETYPE=1006/2006, 1007) are not supported at this time

 \* Point orientation has its own field in DTSGeoJson and does not need a correction to be stored

 \*\* LINE\_STRING corrections are not produced by this connector or any of the other standard connectors. It exists for extensions and optimization

 Besides the i, 1, 1 + ordinates representation, SDO\_GEOMETRY can also store a single point in the bespoke field SDO\_POINT. This form is also supported by the connector.

 Further reading:

- [Oracle Spatial Data Types Overview](#)
- [DTS Geometry Overview](#) <sup>222</sup>

#### 4.5.4 Tables and Views

The DTS Oracle Connector fully supports all tables and views in the Oracle database and does not differentiate between the two.

## DTS Naming

<b>DTS Name</b>	<table_or_view_name>
<b>DTS Key</b>	<schema_name>.<table_or_view_name>

 **DB Synonyms are fully supported**

## Metadata

The connector is also capable of extracting metadata regarding the table/view columns:

- DATA\_TYPE
- DATA\_TYPE\_OWNER
- DATA\_LENGTH
- DATA\_PRECISION
- DATA\_SCALE
- NULLABLE
- CHAR\_LENGTH
- CONSTRAINT -> PRIMARY\_KEY
- CONSTRAINT -> FOREIGN\_KEY

Standard functionality only makes use of DATA\_TYPE, DATA\_TYPE\_OWNER and PRIMARY\_KEY, but the other items are available for extension in DTS descriptor objects.

### 4.5.5 Routine Calls

The DTS Oracle Connector supports remote calls for any Oracle stored procedures and functions.

## DTS Naming

<b>DTS Name</b>	<proc_or_func_name>
<b>DTS Key</b>	<schema_name>.<proc_or_func_name>

## Arguments and Results

- Any of the supported [data types](#) <sup>140</sup> are also supported as procedure/function argument and result types, including the user-defined types.
- **IN OUT parameters** will appear in both the arguments and the results list for a given remote call and will be treated by the connector as two distinct items, one going in, one coming out.
- **Functions returning tables** will have their results wrapped and operated as DTS streams. The same is true for **procedures** with a single OUT argument which is a table.

## Metadata

The connector is also capable of extracting metadata regarding the arguments and results of a procedure or function:

- TYPE\_NAME
- TYPE\_OWNER
- DATA\_TYPE
- DATA\_PRECISION
- DATA\_SCALE
- DATA\_LENGTH
- DATA\_LEVEL
- POSITION
- DEFAULT\_VALUE
- SEQUENCE
- IN\_OUT

Standard functionality only makes use of DATA\_TYPE, TYPE\_NAME, TYPE\_OWNER, POSITION and IN\_OUT, but the other items are available for extension in DTS descriptor objects.

### 4.5.6 Limitations

- The following native data types are not currently supported:

- BLOB
  - CLOB
  - NCLOB
  - LONG RAW
  - BFILE
  - CFILE
  - XMLTYPE
- Any user types that extend or are composed of these native types are also not supported.
  - Solid and multi solid geometries, as well as any surface and solid elements in other types of geometries are not currently supported.
  - Linear Referencing Measure values in geometry GTYPEs are ignored and not passed through the connector.
  - General DTS limitations regarding Z-valued geometries apply. Please see [Known Limitations](#)<sup>322</sup> for more details.

## 4.6 PostgreSQL

---

The PostgreSQL Connector allows DTS and its clients to access data and functionality in PostgreSQL databases.

<b>Development Platform</b>	Java 8
<b>Deployment Paradigm</b>	Docker Container
<b>Metadata Mode Paradigm</b>	Local
<b>DTS_PRODUCER_CATEGORY</b>	PostgreSQL

- [Connection Parameters](#)<sup>147</sup>
- [Types](#)<sup>147</sup>
- [Geometry](#)<sup>149</sup>
- [Tables and Views](#)<sup>151</sup>
- [Function/Procedure Calls](#)<sup>152</sup>
- [Limitations](#)<sup>153</sup>

#### 4.6.1 Connection Parameters

**Connection String**

The connection string JDBC will use to connect to the DB, of the following form:

```
jdbc:postgresql://<hostname>:<port>/<db_name>
```

e.g.:

```
jdbc:postgresql://172.16.10.242:5432/test_db
```

**Username**

Username for the user through which DTS will connect to the DB.

**Password**

Password for the user.

**Schemas**

The names of the schemas DTS should access, separated by commas.

**Port is optional**

The <port> can be skipped in the connection string if using the default port (5432).

**Schemas are optional**

If the Schemas field is left empty, DTS will access all the schemas in the database that the specified user can.

#### 4.6.2 Types

PostgreSQL Type	DTS Type
boolean	BOOLEAN
character	STRING
char	STRING
varchar	STRING
character varying	STRING
text	STRING
smallint	INT16
smallserial	INT16
int	INT32
serial	INT32
bigint	INT64
bigserial	INT64
bytea	BYTE_VECTOR

real	FLOAT
double precision	DOUBLE
numeric	DECIMAL
decimal	DECIMAL
date	DATE
time	TIME
time with timezone	TIME
time without timezone	TIME
timestamp	DATE_TIME
timestamp with timezone	DATE_TIME
timestamp without timezone	DATE_TIME
interval	STRING
uuid	STRING
cidr	STRING
inet	STRING
macaddr	STRING
macaddr8	STRING
bit	STRING
bit varying	STRING
tsvector	STRING
tsquery	STRING
xml	STRING
json	STRING
jsonb	STRING
money	STRING
geometry	DTS_GEOMETRY
box	DTS_GEOMETRY
box2d	DTS_GEOMETRY
box2df	DTS_GEOMETRY
box3d	DTS_GEOMETRY
circle	DTS_GEOMETRY
line	DTS_GEOMETRY
lseg	DTS_GEOMETRY
path	DTS_GEOMETRY
point	DTS_GEOMETRY
polygon	DTS_GEOMETRY

The DTS PostgreSQL Connector also supports all **Composite (User-Defined) Types** that contain fields of supported types as well as all array [ ] types.

 Enumeration, External, Range and Shell types are not currently supported


### 4.6.3 Geometry

PostgreSQL can include two separate geometry sets: native (simple) geometries and PostGIS (ST) geometries.

DTS maps both sets to [DTSGeoJson](#)<sup>[222]</sup>.

## Native Geometry Types

PostgreSQL Geometry Type	Interpretation	GeoJson Geometry Type
point	Single point	Point
lseg	Line segment	LineString
path	Sequence of connected line segments - can be open [] or closed ()	LineString or Polygon
box	Rectangle	Polygon + DTS Correction
box2d	Rectangle	Polygon + DTS Correction
polygon	Polygon	Polygon
circle	Circle	Polygon + DTS Correction

 The infinite "line" type is not supported

 The 3-dimensional "box3d" type is not supported

 Native geometry types can only be used as function inputs when they are direct arguments. Please see [Limitations](#)<sup>[153]</sup> for details.

## PostGIS Geometry Types

There are two categories of objects that are interpreted as geometries in PostgreSQL:

Geometry	Geography
Compliant with the ST_GEOMETRY standard	Compliant with the ST_GEOMETRY standard
Accessible as WKB	Accessible as WKB
Strictly euclidean	Strictly geodetic
Coordinates represent (x, y) or (x, y, z)	Coordinates represent (lon, lat) or (lon, lat, elv)
If specified, coordinate system must be in length units	If specified, coordinate system must be in degrees

The DTS PostgreSQL Connector transforms both categories to [DTSGeoJson](#)<sup>[222]</sup> for transfer, but keeps track of the them as different [internal types](#)<sup>[218]</sup> (DTS\_GEOMETRY, DTS\_GEOGRAPHY) in order to correctly interact with PostgreSQL and other data sources which similarly differentiate spatial data.

Here is how PostGIS geometry and geography types map to GeoJson geometry types.

ST Subtype	Interpretation	GeoJson Geometry Type
POINT	Single point	Point
MULTIPOINT	Multiple points	MultiPoint
LINESTRING	Sequence of straight lines without gaps	LineString
CIRCULARSTRING	Sequence of circle arcs without gaps	LineString + DTS corrections
MULTILINESTRING	Multiple LINESTRINGs (not necessarily connected)	MultiLineString
COMPOUNDCURVE	Sequence of connected LINESTRINGs and CIRCULARLINESTRINGs	MultiLineString + DTS corrections
POLYGON	Polygon (with or without holes)	Polygon
CURVEPOLYGON	Polygon whose boundaries can contain circle arcs	Polygon + DTS corrections
MULTIPOLYGON	Multiple POLYGONs	MultiPolygon (+ DTS corrections)

⚠ Only the 2-dimensional variants of these geometries are supported at present.

⚠ GEOMETRYCOLLECTIONs are not currently supported.

**i** Further reading:

- [PostGIS Spatial Data Types Overview](#)
- [DTS Geometry Overview](#) <sup>222</sup>

#### 4.6.4 Tables and Views

The DTS PostgreSQL Connector fully supports all tables and views in the PostgreSQL database and does not differentiate between the two.

#### DTS Naming

**DTS Name**                    <table\_or\_view\_name>

**DTS Key**                    <schema\_name>.<table\_or\_view\_name>

#### Metadata

The connector is also capable of extracting metadata regarding the table/view columns:

- DATA\_TYPE
- UDT\_NAME
- NUMERIC\_PRECISION
- NUMERIC\_SCALE
- IS\_NULLABLE
- COLUMN\_DEFAULT
- CHAR\_LENGTH
- CHARACTER\_OCTET\_LENGTH
- CONSTRAINT -> PRIMARY KEY
- CONSTRAINT -> FOREIGN\_KEY

Standard functionality only makes use of DATA\_TYPE, UDT\_NAME and PRIMARY\_KEY, but the other items are available for extension in DTS descriptor objects.

#### 4.6.5 Routine Calls

The DTS PostgreSQL Connector supports remote calls for any PostgreSQL stored functions.

 **Stored Procedures are not currently supported. Please see [Limitations](#)<sup>153</sup> for details and workarounds.**

#### DTS Naming

**DTS Name** <func\_name>

**DTS Key** <schema\_name>.<func\_name>

#### Arguments and Results

- Any of the supported [data types](#)<sup>147</sup> are also supported as procedure/function argument and result types, including the user-defined types.
- **Functions returning tables** will be have their results wrapped and operated as DTS streams.

 **Native geometry types can only be used as function inputs when they are direct arguments. Please see [Limitations](#)<sup>153</sup> for details.**

#### Metadata

The connector is also capable of extracting metadata regarding the arguments and results of a function:

- DATA\_TYPE
- UDT\_NAME
- DATA\_PRECISION
- DATA\_SCALE
- DATA\_LENGTH
- POSITION
- DEFAULT\_VALUE

- SEQUENCE
- IN\_OUT

Standard functionality only makes use of DATA\_TYPE, UDT\_NAME and POSITION, but the other items are available for extension in DTS descriptor objects.

#### 4.6.6 Limitations







- Direct procedure calls are not currently supported because of limitations in the Postgres JDBC Driver. To invoke Postgres stored procedures via DTS, please wrap them as stored functions.
- Function overloading is not supported - a single version of a function will be available in DTS and it cannot be controlled which.
- Enumeration, External, Range and Shell user-defined types are not currently supported.
- The "line" native geometry type (representing an infinite straight line) is not currently supported.
- The "box3d" native geometry type is not currently supported.
- User-defined types containing Postgres native (non-PostGIS) geometries ("box", "box2d", "box2df", "circle", "lseg", "path", "point", "polygon") within their structure are not supported as function arguments. These types of geometry can be passed to functions as direct arguments, or PostGIS geometries (geometry, geography) can be used instead, as these are supported in any structure. This limitation does not apply to output data (table columns, function outputs).
- Only 2D PostGIS geometries are currently supported.
- The GEOMETRYCOLLECTION PostGIS (ST) geometry type is not currently supported.
- Functions returning cursors are not supported.

## 4.7 SAP Hana

The SAP Hana Connector allows DTS and its clients to access data and functionality in SAP Hana databases.

<b>Development Platform</b>	Java 8
<b>Deployment Paradigm</b>	Docker Container
<b>Metadata Mode Paradigm</b>	Local

## DTS\_PRODUCER\_CATEGORY SAPHana

- [Connection Parameters](#)  154
- [Types](#)  154
- [Geometry](#)  155
- [Tables and Views](#)  156
- [Function/Procedure Calls](#)  157
- [Limitations](#)  158

## 4.7.1 Connection Parameters

**Connection String**

The connection string JDBC will use to connect to the DB, of the following form:

```
jdbc:sap://<hostname>:<port>/?autocommit=false
```

e.g.:

```
jdbc:sap://172.16.10.185:39015/?autocommit=false
```

**Username**

Username for the user through which DTS will connect to the DB.

**Password**

Password for the user.

**Schemas**

The names of the schemas DTS should access, separated by commas.

**Schemas are optional**

If the Schemas field is left empty, DTS will access all the schemas in the database that the specified user can.


## 4.7.2 Types

SAP Hana Type	DTS Type
bigint	INT64
integer	INT32
smallint	INT16
tinyint	INT16
decimal	DECIMAL

dec	DECIMAL
smalldecimal	DECIMAL
float	FLOAT
real	FLOAT
double	DOUBLE
date	DATE
timestamp	DATE_TIME
seconddate	DATE_TIME
time	TIME
char	STRING
nchar	STRING
varchar	STRING
nvarchar	STRING
shorttext	STRING
alphanum	STRING
string	STRING
binary	BINARY
varbinary	BINARY
st_geometry	DTS_GEOMETRY
st_point	DTS_GEOMETRY

**Array Types** are fully supported and mapped as arrays of the respective element's DTS type.


**Table Types** are also fully supported and mapped as **DTS objects** which reflect the table type's column structure.

 **Currently Unsupported Native Types:**  
blob, clob, nclob, text, bintext

 **See also**  
[DTS Types](#) <sup>218</sup>

### 4.7.3 Geometry

SAPHana uses the ST Geometry format.

 **The available geometry types in Hana are st\_geometry and st\_point, but they are functionally identical (with STPoint being restricted to point geometries). As such, DTS treats them identically.**

## Geometry Types

Here is how SAP Hana (ST) geometry subtypes map to GeoJson geometry types.

ST Subtype	Interpretation	GeoJson Geometry Type
POINT	Single point	Point
MULTIPOINT	Multiple points	MultiPoint
LINESTRING	Sequence of straight lines without gaps	LineString
CIRCULARSTRING	Sequence of circle arcs without gaps	LineString + DTS corrections
MULTILINESTRING	Multiple LINESTRINGs (not necessarily connected)	MultiLineString
COMPOUNDCURVE	Sequence of connected LINESTRINGs and CIRCULARLINESTRINGs	MultiLineString + DTS corrections
POLYGON	Polygon (with or without holes)	Polygon
CURVEPOLYGON	Polygon whose boundaries can contain circle arcs	Polygon + DTS corrections
MULTIPOLYGON	Multiple POLYGONs	MultiPolygon (+ DTS corrections)
GEOMETRYCOLLECTION	Collection of any of the above	GeometryCollection (+ DTS corrections)

Z and M modifiers (e.g. POINT Z) are supported for all geometry types (except GEOMETRYCOLLECTION where they are not applicable), but the M values are always ignored and processing geometries with Z values has certain [limitations](#) <sup>158</sup>.

### Further reading:

- [SAP HANA Spatial Reference](#)
- [DTS Geometry Overview](#) <sup>222</sup>

## 4.7.4 Tables and Views

The DTS SAP Hana Connector fully supports all tables (column-based and row-based) and views in SAP Hana and does not differentiate between them in any meaningful way.

## DTS Naming


<b>DTS Name</b>	<table_or_view_name>
<b>DTS Key</b>	<schema_name>.<table_or_view_name>

## Metadata

The connector is also capable of extracting metadata regarding the table/view columns:

- DATA\_TYPE\_NAME
- LENGTH
- SCALE
- DEFAULT\_VALUE
- IS\_NULLABLE
- POSITION
- CONSTRAINT -> PRIMARY\_KEY

Standard functionality only makes use of DATA\_TYPE\_NAME, POSITION and PRIMARY\_KEY, but the other items are available for extension in DTS descriptor objects.

 Structured types (table types) are not supported as column data types by SAP HANA, only native and array types are. Please see [Types](#)<sup>[154]</sup> for information of how the connector handles these.


### 4.7.5 Routine Calls

The DTS SAP Hana Connector supports remote calls for any SAP Hana stored procedures and functions, scalar-valued as well as table-valued.

## DTS Naming

<b>DTS Name</b>	<proc_or_func_name>
<b>DTS Key</b>	<schema_name>.<proc_or_func_name>

## Arguments and Results

- **Table type arguments** to functions and procedures, as well as results of table-valued functions are fully supported for remote calling
    - The table arguments are always passed through DTS as arrays of custom objects, each object representing a row in the input table
    - The result of a table-valued function is always a DTS stream
-  See [Streams](#) <sup>220</sup>
- SAP Hana does not have a way of representing a **single custom object**, other than a table with a single row, so that extends to DTS, where they will become arrays with a single element.

For argument and result native and user type mappings, please see [Types](#) <sup>154</sup>.

## Metadata

The connector is also capable of extracting metadata regarding the arguments and results of a procedure or function:

- DATA\_TYPE\_NAME
- LENGTH
- SCALE
- POSITION
- PARAMETER\_TYPE
- TABLE\_TYPE
- HAS\_DEFAULT\_VALUE
- IS\_NULLABLE

Standard functionality only makes use of DATA\_TYPE\_NAME, POSITION, PARAMETER\_TYPE and TABLE\_TYPE, but the other items are available for extension in DTS descriptor objects.

### 4.7.6 Limitations

- DB Synonyms are not currently supported.
- The following native data types are not currently supported:
  - **blob**

- **clob**
- **nclob**
- **text**
- **bintext**
- Any array types with elements of these native types are not supported.
- Any table types that contain columns which use these native types are not supported.
- M (measure) values on "POINT M" or "POINT ZM" geometries are ignored by the connector and are not passed through.
- General DTS limitations regarding Z-valued geometries apply. Please see [Known Limitations](#)<sup>322</sup> for more details.

## 4.8 Smallworld

The Smallworld Connector allows DTS and its clients to access data and functionality in GE Smallworld environments.

<b>Development Platform</b>	Smallworld Magik + Java 8
<b>Deployment Paradigm</b>	Docker Container + Prepared <a href="#">SW environment</a> <sup>160</sup>
<b>Metadata Mode Paradigm</b>	Remote
<b>DTS_PRODUCER_CATEGORY</b>	Smallworld

- [Connection Parameters](#)<sup>160</sup>
- [Environment](#)<sup>160</sup>
- [Types](#)<sup>161</sup>
- [Geometry](#)<sup>163</sup>
- [Collections](#)<sup>164</sup>
- [Method/Procedure Calls](#)<sup>166</sup>
- [Limitations](#)<sup>181</sup>

### 4.8.1 Connection Parameters

**Views and Alternatives** These parameter pairs represent the dataset views the connector will access and what alternative will be used for each view.

 **At least one View-Alternative pair must be registered for the connector to be able to find items in the Smallworld datasource**

 **Multiple pairs with the same View name are not supported (i.e. only one Alternative can be registered for each View)**

### 4.8.2 Smallworld Environment

*[NEEDS DETAILING]*

This section explains how to prepare a Smallworld environment for running the DTS Smallworld Connector.

Multiple approaches are possible, and others may be favored depending on the broader environment, but we'll go through the simplest one to better illustrate what's necessary.

For a more tailored solution, please contact support.

#### 1. Smallworld Image


The image the connector will use needs to be open, with access to the desired datasets, and have all the necessary modules loaded for using the dataset objects.

#### 2. Loading the Connector Product

The Smallworld connector product and module need to be loaded into the already open image.

#### 3. Setting environment variables

#### 4. Registering Functionality

Any procedures or methods that the connector should be aware of should now be [registered](#), as well as their type dependencies.

#### 5. Starting the Connector

The connector can now be started using this command:

dts\_manager.activate\_producer()

## 6. Putting it together

To automate this process for normal use, steps 2-5 should be added to a script:

Finally, an alias should be created to open the image and run the script on startup:

## Integrating with DTS

To allow DTS to control instances of this connector, the connector container must be set up to access this Smallworld Environment.

### 4.8.3 Types

Smallworld Type	DTS Type
char16_string	STRING
char16_vector	STRING
char16_vector_vec	STRING
string	STRING
ds_lbyte_charvec	STRING
ds_char16_vector	STRING
ds_char	STRING
ds_char_vec	STRING
ds_charci	STRING
ds_charci_vec	STRING
ds_char16	STRING
ds_char16_vec	STRING
ds_char16canon	STRING
ds_char16canon_vec	STRING
extdb_string	STRING
extdb_char	STRING
extdb_char_vec	STRING
text_join	STRING
ds_byte	UNSIGNED_INT8
ds_short	INT16
ds_ushort	UNSIGNED_INT16

integer	INT32
ds_int	INT32
ds_uint	UNSIGNED_INT32
ds_long	INT64
ds_int64	INT64
ds_uint64	BIG_INTEGER
bignum	BIG_INTEGER
ds_float	FLOAT
ds_double	DOUBLE
float	DOUBLE
ds_double_as_int	DOUBLE
ds_bool	BOOLEAN
ds_kleene	STRING
symbol	STRING
ds_date	DATE
date	DATE
date_time	DATE_TIME
ds_time	TIME
ds_simple_time	TIME
gis_id	UNSIGNED_INT32_VECTOR
gis_id64	UNSIGNED_INT64_VECTOR
simple_point	DTS_POINT_GEOMETRY
point	DTS_POINT_GEOMETRY
simple_chain	DTS_CHAIN_GEOMETRY
chain	DTS_CHAIN_GEOMETRY
simple_area	DTS_AREA_GEOMETRY
area	DTS_AREA_GEOMETRY
text (geometry)	DTS_ANNOTATION_GEOMETRY

Slotted exemplar types need to be registered, but they are only relevant for [Method/Procedure Calls](#)<sup>[166]</sup> and are discussed there.

Join types are automatically generated, but they are only relevant for [Collections](#)<sup>[164]</sup> and are discussed there.

**i See also**  
[DTS Types](#)<sup>[218]</sup>

#### 4.8.4 Geometry

The DTS Smallworld Connector can access geometries in the native Smallworld storage formats when streaming data from Collections and exchanges them using its own Magik wrappers for Method/Procedure Calls.

Here are the mappings:

Smallworld Geometry (DS type)	DTS Magik Wrapper (DTS Type)	GeoJson Geometry Type
point/simple_point	dtc_point (DTS_POINT_GEOMETRY)	Point
chain/simple_chain	dtc_chain (DTS_LINE_GEOMETRY / DTS_MULTILINE_GEOMETRY)	LineString / MultiLineString (+ DTS Corrections)
area/simple_area	dtc_area (DTS_AREA_GEOMETRY / DTS_MULTIAREA_GEOMETRY )	Polygon / MultiPolygon (+ DTS Corrections)
text	dtc_text (DTS_ANNOTATION_GEOMETRY)	Point (+ DTS Extras)

Smallworld differentiates between networked (topologically embedded) and simple (with no topology) geometries. The Smallworld Connector removes that distinction, as there is no standard for codifying topological data. It is still available in raw form if desired.

Smallworld chain and area geometry objects can encode multiple disconnected lines and polygons respectively, so they can result in the simple or "Multi" variant as they are streamed out.

When accessing [Collections](#)<sup>[164]</sup> with geometry fields through the Smallworld Connector, the geometries are automatically transmuted through their respective DTS Magik Wrappers and serialized as **DTSGeoJson** - no extra action is needed.

When using geometries within Method or Procedure arguments or results, the respective items must be registered with their DTS Type (see [Method/Procedure Calls](#)<sup>[166]</sup>). They will receive and provide geometry data in the DTS Magik Wrapper format. For convenience, the DTS Magik Wrapper exemplars provide the following methods:


- **new\_from(<a\_geom>)** : creates a new instance of a DTS Magik Wrapper with all the usable characteristics of the input geometry. Both ds and pseudo geometries are accepted.

- **as\_pseudo\_[point|chain|area|text]()** : transforms the DTS Magik Wrapper object into the corresponding pseudo geometry (pseudo\_point, pseudo\_chain, pseudo\_area, pseudo\_text).

Smallworld geometries can encode more information than standard GeoJson can accommodate, so the Connector uses DTS Geometry **Corrections** and **Extras** to include that data:

- **(simple)\_points** can be oriented in Smallworld. The Connector uses the DTSGeoJson **orientation** field to encode this information.
- **texts** are oriented points with extra information like justification and the actual content string. This extra information is placed in the **extras** field of the DTSGeoJson.
- **(simple)\_chains** and **(simple)\_areas** can contain sectors that are not straight line strings. These will be approximated for standard use and the exact representation included in the DTSGeoJson corrections field as follows:

Smallworld Sector Type	Interpretation	Correction Line Type
sector	Basic sector, represents a straight line string	none (LINE_STRING*)
arc	Single elliptical arc	CP_ARCS or NURBS **
circle	Circle	CIRCLE
rational_b_spline	Rational B-spline	NURBS

 **sector\_z** is not currently supported (i.e. only 2D geometries are currently supported)

 \* **LINE\_STRING** corrections are not produced by this connector or any of the other standard connectors. It exists for extensions and optimization.

 \*\* If the arc is circular, it results in a **CP\_ARCS** correction, otherwise it's encoded as **NURBS**. Multiple **CP\_ARCS** corrections may be compounded into a single one.

 **Further reading:**

- [DTS Geometry Overview](#) 

#### 4.8.5 Collections

The DTS Smallworld Connector provides access to all collections in the requested DS Views (for the given alternatives), with the following exceptions:

- internal collections (e.g. join intermediate tables, geometry storage tables, etc.);

- drawing and raster collections (dimensioning, raster\_map, background\_raster, logo, function\_geometry, drawing\_template, drawing\_function).

## DTS Naming

**DTS Name** <collection\_name>

**DTS Key** <view\_name>.<collection\_name>

## Fields

The Smallworld Connector handles various fields as follows:

- **Physical Fields:** All physical fields of [known types](#) <sup>[161]</sup> are supported, except internal fields, which are ignored.
- **Logical Fields:** Derived fields are treated like physical fields, so the same rules apply.

### **Logical Field return values are not type-enforced in Smallworld in any way**

The Connector relies on the types declared for the field in its metadata (case), but has no way to verify that the method associated to the field always returns the declared type. If there are discrepancies, this can cause errors while streaming data!

- **Geometry Fields:** All geometry fields are supported except dimension and raster fields, which are ignored.
- **Join Fields:** Homogeneous join fields of all cardinalities (0:0, n:1, m:n, etc.) are supported, but heterogeneous join fields currently are not.
  - The Smallworld Connector packages join field values according to result multiplicity (i.e. whether more than one result can be expected):
    - **Single Result:** The join value offered through DTS will be a single *Minimal Record* \* for the collection targeted by the join. (e.g.: n:1, 0:1)
    - **Multiple Results:** The join value offered through DTS will be an array of *Minimal Records* \* for the collection targeted by the join. (e.g.: m:n, 1:n)

 \* A Minimal Record is a record from a collection containing only the fields which compose the primary key. It serves as an identifier, or a foreign key.

 Text join fields are treated like physical fields and will be populated with a STRING

## Metadata

The Smallworld Connector includes the following field metadata in the DTS descriptor structures:

- type
- type\_size
- mandatory? (can\_be\_unset)
- simple\_geometry

#### 4.8.6 Routine Calls

Since a Smallworld image is a large development environment, it is not feasible to automatically extract available methods and procedures. There are simply too many, and the vast majority will be unrelated to the desired functionality.

Further more, method and procedure declarations in Magik do not specify the types of the arguments and returns (Magik is a weak-typed language), so even given a method and procedure, its parameters cannot be automatically mapped.

As a result, the DTS Smallworld Connector expects that any method or procedure it should be aware of will be **pre-registered**.

#### DTS Naming

**DTS Name** <registered\_dts\_key>

**DTS Key** <registered\_dts\_key>

##### 4.8.6.1 Registration

The Smallworld Connector provides a set of methods for the purpose of registering routine objects. These methods should be used in a registration script that is executed after the Connector is loaded into the Smallworld image and before the Connector is activated (see [Environment](#)<sup>[160]</sup>).

#### A. Exemplar Type Registration

```
dts_manager.register_type(  
    p_dts_key,
```

```
p_exemplar,  
p_slots)
```

- **p\_dts\_key**: the key by which this type will be known throughout DTS
- **p\_exemplar**: the slotted exemplar the type describes (this should be the actual exemplar, not just its name)
- **p\_slots**: the slots on the exemplar that should be included in the type as a vector of this form:

```
{  
  {<name>, <dts_type>, <options>...},  
  ...  
}
```

- **<name>**: the name of the slot on the exemplar
- **<dts\_type>**: the DTS type key for the slot. Can be a [pre-defined type](#)<sup>161</sup>, or a compound type that has already been registered using this method or the next one.
- **<options>...**: a list of key-value pairs defining various options for the slot
  - The only option currently supported is **:is\_vector?**, which expects a value of **\_false** or **\_true** and determines if the parameter is a **<dts\_type>** or an **array of <dts\_type>**. If not provided, the default is **\_false**.

 The Smallworld Connector will strictly adhere to the types it is instructed to used on each slot. If it is provided with a different object type, communication may fail.

## B. DS Type Registration

```
dts_manager.register_ds_type(  
  p_dts_key,  
  p_collection,  
  p_fields)
```

- **p\_dts\_key**: the key by which this type will be known throughout DTS
- **p\_collection**: the collection that this type describes (this should be a handle on the actual collection, not just its name)
- **p\_fields**: the fields from p\_collection that should be included in the type as a vector of field names:

```
{<field1>, <field2>, ...}
```

## C. Remote Call Registration

```
dts_manager.register_api_call(  
  p_dts_key,  
  p_name,
```

```
p_parameters,
p_results,
_optional p_options)
```

- **p\_dts\_key**: the key by which the remote call will be known throughout DTS
- **p\_name**: the name which will be used to invoke the method or procedure locally
- **p\_parameters**: the input parameters of the method/procedure as a simple\_vector of this form:

```
{
  {<dts_type>, <options>...},
  ...
}
```

- **<dts\_type>**: the DTS type key for the parameter. Can be a [pre-defined type](#)<sup>161</sup>, or a compound type that has already been registered using one of the **type registration methods**
- **<options>...**: a list of key-value pairs defining various options for the parameter.
  - The only option currently supported is **:is\_vector?**, which expects a value of **\_false** or **\_true** and determines if the parameter is a **<dts\_type>** or an **array of <dts\_type>**. If not provided, the default is **\_false**.
- **p\_results**: the outputs of the method/procedure, in the same form as **p\_parameters**, except:
  - Single results can also use the **:stream?** option with a value of **\_true** or **\_false**, which determines whether the routine will produce a stream output.

⚠ Routines that have a result marked as stream, should only have that single result - registering multiple results will cause errors.

**i** For more information regarding stream-valued routines in Smallworld, please see the [dedicated documentation page](#)<sup>169</sup>.

- **p\_options**: a property\_list containing various options for the remote call
  - **:procedure?** : (**\_true** or **\_false**) represents whether or not this call is to a **procedure**. If **\_false**, the call is to a **method** and a **:target** option is also expected. The default value is **\_true**.
  - **:target** : the name of the **exemplar** which will be used to invoke the method.

**i** Methods are invoked statically on the exemplar provided in **p\_options[:target]**, so it should function as a singleton in when receiving the **p\_name** message

**i** **p\_name** and **p\_options[:target]** should be fully qualified with the package name, as well as brackets (if required)

 The Smallworld Connector will strictly adhere to the types it is instructed to use on each parameter. If it is provided with a different object type, communication may fail.

#### 4.8.6.2 Stream Results

For a Magik routine (method or procedure) to produce a stream for DTS, it must return only one result and it must be a *stream-type object*.

DTS considers any object that responds to **close()** and **get\_upto\_n()** to be a *stream-type object*.

While certain existing Magik constructs like *record\_stream*, *difference\_stream* and others do satisfy the stream-type object criteria, DTS also provides its own stream wrapper (**dts:dts\_record\_stream**) for generating streamable responses.

**dts:dts\_record\_stream** provides the following constructor methods:


##### A. For Wrapping Collections:

```
dts_record_stream.new_for_collection(
    p_id,
    p_collection,
    _optional p_predicate)
```

- **p\_id**: is ignored in when the method is used in this scenario;
- **p\_collection**: the collection to stream from;
- **p\_predicate**: a predicate to query the collection. If unset, the entire collection will be streamed.

Example:

```
_global stream_example1 <<
_proc @stream_example1 ()
    _local collection << gis_program_manager.cached_dataset(:gis).collection(:m
    _return dts:dts_record_stream.new_for_collection(_unset, collection, predic
_endproc
$
```

 In this case, we would need to register a *min\_road* DS type with DTS first, then register the routine call returning a stream of *min\_road* types.

##### B. For Wrapping Other Streams:

```
dts_record_stream.new_for_stream(
```

```
p_stream)
```

- **p\_stream**: a stream-type object

**i** While it may seem somewhat redundant to wrap a stream-type object into a `dts_record_stream`, we recommend this to be done whenever feasible due to better compliance of the `dts_record_stream` objects with the DTS engine

Example:

```
_global stream_example2 <<
_proc @stream_example2 ()
    _local stream << gis_program_manager.cached_dataset(:gis).collection(:min_r
    _return dts:dts_record_stream.new_for_stream(stream)
_endproc
$
```

### C. For Wrapping Vectors:

```
dts_record_stream.new_for_vector(
    p_vector)
```

- **p\_vector**: a Magik vector-type object (must respond to `subseq()` and `size`) to be wrapped into a stream.

Example:

```
_global stream_example3 <<
_proc @stream_example3 (p_size)
    _local vector << rope.new()
    _for i_index _over 1.upto(p_size)
    _loop
        vector.add(some_object.new(i_index))
    _endloop
    _return dts:dts_record_stream.new_for_vector(vector)
_endproc
$
```

**i** In this case, we would to register an exemplar type for `some_object` with DTS, then register the routine call returning a stream of `some_object` types.

#### 4.8.6.3 Registration Example

```
_package user
```

```
#####
## Count underground routes ##
#####

# Here's our mockup procedure

_global !count_all_underground_routes! <<
  _proc @!count_all_underground_routes!()
    _local v << gis_program_manager.cached_dataset(:gis)
    _return v.collection(:underground_route).size
  _endproc
$

# This procedure can be registered directly, since it only uses a predefined type

dts_manager.register_api_call("count_all_underground_routes", "user:!count_all_u
    {},
    {
        {dts:dts_type_mapping.uint32}
    })
$

#####
## Get routes in the given area ##
#####

# Another one - a bit more complex

_global !get_uroute_ids_in_area! <<
  _proc @!get_uroute_ids_in_area!(p_area, p_max_size)
    _local v << gis_program_manager.cached_dataset(:gis)
    _local l_area << _if p_area.is_class_of?(dts:dts_area)
      _then >> pseudo_area.new_for_world(p_area.as_pseudo_area())
      _else >> pseudo_area.new_for_world(p_area, v.world)
    _endif
    _local rts << v.collection(:underground_route).select(predicate.within
    _local rt_ids << rope.new()
    _local idx << 1
    _for rt _over rts.fast_elements()
    _loop
      _if idx > p_max_size _then _leave _endif
      rt_ids.add(rt.id)
      idx +<< 1
    _endloop
    _return rt_ids.as_simple_vector()
  _endproc
$
```

```

# This procedure also used only predefined types, but has arguments and returns a
dts_manager.register_api_call("get_uroute_ids_in_area", "user:!get_uroute_ids_in_
  {
    {dts:dts_area.dts_type_key},
    {dts:dts_type_mapping.uint32}
  },
  {
    {dts:dts_type_mapping.uint32, :vector?, _true}
  })
$

#####
## Get details about underground route ##
#####

# For this one, we'll need some custom objects, which we define here:

def_slotted_exemplar(:cable_info,
  {
    {:id, _unset, :writable, :public},
    {:name, _unset, :writable, :public},
    {:kind, _unset, :writable, :public},
    {:length, _unset, :writable, :public}
  })
$

_method cable_info.new()
  _return _clone.init()
_endmethod
$

_private_method cable_info.init()
  _return _self
_endmethod
$

def_slotted_exemplar(:uroute_info,
  {
    {:uroute_record, _unset, :writable, :public},
    {:cable_infos, _unset, :writable, :public}
  })
$

_method uroute_info.new()
  _return _clone.init()
_endmethod
$

```

```

_private _method uroute_info.init()
    _return _self
_endmethod
$

# And here is the procedure

_global !get_uroute_details! <<
    _proc @!get_uroute_details!(p_id)
        _local v << gis_program_manager.cached_dataset(:gis)

        _local uroute << v.collection(:underground_route).at(p_id)

        _local response << uroute_info.new()
        response.uroute_record << uroute

        _local cblds << rope.new()
        _for c _over uroute.mit_structure_route.mit_cables.fast_elements()
        _loop
            _local cbl << cable_info.new()
            _if c.copper_cable _isnt _unset
            _then
                cbl.id << c.copper_cable.id
                cbl.name << c.copper_cable.name.write_string
                cbl.kind << "copper"
                cbl.length << c.copper_cable.calculated_length.as_float
            _elif c.sheath_with_loc _isnt _unset
            _then
                cbl.id << c.sheath_with_loc.id
                cbl.name << c.sheath_with_loc.name.write_string
                cbl.kind << "fiber"
                cbl.length << c.sheath_with_loc.calculated_fiber_length.as_float
            _else
                cbl.id << c.id
                cbl.name << "uninteresting cable"
                cbl.kind << "irrelevant"
                cbl.length << _unset
            _endif
            cblds.add_last(cbl)
        _endloop
        response.cable_infos << cblds.as_simple_vector()
        _return response
    _endproc
$

# First, we register a DS type for :underground_route with the fields we're inte

```

```

dts_manager.register_ds_type("dts_uroute", gis_program_manager.cached_dataset(:gis_program_manager) {
  {
    "id",
    "construction_status",
    "underground_route_type",
    "route"
  })
$

# Then, we register slotted exemplar types for our cable_info and uroute_info exemplars
# Note how uroute_info refers to cable_info. The referrer always must come after the referree

dts_manager.register_type("cable_info", cable_info,
  {
    {"id", dts:dts_type_mapping.uint32},
    {"name", dts:dts_type_mapping.string},
    {"kind", dts:dts_type_mapping.string},
    {"length", dts:dts_type_mapping.float}
  })
$

dts_manager.register_type("uroute_info", uroute_info,
  {
    {"uroute_record", "dts_uroute"},
    {"cable_infos", "cable_info", :vector?, _true}
  })
$

# Finally, we can register the routine

dts_manager.register_api_call("get_uroute_details", "user:!get_uroute_details!",
  {
    {dts:dts_type_mapping.uint32}
  },
  {
    {"uroute_info"}
  }
)
$

#####
## Method ##
#####

# To have a method to call, we need a singleton exemplar

def_slotted_exemplar(:test_singleton, {})
$

```

```

# And a test method

_method test_singleton.test_multi(p_string, p_int)
  _return (p_string, p_int, 0.003)
_endmethod
$

# Here's how we register it
# Notice that we populated to options in order to tell the connector that it's a

dts_manager.register_api_call("test_singleton.test_multi", "test_multi()",
                              {{dts:dts_type_mapping.string}, {dts:dts_type_mapping.uint}},
                              {{dts:dts_type_mapping.string}, {dts:dts_type_mapping.uint}},
                              property_list.new_with(:procedure?, _false, :target, "user")
$

#####
## Streaming Routine ##
#####

# First, we need an exemplar for the stream elements

def_slotted_exemplar(:test_item,
  {
    {:part1, _unset, :writable, :public},
    {:part2, _unset, :writable, :public}
  }, {})
$

_method test_item.new(a_string, an_int)
  _return _clone.init(a_string, an_int)
_endmethod
$

_private _method test_item.init(a_string, an_int)
  .part1 << a_string
  .part2 << an_int
  _return _self
_endmethod
$

# We register the type with DTS

dts_manager.register_type("test_item", test_item,
  {"part1", dts:dts_type_mapping.string},

```

```

                                {"part2", dts:dts_type_mapping.int32}})
$

# Here is our procedure that returns a stream of test_items

_global !streamy_fun_test! <<
_proc @!streamy_fun_test!(p_size)
    _local v << rope.new()
    _for i _over 1.upto(p_size)
    _loop
        v.add(test_item.new(write_string(i), i))
    _endloop
    _return dts:dts_record_stream.new_for_vector(v)
_endproc
$

# And finally we register the procedure with DTS

dts_manager.register_api_call("streamy_fun_test", "user:!streamy_fun_test!",
                                {{dts:dts_type_mapping.int32}},
                                {"test_item", ":stream?", _true}})
$

```

#### 4.8.7 Smallworld Client

The same fileset used for the Smallworld Connector also provides [Client](#)<sup>213</sup> functionality. This allows you to access the resources included in a DTS project from the Magik console or from Magik code.

#### Preparation

The Smallworld Client is enabled using the following steps:

- Open a Smallworld image - anything based on swaf (or swaf itself) is sufficient;
- Load the DTS Smallworld product and module in the image;
- Set the environment variables for accessing DTS:

Variable	Value
<b>DTS_JAVA_HOME</b>	The path to the Java binary directory
<b>DTS_DIST_HOME</b>	The path to the Smallworld Connector's jars directory

<code>DTS_REDIS_HOST_NAME</code>	The hostname or IP address of the Redis server (default is localhost)
<code>DTS_REDIS_PORT</code>	The port for the Redis server (default is 6379)

 Being based on the [Client Java Library](#)<sup>[257]</sup>, the Smallworld Client also responds to all other environment variables specified there.

 If the security scheme is enabled in the DTS cluster, the Smallworld Client must also implement it. See [Security Setup](#)<sup>[252]</sup> for details.

- Boot the client:

```
Magik> dts_manager.activate_consumer("<project_name>")
```

## API

Once the Client is booted, you have access to the `dts_remote_manager` singleton exemplar where you can begin accessing resources.

Here are the currently available methods for `dts_remote_manager`:

Method	Result
<code>connectors</code>	equality set of the available connectors ( <code>dts_remote_connector</code> ), keyed by name

Each connector entry value is a `dts_remote_connector`, where you can access:

Method	Result
<code>collections</code>	equality set of the available collections for the connector ( <code>dts_remote_collection</code> ), keyed by native identifier
<code>functions</code>	equality set of the available routines for the connector ( <code>dts_remote_function</code> ), keyed by native identifier
<code>topics</code>	equality set of the available topics for the connector ( <code>dts_remote_topic</code> ), keyed by native identifier

For a `dts_remote_collection`, you can call:

Method	Result
--------	--------

<code>get_record(_opt)</code>	the first record in the collection (that matches the predicate)  <code>p_predicate</code> can be any regular Magik predicate or a <a href="#">dts_predicate</a> <sup>[239]</sup>
<code>get_records(p_size)</code>	the first <code>p_size</code> records in the collection (that match the predicate)
<code>get_record_stream</code>	opens a stream on the collection (using the predicate) and returns it as a <code>dts_remote_record_stream</code> object
<code>descriptor</code>	returns the <code>dts_type_descriptor</code> (see below) object that describes the collection

For a `dts_remote_topic`, you can call:

Method	Result
<code>subscribe(_opt)</code>	subscribes to the topic and returns a <code>dts_remote_record_stream</code> object  <code>p_group_id</code> must be a string and <code>p_properties</code> can be a <code>property_list</code> or <code>equality_property_list</code>
<code>push(p_key, p_msg)</code>	pushes the <code>p_key</code> , <code>p_msg</code> pair to the Topic and returns DONE or raises a condition
<code>push_lists(p_keys)</code>	pushed all pairs <code>p_keys[i]</code> , <code>p_msgs[i]</code> to the Topic and returns DONE or raises a condition
<code>push_properties</code>	pushes all key-value pairs in <code>p_properties</code> (which should be a <code>property_list</code> or <code>equality_property_list</code> ) and returns DONE or raises a condition
<code>descriptor</code>	the descriptor for the topic as a <code>dts_topic_descriptor</code>
<code>record_wrapper</code>	the <code>dts_type_descriptor</code> for the topic's stream records

For a `dts_remote_record_stream`, you can call:

Method	Result
<code>get()</code>	gets a single record from the stream
<code>get_upto_n(p_size)</code>	gets <code>p_size</code> records from the stream or how many are left if less than <code>p_size</code>

<code>poll()</code>	special implementation of <code>get_upto_n()</code> for Topics with no size and no <code>has_more?</code> impact
<code>has_more?</code>	whether or not there are more records to get from the stream (for Topic streams it always returns <code>_true</code> )
<code>close()</code>	closes the stream
<code>unsubscribe()</code>	same as <code>close()</code> but in Topic vernacular

For a `dts_remote_function`, you can call:

Method	Result
<code>execute(_gather p_args)</code>	executes the remote routine using the provided arguments ( <code>p_args</code> ) and returns the results
<code>input_wrapper_descriptor</code>	returns the <code>dts_type_descriptor</code> (see below) object that describes the inputs of the routine (in the form of a wrapper object whose attributes are the actual inputs)
<code>output_wrapper_descriptor</code>	returns the <code>dts_type_descriptor</code> (see below) object that describes the outputs of the routine (in the form of a wrapper object whose attributes are the actual inputs)

For remote collection and routine metadata, `dts_type_descriptor` objects provide the following:

Method	Result
<code>key</code>	the type key of the resource structure
<code>attributes</code>	a <code>simple_vector</code> containing the resource structure's attributes as <code>dts_attribute_descriptor</code> objects
<code>stream_exemplar</code>	the Magik exemplar used for wrapping resource structures of this type for ACPT data marshalling

Each `dts_attribute_descriptor` provides the following:

Method	Result
--------	--------

<code>name</code>	the name of the attribute
<code>type</code>	the type key of the attribute
<code>category</code>	the type category of the attribute
<code>details</code>	details regarding the attribute as a <code>simple_vector</code> of <code>dts_attribute_detail</code> objects, each answering to <code>.name</code> and <code>.info</code>
<code>get_detail_info(p_</code>	returns the value of the detail identified by <code>p_detail_name</code>
<code>can_be_unset?</code>	returns whether or not the attribute can be <code>_unset</code> (i.e. the value of the "unset?" detail)
<code>is_vector?</code>	returns whether or not the attribute is a vector (i.e. the value of the "vector?" detail)

Each `dts_topic_descriptor` provides the following:

Method	Result
<code>key</code>	the native identifier of the topic
<code>record_wrapper_type</code>	the type key for the topic's stream record
<code>key_type_key</code>	the type key for the topic's key field
<code>msg_type_key</code>	the type key for the topic's msg field

Sometimes, attributes can have type keys that represent other slotted (structure) types. To access the `dts_type_descriptor` for any type key, you can use:

```
Magik> dts:dts_types_repository.types["<type_key>"].descriptor
```

## Shutdown

To disconnect from DTS , use:

```
Magik> dts_manager.shutdown()
```

The Smallworld Client will automatically shutdown whenever `quit()` is invoked in the Smallworld session.

### Other Considerations

- The DTS Smallworld Client keeps processes running in the background. In the event of an improper shutdown of the session, these processes can be left orphaned and impede subsequent client operations. In such situations, it is recommended that sw\_magik and java processes running on the machine are audited and manually stopped as necessary.
- The Magik language is weakly typed, so there is not type enforcement on objects fed as parameters for the routines and collection operations DTS makes available. Most often, improper types will be met with ACPT exceptions generated by the Smallworld data marshalling engine that the Smallworld Connector employs.

#### 4.8.8 Limitations

- Only one alternative can be available for any given dataset view
- 3D geometries are not currently supported
- Heterogeneous joins are not currently supported and are filtered out

## 4.9 Web Service

---





The Web Service Connector allows DTS and its clients to access functionality from REST and SOAP Web Services.

**Development Platform** Java 8 (Apache CXF client generation)

**Deployment Paradigm** Docker Container

**Metadata Mode Paradigm** Local


**DTS\_PRODUCER\_CATEGORY** WebService

- [Connection Parameters](#)  182
- [Types](#)  182
- [Method Calls](#)  183
- [Limitations](#)  184


#### 4.9.1 Connection Parameters

<b>URL</b>	The URL to the Web Service's specification file. For SOAP services, this is a WSDL specification. For REST services, this can be an OpenAPI/Swagger (json or yaml) or a WADL specification.
<b>Username</b>	Username for the user through which DTS will connect to the service. Only required for services which require login authentication.
<b>Password</b>	Password for the user.
<b>Token</b>	The token or application id to be used for accessing the service. Only required for services with token authentication.

 The URL must point directly to the specification file (e.g. "https://my.site/service/openapi.json").

 The URL can also be an absolute network path accessible from inside the DTS Docker containers (e.g. "//a\_machine/share/service.wsdl") or a local path inside the container that is mapped to a host path (e.g. "/usr/local/wss/producer/openapi.json").

 Username and Password authentication is currently supported only as standard Basic Authentication.

 While most SOAP services have a WSDL specification attached (most application servers generate it upon deployment), this is not necessarily the case for OpenAPI/Swagger, WADL and REST. Depending on the service you are looking to consumer, it may be necessary to manually create a specification file for it. We recommend using OpenAPI 3.0 in this case. The [OpenAPI documentation](#) can help guide that process.

#### 4.9.2 Types

The DTS Web Service Connector uses **wsdl2java**, **wadl2java** (from Apache CXF) and **openapi-generator** to generate Java client classes for consuming SOAP and REST web services.

As such, initial type mapping is done by these tools, from the declared specification types to Java object types. The connector then maps the resulting Java base types to DTS types as follows:

Java Type	DTS Type
Object	STRING
String	STRING
XMLGregorianCalendar	DATE_TIME
LocalDate	DATE_TIME
Byte/byte	INT8

Short/short	INT16
Integer/int	INT32
Long/long	INT64
Float/float	FLOAT
Double/double	DOUBLE
Boolean/boolean	BOOLEAN
BigInteger	BIG_INTEGER
BigDecimal	DECIMAL
Duration	STRING
QName	STRING
byte[]	BYTE_VECTOR

Any complex object structure that can resolve down to these base types is fully supported.

**i** While other connectors use DTS's own Java class generation engine, the Web Service Connector relies on CXF and openapi-generator for this task.

This ensures the best possible compliance with increasingly complex and heterogeneous web standards.

**i** Further reading

[XSD \(WSDL, WADL\) - Java Type Mappings](#)

[OpenAPI Data Types](#)

**i** See also

[DTS Types](#) <sup>218</sup>

### 4.9.3 Routine Calls

The DTS Web Service Connector supports remote calls for any Web Methods exposed by a SOAP or REST service as long as they are correctly described in the specification document.

#### DTS Naming

**DTS Name** <web\_method\_name>

**DTS Key** <port\_name>.<web\_method\_name>

For WSDL specifications, the <port\_name> and <web\_method\_name> are extracted directly from the specification document as the **name** attributes of the **portType** nodes and the **name** attributes of the **operation** nodes respectively.

For OpenAPI/Swagger and WADL specifications, method and port (actually the client class in this case) naming is left to the respective generator tools (**openapi-generator**, **wadl2java**), which use **tags**, **operation types** and **operation ids** to generate the names.

## Requests and Responses

The Web Service Connector, via its Apache CXF core, will make requests and expect responses for Web Method Calls in the document formats that the specification file calls for.

Any combination of **REST** and **SOAP** on one hand and **application/xml** and **application/json** on the other are supported by the engine, with the [FasterXML/jackson](#) library being used for (de) serialization.



### Further reading

[OpenAPI v3 Specification](#)

[OpenAPI v2 \(Swagger\) Specification](#)

[WADL Specification](#)

[WSDL Specification](#)

[Apache CXF](#)

### 4.9.4 Limitations

- There is no bespoke geometry support in the Web Service Connector. None of the Web Service specification standards used have special provisions for geometry types, as such, any geometry object will be described in terms of the simpler types it contains.



# Webservices

## 5 Webservices

DTS has the capability to generate Web Clients for its projects without any programming intervention. These Clients simplify access to DTS data for any system that is capable of performing REST or SOAP Web Requests.

This chapter explains how DTS generates and manages Webservices, what features they can provide and how to integrate them into your business process.

The following topics will be approached:

- [Types](#) <sup>186</sup>
- [Functionality](#) <sup>187</sup>
- [Specification](#) <sup>195</sup>
- [Access](#) <sup>195</sup>
- [Integration](#) <sup>199</sup>

**i** For information on how to create and deploy Webservices from the Web UI, see [Web UI - Webservices](#) <sup>70</sup>

### 5.1 Types

---

DTS can currently generate two types of services: REST and SOAP.

The type you should use largely depends on your [integration](#) <sup>199</sup> necessities. If you are in doubt about what type of service you should generate in a specific case, please contact support for more information.

This section lists some particularities of each service type as generated by DTS.

#### REST

- Implements the REST API using Apache CXF over JAX-RS;
- Communicates in JSON format;
- Provides GET operations where possible (the input parameters are compatible with URL parametrization, i.e. STRING and INTEGER variants) - see [Functionality](#) <sup>187</sup> for specifics;

- Provides [specifications](#)<sup>[195]</sup> in WADL and OPENAPI 3.0 formats;
- Is less strict in the interpretation of input and output formats.

 [Read more about REST APIs](#)

## SOAP

- Implements the SOAP API using Apache CXF over JAX-WS;
- Communicates in XML format;
- Requests are always in the form of documents - see [Functionality](#)<sup>[187]</sup> for specifics;
- Provides [specifications](#)<sup>[195]</sup> in WSDL format;
- Is more strict in the interpretation of input and output formats.

 [Read more about SOAP APIs](#)

## 5.2 Functionality

---

DTS Webservices provide access to [DTS Client](#)<sup>[213]</sup> features by wrapping them in standard REST or SOAP operational protocols.

### Connection and Setup

DTS Webservices fully automate the steps the Client uses for connecting to the DTS core modules and setting up for general operation. They are generated with the specific parameters (as configured) hard-coded inside them, thus requiring no initial input from the user.

DTS Webservices are enabled upon deployment, but do not immediately connect to the DTS core modules. Instead, they do so when the first request comes in. This provides more flexibility in deployment and keeps the application server more independent of the DTS core.

 **Typically, the initial request receives a "500 - Internal Server Error" reply**

This error states that the required Endpoint is not ready for use and will persist until the service has completed the initialization procedures and is connected to the respective producer. Subsequent requests will function normally. During normal operation, this same error signals that the targeted data producer is no longer available and the request cannot be serviced.

### Remote Routine Calls

Remote routines (methods, procedures, functions) included in the Webservice will be wrapped as REST or SOAP operations.

## REST

- Each routine can be wrapped as either a GET or POST call using the setting in the UI.
- **i** POST is preset and mandatory when at least one of the routine's input parameters cannot be codified within the request URL as a Query Parameter.  
Only STRING and INTEGER types can be query parameters.
- The URL component for invoking the operation is customizable.
- Routines returning streams also provide an option to customize the URL component of the **Get Records from Stream** operation.

## SOAP

- Calls will be wrapped as SOAP operations with parameters fed inside a SOAP XML request
- The URL for invoking all operations on a service is the same (i.e. the endpoint URL)
- The operation name is customizable.
- Routines returning streams also provide an option to customize the name of the **Get Records from Stream** operation.

**i** See [Access](#)<sup>196</sup> for details on URLs and Operation Naming

**i** See [Webservice Routine Details Drawer](#)<sup>86</sup> for information on interacting with Remote Routine Calls in the Web UI

## Collections and Aggregates

When Collections or Aggregates are included in a Webservice, various operations for requesting records and controlling streams become available.

**i** Aggregates behave exactly like Collections when designing a Webservice

### Open Stream with Inline Parameters

Opens a persistent stream on a collection based on Query Parameters fed through the URL

#### REST

- GET request
- Supports Query Parameters only for fields that are variants of STRING and INTEGER types
- Returns a *stream\_id* to be used for record requests
- The URL for invoking the operation is customizable

#### SOAP N/A

## Open Stream with Predicate

Opens a persistent stream on a collection based on a [Predicate](#) 239 object

- REST**
- POST request
  - Supports queries on all fields, as well as non-equality and compound queries
  - The Predicate is expected as a JSON inside the body of the request
  - Returns a `STREAM_ID` to be used for record requests
  - The URL for invoking the operation is customizable
- SOAP**
- Supports queries on all fields, as well as non-equality and compound queries
  - The Predicate is expected inside the XML SOAP Request
  - Returns a SOAP XML Response containing the `STREAM_ID` to be used for record requests
  - The operation name is customizable

## Get Records from Stream

Requests a batch of records from a given stream

- REST**
- GET request
  - Has two Query Parameters: `stream_id` and `dts_size`
  - Will return `dts_size` records from the stream identified by `stream_id` (or however many are left, if less than `dts_size` )
  - The URL for invoking the operation is customizable
- SOAP**
- Expects two parameters in the SOAP REQUEST: `stream_id` and `dts_size`
  - Will return `dts_size` records from the stream identified by `stream_id` (or however many are left, if less than `dts_size` )
  - The operation name is customizable

## Stream Has More

Queries whether there are more records to get from a given stream

- REST**
- GET request
  - Has a single Query Parameter: `stream_id`
  - Returns true or false
  - The URL for invoking the method is fixed:  
`[SERVICE_URL]/stream-has-more?stream_id=...`

- SOAP**
- Expects a single parameter in the SOAP REQUEST: *stream\_id*
  - Returns a SOAP response containing true or false for the output value
  - The operation name is fixed: **streamHasMore**

## Close Stream

Requests the closing of a given stream

- REST**
- GET request
  - Has a single Query Parameter: *stream\_id*
  - Returns **null**
  - The URL for invoking the method is fixed:  
**[SERVICE\_URL]/stream-close?stream\_id=...**

- SOAP**
- Expects a single parameter in the SOAP REQUEST: *stream\_id*
  - Returns a SOAP response containing no output
  - The operation name is fixed: **streamClose**

## Get Records with Inline Parameters

Requests a specific number of records that satisfy the given Query Parameters

- REST**
- GET request
  - Supports Query Parameters only for fields that are variants of STRING and INTEGER types
  - Aside from the collection fields, the *dto\_size* Query Parameter is also supported. It specifies how many records should be retrieved. If not included, it defaults to 1.
  - Returns the requested records
  - Is equivalent to the following sequence of operations: Open Stream With Inline Parameters -> Get Records From Stream -> Close Stream
  - The URL for invoking the operation is customizable

**SOAP** N/A

## Get Records with Predicate

- REST**
- POST request
  - Supports queries on all fields, as well as non-equality and compound queries

Requests a specific number of records that satisfy the given [Predicate](#) 239

- The Predicate is expected inside a JSON in the body of the request
- Aside from the Predicate, the `dots_size` field is also supported in the request body. It specifies how many records should be retrieved. If not included, it defaults to 1.
- Returns the requested records
- Is equivalent to the following sequence of operations: Open Stream With Predicate -> Get Records From Stream -> Close Stream
- The URL for invoking the operation is customizable

## SOAP

- Supports queries on all fields, as well as non-equality and compound queries
- The Predicate is expected inside the XML SOAP Request
- Aside from the Predicate, the `dots_size` field is also expected in the SOAP Request. It specifies how many records should be retrieved. If not included, it defaults to 1.
- Returns a SOAP XML Response containing the requested records
- Is equivalent to the following sequence of operations: Open Stream With Predicate -> Get Records From Stream -> Close Stream
- The operation name is customizable

## Get Record for Key REST

Requests a single record that is identified by a given field value (should be a primary key or unique field)

- GET request
- Supports a single Path Parameter for the Key Field, which must be a variant of STRING and INTEGER types
- Returns the requested record
- Is equivalent to the following sequence of operations: Open Stream With Inline Parameters (?key\_field=...) -> Get Records From Stream (dots\_size=1) -> Close Stream
- The URL for invoking the operation is customizable

## SOAP

- Supports a single parameter in the SOAP Request for the Key Field

- Returns a SOAP XML Response containing the requested record
- Is equivalent to the following sequence of operations: Open Stream With Predicate (eq, key\_field) -> Get Records From Stream (dts\_\_size=1) -> Close Stream
- The operation name is customizable

 See [Access](#)<sup>195</sup> for details on URLs and Operation Naming

 See [Webservice Stream Operations Drawer](#)<sup>79</sup> for information on interacting with Remote Routine Calls in the Web UI

## Topics

Each topic included in the Webservice will show including/excluding various operations and customizing their accessibility.

### Subscribe

Subscribes to the topic using the default properties and the group\_id provided as a query parameter

#### REST

- GET request
- *group\_id* is optional - if absent, a unique one will be generated for this subscription
- Returns a *stream\_id* to be used for poll requests
- The URL for invoking the operation is customizable
- All properties set for the Topic and its Connector are active for the Subscription

#### SOAP

N/A

### Subscribe with Properties

Subscribes to the topic using custom properties

#### REST

- POST request
- The *properties* and the *group\_id* are expected in the body of the request in JSON format (they are all optional)
- Returns a *stream\_id* to be used for poll requests
- The URL for invoking the operation is customizable
- The provided Properties will be added to the ones set for the Topic and its Connector and override any with the same key

	<b>SOAP</b> <ul style="list-style-type: none"> <li>• Supports queries on all fields, as well as non-equality and compound queries</li> <li>• The <i>properties</i> and the <i>group_id</i> are expected inside the XML SOAP Request</li> <li>• Returns a SOAP XML Response containing the <i>stream_id</i> to be used for record requests</li> <li>• The operation name is customizable</li> <li>• The provided Properties will be added to the ones set for the Topic and its Connector and override any with the same key</li> </ul>
<b>Poll</b>  Polls a batch of records from a given subscription stream	<b>REST</b> <ul style="list-style-type: none"> <li>• GET request</li> <li>• Has a single Query Parameter: <i>stream_id</i></li> <li>• Will return any records available for the subscription identified by <i>stream_id</i> (normally all of them, but a maximum number can be specified during subscription using the <i>properties</i> )</li> <li>• The URL for invoking the operation is customizable</li> </ul> <b>SOAP</b> <ul style="list-style-type: none"> <li>• Expects a single parameter in the SOAP REQUEST: <i>stream_id</i></li> <li>• Will return any records available for the subscription identified by <i>stream_id</i> (normally all of them, but a maximum number can be specified during subscription using the <i>properties</i> )</li> <li>• The operation name is customizable</li> </ul>
<b>Close Stream</b>  Requests the closing of a given stream	<b>REST</b> <ul style="list-style-type: none"> <li>• GET request</li> <li>• Has a single Query Parameter: <i>stream_id</i></li> <li>• Returns <b>null</b></li> <li>• The URL for invoking the method is fixed:   <code>[SERVICE_URL]/stream-close?stream_id=...</code> </li> </ul> <b>SOAP</b> <ul style="list-style-type: none"> <li>• Expects a single parameter in the SOAP REQUEST: <i>stream_id</i></li> <li>• Returns a SOAP response containing no output</li> <li>• The operation name is fixed: <b>streamClose</b></li> </ul>

**Push**

Pushes an entry to the Topic

**REST** • GET request

- Has two query parameters: *key* and *msg*
- It only supports pushing a single entry onto the topic with the key and message in string format
- The URL for invoking the operation is customizable

**SOAP**

- Expects two parameters in the SOAP REQUEST: *key* and *msg*
- It only supports pushing a single entry onto the topic with the key and message in string format
- The operation name is customizable

**Push Many**

Pushes multiple entries to the topic

**REST** • POST request

- Expects the entries as an array of objects with *key* and *msg* attributes in JSON format. For example:

```
[
  {"key": "some_key", "msg": "some_msg"},
  {"key": "another_key", "msg": "another_msg"}
]
```

- The URL for invoking the operation is customizable

**SOAP**

- Expects an array of objects (*arg0*) with *key* and *msg* sub-elements in the XML SOAP Request. For example:

```
<soapenv:Body>
  <ws:pushMany_some_topic>
    <arg0>
      <ws:key>some_key</ws:key>
      <ws:msg>some_msg</ws:msg>
    </arg0>
    <arg0>
      <ws:key>another_key</ws:key>
      <ws:msg>another_msg</ws:msg>
    </arg0>
  </ws:pushMany_some_topic>
</soapenv:Body>
```

- The operation name is customizable

 See [Access](#)  195 for details on URLs and Operation Naming

 See [Webservice Topic Details Drawer](#)<sup>[89]</sup> for information on interacting with Topics in the Web UI

## 5.3 Specification

All Webservices generated by DTS will provide standard specification files fully describing their endpoints, access patterns and data schema, regardless of the type of Application Server they are deployed on.

The flavor and access URLs of these specification files depend on the [type](#)<sup>[186]</sup> of Webservice:

WS Type	Spec Type	URL	Resource
REST	WADL	<code>[SERVICE_URL]?_wadl</code>	<a href="https://www.w3.org/Submission/wadl/">https://www.w3.org/Submission/wadl/</a>
	OPENAPI 3.0	<code>[SERVICE_URL]/openapi.j</code> <code>[SERVICE_URL]/openapi.y</code>	<a href="https://swagger.io/specification/">https://swagger.io/specification/</a>
SOAP	WSDL	<code>[SERVICE_URL]?wsdl</code>	<a href="https://www.w3.org/TR/wsdl.html">https://www.w3.org/TR/wsdl.html</a>

## 5.4 Access

This section explains how DTS Webservice functionality is accessed.

All operation URLs for DTS Webservices will have the following form:

`[APP_SERVER_URL] / [WEBSERVICE_NAME] / [MAIN_ELEMENT] / [OPERATION_ELEMENTS]`

**APP\_SERVER\_URL** The URL where the Application Server has its Webservice root

e.g.: `http://my.tomcat.srv:8080`

**WEBSERVICE\_NAME** The name given to the generated Webservice WAR

e.g.: `prj1_coolservice_DEV_v3`

**MAIN\_ELEMENT** A required imbrication element that is specified in the [Webservice Details Dialog](#)<sup>[73]</sup> before deployment

e.g.: **app**

**i** This element is required by the CXF platform and cannot be omitted, only customized

**OPERATION\_ELEMENTS** A sequence of URL path elements that points to the specific method of the specific asset to access and which also sets any necessary in-line parameters. The Asset Elements can be customized using the controls in the [Webservice Asset Details Drawer](#)<sup>[78]</sup>

Asset elements vary by the service and operation types, and we will detail them here.

## SOAP Asset Elements

A DTS SOAP service uses a single endpoint, and therefore a single URL to access all of its functionality. Details regarding operations and parameters will be included in the SOAP Request Body.

As such, the **OPERATION\_ELEMENTS** part of the URL for SOAP services is not present, the endpoint URL resolving to simply:

**[APP\_SERVER\_URL] / [WEBSERVICE\_NAME] / [MAIN\_ELEMENT]**

Example:

**http://my.tomcat.srv:8080/prj1\_coolservice\_DEV\_v3/app**

## REST Asset Elements

REST services identify operations through their specific URLs, which can also codify parameter values in the case of GET requests. We will look at each request type separately.

## REST Remote Routine Calls

The **OPERATION\_ELEMENTS** part of a REST Remote Routine Call URL is fully customizable from the [Webservice Routine Details Drawer](#)<sup>[86]</sup>.

By default, it is **[CONNECTOR\_NAME] / [NATIVE\_ROUTINE\_NAME]**

If all of the routine's inputs are supported as Query Parameters, this will be a GET request and the inputs can be fed inline.

Example:

**http://my.tomcat.srv:8080/prj1\_coolservice\_DEV\_v3/app/postgres/a\_complicated\_pro**

## REST Collection Stream Operations

The **OPERATION\_ELEMENTS** for these operations are split into two parts, both customizable in the [Webservice Stream Operations Drawer](#)<sup>[79]</sup>.

**[URL\_PREFIX] / [METHOD\_NAME]**

**URL\_PREFIX** will apply to the URLs for all operations on a given collection. Its default is **[CONNECTOR\_NAME] / [NATIVE\_COLLECTION\_NAME]**

Example (up to and including **URL\_PREFIX**):

`http://my.tomcat.srv:8080/prj1_coolservice_DEV_v3/app/postgres/nice_table`

**METHOD\_NAME** identifies the specific operation to be executed:

Operation	Default	Example
Open Stream with Inline Parameters		<code>http://my.tomcat.srv:8080/prj1_coolse app/postgres/nice_table?lastname=Smit</code>
Open Stream with Predicate	stream-with-pred	<code>http://my.tomcat.srv:8080/prj1_coolse app/postgres/nice_table/stream-with-p</code>
Get Records from Stream	stream-get	<code>http://my.tomcat.srv:8080/prj1_coolse app/postgres/nice_table/stream-get? stream_id=569caeaf-5445-4912-9047-ado &amp;ds__size=30</code>
Get Records with Inline Parameters	records	<code>http://my.tomcat.srv:8080/prj1_coolse app/postgres/nice_table/records?lastr ds__size=15</code>
Get Records with Predicate	records-with-pred	<code>http://my.tomcat.srv:8080/prj1_coolse app/postgres/nice_table/records-with-</code>
Get Record for Key	{key}	<code>http://my.tomcat.srv:8080/prj1_coolse app/postgres/nice_table/11223344</code>

## REST Topic Operations

The **OPERATION\_ELEMENTS** for these operations are split into two parts, both customizable in the [Webservice Topic Operations Drawer](#)<sup>[89]</sup>.

**[URL\_PREFIX] / [METHOD\_NAME]**

**URL\_PREFIX** will apply to the URLs for all operations on a given collection. Its default is **[CONNECTOR\_NAME]/[NATIVE\_TOPIC\_NAME]**

Example (up to and including **URL\_PREFIX**):

`http://my.tomcat.srv:8080/prj1_coolservice_DEV_v3/app/kafka/some_topic`

**METHOD\_NAME** identifies the specific operation to be executed:

Operation	Default	Example
Subscribe		<code>http://my.tomcat.srv:8080/prj1_coolservice_DEV_v3/app/kafka/some_topic?group_id=some_group_id</code>
Subscribe with Properties	subscribe-props	<code>http://my.tomcat.srv:8080/prj1_coolservice_DEV_v3/app/kafka/some_topic/subscribe-props</code>
Poll	poll	<code>http://my.tomcat.srv:8080/prj1_coolservice_DEV_v3/app/kafka/some_topic/poll?stream_id=569caeaf-5445-4912-9047-adding</code>
Push	push	<code>http://my.tomcat.srv:8080/prj1_coolservice_DEV_v3/app/kafka/some_topic/push?key=some_key</code>
Push Many	push-many	<code>http://my.tomcat.srv:8080/prj1_coolservice_DEV_v3/app/kafka/some_topic/push-many</code>

## REST Generic Stream Operations

Stream closing and querying whether a stream has more records are operations which are independent of the specific collection the stream is opened on, as they only require the *stream\_id*.

As a result, they do not use the **COLLECTION\_PATH\_PREFIX** above and are derived from the base service URL instead.

Operation	Default	Example
Stream Has More	stream-has-more	<code>http://my.tomcat.srv:8080/prj1_coolservice_DEV_v3/app/stream-has-more?stream_id=569caeaf-5445-4912-9047-adding</code>
Close Stream	stream-with-pred	<code>http://my.tomcat.srv:8080/prj1_coolservice_DEV_v3/app/stream-close?stream_id=569caeaf-5445-4912-9047-adding</code>

## 5.5 Integration

DTS produces standard REST/SOAP Webservices which can be used in any desired integration patterns. This section discusses the prerequisites for deploying and running the services in your Application Server as well as some ideas regarding Security and Load Balancing.

- [Application Server](#)<sup>199</sup>
- [Security and Load Balancing](#)<sup>201</sup>
- [Logging](#)<sup>203</sup>

### 5.5.1 Application Server

#### Preparation

DTS produces Webservices in the form of WebARchives, which contain all the necessary libraries for the service to run. As such, most Application Servers should be able to deploy and run the DTS Webservices.

To prepare the Application Server for DTS Webservices, the environment variables which direct specify the DTS connection point need to be set within the server's configuration files. These variables are:

- DTS\_REDIS\_HOST\_NAME** The host name for the DTS Redis Server
- DTS\_REDIS\_PORT** The port for the DTS Redis Server
- DTS\_SYNC\_INIT** [true/false] Whether the Webservice client will initialize the project synchronously (i.e. delay the first request until initialization is done and a response can be provided). Default is false.

Different Application Servers will have different ways of setting these variables. Here are some common examples:

Application Server	OS	Instructions
Tomcat	Linux	<ul style="list-style-type: none"> <li>• edit or create the <code>[CATALINA_HOME]/bin/setenv.sh</code> file</li> <li>• add the following: <pre># DTS export DTS_REDIS_HOST_NAME=[...]</pre> </li> </ul>

		<pre>export DTS_REDIS_PORT=[...]</pre> <ul style="list-style-type: none"> <li>• save the file and restart Tomcat</li> </ul>
	Windows	<ul style="list-style-type: none"> <li>• edit or create the [CATALINA_HOME]\bin\setenv.bat file</li> <li>• add the following: <pre># DTS set DTS_REDIS_HOST_NAME=[...] set DTS_REDIS_PORT=[...]</pre> </li> <li>• save the file and restart Tomcat</li> </ul>
JBoss/Wildfly	Linux	<ul style="list-style-type: none"> <li>• edit the [JBOSS_HOME]/bin/standalone.sh file</li> <li>• add the following (preferably at the beginning): <pre># DTS export DTS_REDIS_HOST_NAME=[...] export DTS_REDIS_PORT=[...]</pre> </li> <li>• save the file and restart JBoss/Wildfly</li> </ul>
	Windows	<ul style="list-style-type: none"> <li>• edit the [JBOSS_HOME]\bin\standalone.bat file</li> <li>• add the following (preferably at the beginning): <pre># DTS set DTS_REDIS_HOST_NAME=[...] set DTS_REDIS_PORT=[...]</pre> </li> <li>• save the file and restart JBoss/Wildfly</li> </ul>

 For instructions regarding other Application Servers, please consult the vendor's documentation.

 If the Application Server is configured together with DTS using docker-compose, the environment variables can usually be set there.

 Being based on the [Client Java Library](#)<sup>[257]</sup>, Webservices also respond to all other environment variables specified there.



 Additionally, the DTS Webservice responds to certain environment variables for configuring specific logging parameters (see [Webservice Logging](#)<sup>[203]</sup>).

 If the security scheme is enabled in the DTS cluster, the Webservice must also implement it. See [Security Setup](#)<sup>[252]</sup> for details.

## Exploded Deployment

DTS Webservices need to be able to dynamically load libraries at runtime. While all the required libraries are bundled inside a service WAR, these need to be accessible after deployment, so the WAR must be exploded.

Depending on the Application Server and deployment method, this will either be done automatically, or may require some extra steps. Here are some common examples:

Application Server	DTS Deployment	Exploding
Tomcat	SCP/Samba (automatic copy)	Done automatically by the Application Server
Tomcat	WAR (web console / manual copy)	Done automatically by the Application Server on WAR deployment
Tomcat	TomcatHTTP	Done automatically by the Application Server
JBoss/Wildfly	SCP/Samba (automatic copy)	Not done automatically - exploded flag must be set in the <a href="#">App Server Parameters</a> <sup>[97]</sup> to instruct DTS to do it
JBoss/Wildfly	WAR (web console / manual copy)	The WAR file must be unzipped into a directory with the same name (including ".war") and the directory must be copied into the deployments folder of the Application Server   <b>Starting with version 12, Wildfly provides Web Console features to explode deployed WARs.</b>
JBoss/Wildfly	JBossCLI	Done automatically by DTS   <b>This deployment method requires Wildfly 12 or higher</b>

 See [App Server Parameters](#)<sup>[97]</sup> for details on all the supported deployment methods.

### 5.5.2 External Security and Load Balancing

DTS Webservices do not provide an external Security Layer themselves and, while the distribution of requests within DTS is load balanced between the various data producer instances, another layer of load balancing may be desirable at the entry point of certain services. In these situations, we recommend using DTS Webservices in conjunction with a dedicated reverse proxy and load balancer, (e.g. Nginx, HAProxy, etc.).

In these section we will discuss when using such a layer may be desirable and what the advantages of using such a solution would be.

## Security

An entry-point security layer is desirable if you need to provide access to the DTS services from outside your organization, or through the internet without network virtualization. It is also essential if you wish to setup a credentials barrier for accessing the services.

The advantages of using a reverse proxy in this case are:

- The solution will use web-standard https with your desired encryption flavor and depth (SSL, TLS, etc.)
- It will ensure encryption of all of the services' traffic, not just the message bodies (like document based encryption methods would)
- It provides an easy way to define and configure credential barriers for various services with a lot of flexibility regarding login security
- Can provide complementary features like redirecting unencrypted requests, DDoS attack protection, etc.

## Load Balancing

An entry-point load balancer is desirable if high levels of traffic are expected on certain services and the initial request handling is likely to become a bottleneck. For such situations, multiple instances of a DTS webservice can be deployed on multiple application servers, and an external load balancer can be used to direct requests to the instances.

Advantages:

- Greatly increases the intake capacity of individual webservices
- Allows various criteria to be used for traffic routing
- Allows the persistence of sessions and streams

## All-In-One

Many reverse proxy / load balancing solutions provide all of the features required for securing and routing traffic from a single instance, are very efficient and conform to Web standards, thus providing the ideal approach for such situations.

### 5.5.3 Logging

DTS Webservices can be created with special logging parameters in order to create customized logs that can be processed by other tools for reporting, statistics, or other purposes.

**i** For the moment, customized logging is only compatible with Tomcat application servers.

## Separate Logging

By default, DTS Webservices will log messages in the application server's main output log (e.g. [tomcat]/logs/catalina.out).

To turn on customized Webservice logging, the [GUI Controller](#)<sup>[211]</sup> must be started with the **DTS\_WS\_LOGGING\_PATH** environment variable set.

```
DTS_WS_LOGGING_PATH=/opt/tomcat/logs/dts
```

This directory path set here will be the target for the logs created by all DTS Webservices on the machine or container running the application server where the services are deployed.

**i** Only Webservices generated while the Webapp is connected to a GUI Controller with **DTS\_WS\_LOGGING\_PATH** set will be capable of customized logging.

**i** If the Application Server is running in a Docker container, it is a good idea to map this path to the host for easy access to the logs.

The webservices will create daily log files in this directory with the following names:

```
[service-name].YYYY-MM-DD.log
```

When separate/customized logging is enabled, the log messages created by DTS Webservices will use the format:

```
[YYYY-MM-DD HH:MM:SS] [LEVEL]: [MESSAGE] [STACK_TRACE]
```

- **[LEVEL]** represents the logging level of the message (INFO, WARNING, ERROR, DEBUG, SEVERE, etc.)
- **[MESSAGE]** represents the actual message being logged (created either by the Application Server or the Webservice)
- **[STACK\_TRACE]** represents a JVM stack trace to help diagnose errors.

## Customization Parameters

Certain parameters can be customized for Webservice logging. This is done by setting environment variables either globally on the machine or container running the application server, or within the startup scripts of the application server itself.

Like all DTS modules, the Webservices respond to the **DTS\_DEBUG\_LOGGING** environment variable which, if set to true, will increase the verbosity of the output for debugging purposes.

**DTS\_DEBUG\_LOGGING=true**

**i** Debug logging can increase output volume substantially. If log files become too large, please check if DTS\_DEBUG\_LOGGING is on and needed.

**i** Unlike other logging customization options, debug logging can be enabled for any Webservice, regardless of how it was created.

DTS Webservices can also be configured to log all transactions they handle by setting the **DTS\_TRANSACTION\_LOGGING** environment variable to true.

**DTS\_TRANSACTION\_LOGGING=true**

**i** Transaction logging will fork the input and output streams of request and response message in order to perform its function, so it may have a perceivable performance impact in certain scenarios.

Transaction logging can be further customized by setting the **DTS\_TRANSACTION\_FORMAT** environment variable to a specific tokenized string, thus instructing the logger to create messages that include certain information.

The following tokens are currently available for the transaction format:

Token	Name	Description
%a	REMOTE_HOST_TOKEN	The address of the remote host that invoked the transaction
%u	REMOTE_USER_TOKEN	The username that invoked the transaction
%t	TIMESTAMP_TOKEN	The timestamp of the request receipt
%T	RETURN_TIME_TOKEN	The number of milliseconds it took for the service to resolve the request
%m	METHOD_TOKEN	The type of http method that was invoked (GET, POST, etc.)



## 5.6 Limitations

---






- Remote routine calls with Geometry parameters (or complex parameters containing geometries) can be mapped for Webservice access only if the respective parameters are of specific geometry types (i.e. Point, LineString, Polygon, etc.). For example, Oracle has no mechanism to specify the geometry type for a parameter (they will all be declared as SDO\_GEOMETRY), so DTS will not be able to map such a routine into a Webservice. The workaround for this limitation is to modify such routines to receive geometry parameters in some other format (user types, coordinate arrays, etc.). Routine outputs are not affected by this limitation - generically defined geometries are fully supported in routine output structures.



# Technical Guide

## 6 Technical Guide

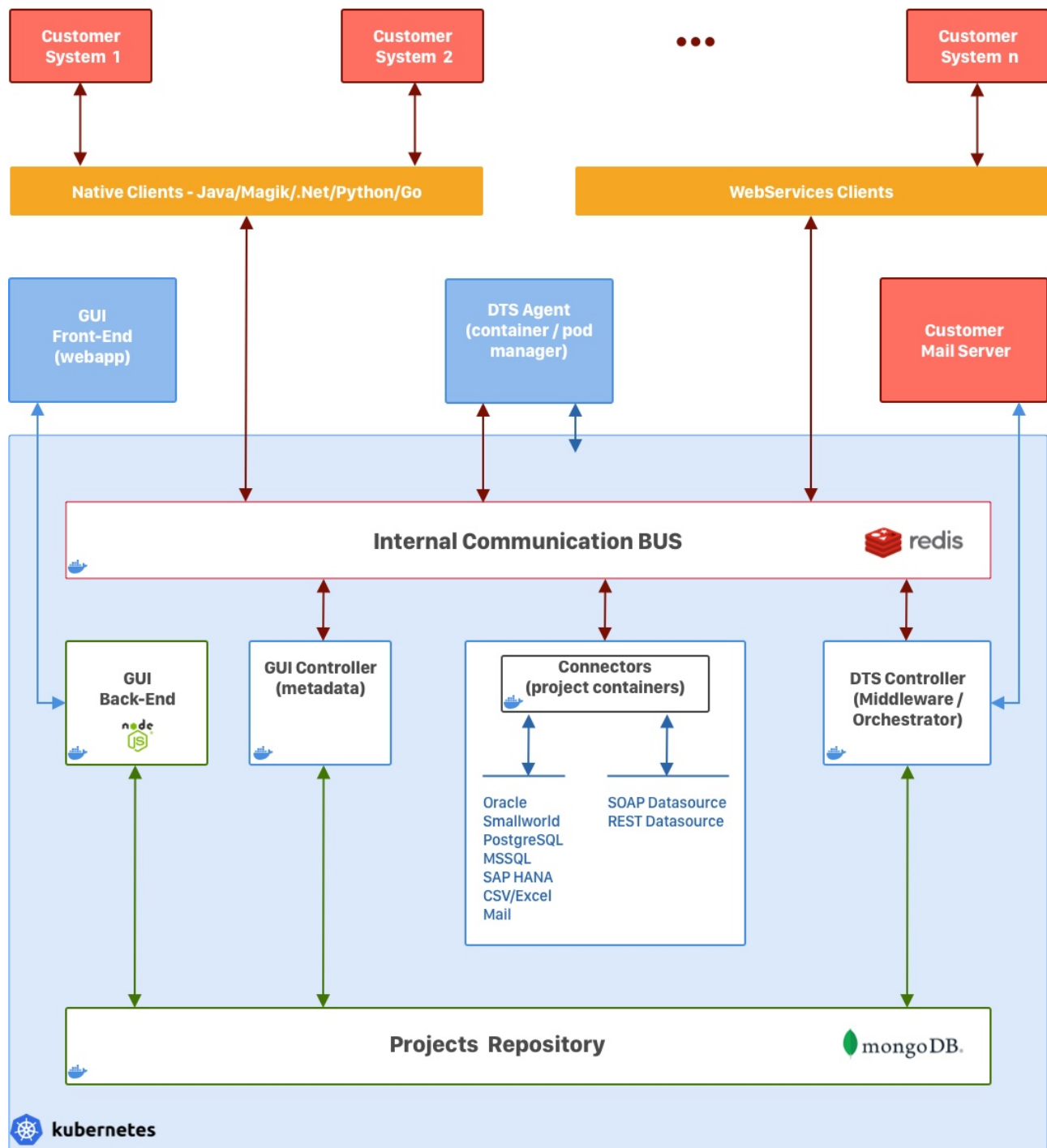
This section provides some technical details about the DTS software stack.

- [Architecture](#)  208
- [Communication](#)  218
- [Types](#)  218
- [Geometry](#)  222
- [Predicate](#)  239

### 6.1 Architecture

---

DTS functions as a distributed middleware layer based on multiple interacting modules.



Depending on their type and purpose, modules can be deployed in containers, stand-alone, or as part of other applications.

The types of modules employed by DTS are:

Container	Description	In Cluster?
<b>Internal Communication Bus (Redis)</b>	The Redis messaging system used by DTS	Yes
<b>Projects Repository (MongoDB)</b>	The Mongo database used internally by DTS	Yes
<a href="#">GUI</a> <sup>217</sup> <b>Front-End</b>	The in-browser part of the GUI webapp	No
<a href="#">GUI</a> <sup>217</sup> <b>Back-End</b>	The JS webapp which serves as the GUI Engine	Yes
<a href="#">GUI Controller</a> <sup>211</sup>	Houses the back end of the GUI	Yes
<a href="#">DTS Controller</a> <sup>210</sup>	The main middleware module	Yes
<a href="#">Connectors</a> <sup>212</sup> <b>(Producers)</b>	Generic data connector which is instantiated as specific producers	Yes
<a href="#">Aggregators</a> <sup>215</sup>	Independent module which serves aggregation requests for a specific project	Yes
<a href="#">DTS Agent</a> <sup>216</sup>	The proxy module that allows DTS to control the cluster it's deployed in.	No
<b>Native</b> <a href="#">Clients</a> <sup>213</sup>	Bespoke DTS clients, based on the Client libraries	No
<b>Web Service</b> <a href="#">Clients</a> <sup>213</sup>	REST/SOAP webservice automatically generated by DTS to act as clients	No

 Please see the [Communication](#)<sup>218</sup> section for information on the communication paradigms used by DTS

## 6.1.1 Controller

The DTS Controller is the central module of the system during general operation.

It is tasked with the following responsibilities:

- Registers and manages all [Producer](#)<sup>212</sup> and [Client](#)<sup>213</sup> sessions in the system.
- Routes requests and ensures load balancing.
- Manages data streams.

- Manages internal communications channels.
- Manages in-use projects.
- Orders boot-up and scaling of producer sessions.
- Monitors producer sessions for responsiveness.
- Manages event notifications.
- Serves as the security hub.

The Controller is the first module to be started and its continued operation is critical to the entire system. No registration or data requests can be resolved without the Controller.

In a standard deployment, the Controller is a single instance housed inside a container.

The Controller communicates with other modules as follows:

- Internal Communications Bus (Redis): All communication with [Producers](#)<sup>[212]</sup>, [Clients](#)<sup>[213]</sup> and the [Agent](#)<sup>[216]</sup> is done via this bus. Moreover, the Controller generates the specific communication channels on the ICB.
- Mongo API over TCP/IP: The Controller reads all project data from the Projects Repository using this method.
- SMTP: The Controller uses this protocol to send notifications regarding various events.

## 6.1.2 GUI Controller

The DTS GUI Controller serves as the metadata hub for the system and represents the central module for GUI / project editing operations.

It is tasked with the following responsibilities:

- Receives and resolves metadata requests from the [GUI](#)<sup>[217]</sup>.
- Creates instances of various [Connectors](#)<sup>[118]</sup> within its own classpath as Local Metadata Providers to facilitate metadata request resolution.
- Registers and manages remote Connectors as Remote Metadata Providers to facilitate metadata request resolution.
- Generates web service clients bytecode and packages them for deployment at the GUI's request.

- Ensures security in communication with remote providers.

The GUI Controller is critical for all project manipulation actions as it the module that ultimately resolves all requests for metadata and other GUI operations.

In a standard deployment, the GUI Controller is a single instance housed inside a container.

The GUI Controller communicates with other modules as follows:

- Internal Communications Bus (Redis): The GUI Controller uses the ICB to communicate with the [GUI](#)<sup>[217]</sup>, the [Agent](#)<sup>[216]</sup> and any connected Remote Metadata Providers ([Connectors](#)<sup>[118]</sup>).
- Mongo API over TCP/IP: The Controller reads all project data from the Projects Repository using this method and fills in metadata and mappings as needed.

## 6.1.3 Producer

The DTS Producer serves as the data request resolution module of the system.

It is tasked with the following responsibilities:

- Morphs into a specific [Connector](#)<sup>[118]</sup> by including the necessary libraries into its own classpath.
- Registers with the [Controller](#)<sup>[210]</sup> and implements the specific connector configuration it receives from it.
- Receives and resolves data and execution requests routed to it by the Controller using the Connector libraries and provides responses directly to the requesting [Clients](#)<sup>[213]</sup>.
- For certain Connectors (e.g. Smallworld), the Producer must also serve as a Remote Metadata Provider, register with the [GUI Controller](#)<sup>[211]</sup>, and resolve metadata requests.
- Implements communication security protocols.

DTS Producers can also be seen as instances of [Connectors](#)<sup>[118]</sup>. While a Connector, as defined in the Project, represents a logical connection to a data source, a producer is a physical process that implements a connector definition.

In a standard deployment there can be multiple producer instances running inside separate containers. Depending on the scaling configuration and the current load of the system, there can also be multiple active producers for each logical Connector definition.

The Producer communicates with other modules as follows:

- Internal Communications Bus (Redis): The Producer uses the ICB to receive requests routed by the Controller, to respond to Clients and to provide status reports to the Controller.
- Other APIs: The Producer also uses specific APIs for each Connector type to access the respective datasource (e.g. Oracle - ojdbc).

## 6.1.4 Client

The DTS Client is a blanket term for the library modules that provide client access to the system from various environments.

It is tasked with the following responsibilities:

- Encapsulates DTS client operations in simple functions relative to the environment.
- Registers with the [Controller](#)<sup>[210]</sup>.
- Facilitates the opening of Projects.
- Encapsulates requests in environmentally relevant forms and addresses inside the system.
- Packages results received from the queried [Producer](#)<sup>[212]</sup> in a standard format for the environment.
- Offers simple calls for creating, managing and consuming data streams.
- Implements communication security protocols.

Since the Client is a library which would generally be called by other routines in the environment, it has no specified deployment paradigm.

The only Client library currently available for development is for Java.

The Client communicates with other modules as follows:

- Internal Communications Bus (Redis): The ICB is used by the Client to address requests to the [Controller](#)<sup>[210]</sup> for routing and to receive responses from [Producers](#)<sup>[212]</sup>.

DTS provides out-of-the-box Client environment implementations for:

- [REST/SOAP](#)<sup>[214]</sup>

- [Smallworld](#) <sup>215</sup>

## 6.1.4.1 Webservice Client

DTS Webservice Clients are implementations of the Java [Client](#) <sup>213</sup> Library that are generated by the system upon request for a given Project and set of resources.

A Webservice Client is tasked with the following responsibilities (in addition to the ones of the base library):

- Provides access to the configured resources via SOAP or REST requests.
- Is packaged as an easily deployable Web ARchive (WAR).
- Is compatible with various Application Servers (e.g. Tomcat, JBoss, Wildfly, etc.)
- Encapsulates arguments and results of targeted routines to ensure they are callable using the configured protocol, but also translate to the native datasource being accessed.
- Provides SOAP/REST operations for creating, managing and consuming data streams.
- Uses standard data formats (XML/JSON) for inputs and outputs.
- Uses standard representations for complex objects (e.g. GeoJson for geometries).
- Can be multiplied and clustered for load balancing.

A Webservice Client must be deployed inside an Application Server. The deployment of the Application Server itself is irrelevant to the system as long as it has access to the Internal Communications Bus.

Multiple identical Webservice Clients can be deployed simultaneously and requests can be routed to them using 3rd party load balancing software, like Nginx.

The Webservice Client itself provides no external security features. To implement authentication, authorization or encryption on the resulting Webservices, an extra layer is required. A tool like Nginx is recommended for this as well.

 For more information on the operations and usage of DTS Webservices, please see the dedicated [section](#) <sup>186</sup>.

#### 6.1.4.2 Smallworld Client

The DTS Smallworld Client is an implementation of the Java [Client](#)<sup>[213]</sup> Library specific for the GE Smallworld environment.

It is tasked with the following responsibilities:

- Provides access to all resources in a given project from the Smallworld Magik console, or via Magik code.
- Encapsulates arguments and results of targeted routines within native Magik objects.
- Provides Magik code methods to create, manage and consume data streams.
- Uses native Magik objects as inputs and produces native Magik objects as outputs.
- Translates complex objects to and from relevant Magik forms (e.g. `pseudo_geometry` for geometry).

The Smallworld Client is part of the [Smallworld Connector](#)<sup>[159]</sup> fileset and must be deployed for use inside a Smallworld image (only swaf is required).

 For more information on how to use the Smallworld Client, please see the [dedicated section](#)<sup>[176]</sup>.

#### 6.1.5 Aggregator

The DTS Aggregator serves as the data request resolution module for aggregate data requests.

 For an overview of this feature, see [Aggregation](#)<sup>[242]</sup>.

It is tasked with the following responsibilities:

- Registers with the [Controller](#)<sup>[210]</sup> and implements the Aggregate definitions it receives from it.
- Also registers with the Controller as a [Client](#)<sup>[213]</sup> to make requests for specific aggregate elements.
- Receives and resolves aggregate data requests routed to it by the Controller using the client connection to pass connector-specific requests onwards and provides responses directly to the requesting [Clients](#)<sup>[213]</sup>.
- Implements communication security protocols.

DTS Aggregators are directly linked to Projects (i.e. an Aggregator can only serve a single project's Aggregates, but will serve all of them). The Aggregator behaves both as a [Producer](#)<sup>[212]</sup> as it receives requests and provides responses to the respective Clients, and as a Client as it generates requests for specific connectors in order to build the Aggregate records.

In a standard deployment there can be multiple Aggregator instances running inside separate containers. Depending on the scaling configuration and the current load of the system, there can also be multiple active Aggregators for each running DTS Project.

The Aggregator communicates with other modules as follows:

- Internal Communications Bus (Redis): The Aggregator uses the ICB to receive requests routed by the Controller, to respond to Clients, to make connector-specific data requests, and to provide status reports to the Controller.

## 6.1.6 Agent

The DTS Agent serves as the interface between the system and the cluster that runs its main module containers.

It is tasked with the following responsibilities:

- Can interact with Kubernetes as well as Docker-only deployments of DTS.
- Starts and stops containers/pods at the request of the [Controller](#)<sup>[210]</sup>.

The system can function without an Agent running, but it will not be able to self-regulate - all [producers](#)<sup>[212]</sup> will need to be started and stopped manually or by other means, and automatic restarting of unresponsive producers will not be possible.

In order to control the cluster, the Agent must run outside of it, generally as a stand-alone application or wrapped in a service. It can function on the same machine as the cluster controller or on a remote one with SSH access.

The Agent communicates with other modules as follows:

- Internal Communications Bus (Redis): The ICB is used by the Agent to receive commands from the [Controller](#)<sup>[210]</sup> and report on its status.
- SSH/SCP: SSH is used by the Agent to control the deployment cluster if not running on the same machine.

## 6.1.7 GUI

The DTS GUI is the user interface of the system.

It is tasked with the following responsibilities:

- Provides the means for users to design Projects in a friendly graphical environment.
- Provides information about and control over certain internal operations.
- Allows users to publish Projects into use of the other systems.
- Allows users to create, alter and deploy [Webservice Clients](#)<sup>214</sup>.

The GUI is a JavaScript Web Application with a Front-End running in the user's browser and the Back-End using NodeJS and typically running in a single container in the main deployment cluster.

Multiple GUI deployments functioning simultaneously are supported, but scenarios where this is necessary are limited.

The GUI communicates with other modules as follows:

- HTTP is used to communicate between the Front-End and the Back-End of the GUI.
- Internal Communications Bus (Redis): The ICB is used by the GUI Back-End to communicate with the GUI Controller for resolution of requests.

## 6.1.8 CLI

Command reference coming soon.

In the meantime please use the help commands:

[Linux]

```
$ ./dts-cli.sh --help
```

[Windows]

```
> dts-cli.bat --help
```

## 6.2 Communication

---

DTS uses a number of communication paradigms to pass information between various modules as well as to and from external actors:


- The **Internal Communication Bus (ICB)** is the main communication channel of the system. Most of the first-party modules communicate using this bus, and it can also be used by third party clients that adhere to the API. In actuality, it represents a standard Redis server.
- **Mongo API over TCP/IP** is used by all the modules which need to access the Projects Repository to perform the necessary operations.
- **HTTP** is used by the GUI Front-End to communicate to the GUI Back-End.
- **SMTP** is used between the GUI Controller and the Customer Mail Server.
- **SSH/SCP** is used by the GUI Controller to directly deploy web services into certain Application Servers and by the Agent to control the deployment cluster remotely.
- **Other APIs** are used by the various Connectors to access the specific datasources (e.g. ojdbc, ACPT, etc.). These generally also function over TCP/IP.

### Ports

DTS deployments can vary depending on network topology, resources and usage. However, certain functions will always require remote access on configured ports.

The required ports are:

- The GUI Webapp port (configurable) must be opened to allow remote users to interact with the web GUI.
- The ICB (Redis) port (configurable) must be opened to allow remote clients to access the system.
- The SSH port (22) must be opened to allow installation and maintenance operations.

 For more information on deployment models and specific ports, please see [Deployment](#) <sup>16</sup>.

## 6.3 Types

---

DTS accesses various data sources with various type architectures and needs to provide a middle layer to facilitate translation of data between all the native types.

This section lists all of the data types defined in the DTS middleware (we'll simply call them DTS Types) and how they relate to generic Java client types, as well webservice client XSD types and OpenAPI types.

DTS Type	Java Type	XSD Type	OpenAPI Type
STRING	java.lang.String	xs:string	string
INT8	java.lang.Byte	xs:byte	integer
INT16	java.lang.Short	xs:short	integer
INT32	java.lang.Integer	xs:int	integer
INT64	java.lang.Long	xs:long	integer
UNSIGNED_INT8	java.lang.Short	xs:short	integer
UNSIGNED_INT16	java.lang.Integer	xs:int	integer
UNSIGNED_INT32	java.lang.Long	xs:long	integer
UNSIGNED_INT64	java.math.BigInteger	xs:short	integer
BIG_INTEGER	java.math.BigInteger	xs:integer	integer
FLOAT	java.lang.Float	xs:float	number
DOUBLE	java.lang.Double	xs:double	number
DECIMAL	java.math.BigDecimal	xs:decimal	number
BOOLEAN	java.lang.Boolean	xs:boolean	boolean
DATE	java.time.LocalDate	xs:date	string
DATE_TIME	java.time.Instant	xs:dateTime	string
TIME	java.time.Instant	xs:time	string
DTS_GEOMETRY	com.alloy.dts.type.json. geojson.GeoJson	[custom]*	[custom]**
DTS_POINT_GEOMETRY	com.alloy.dts.type.json. geojson.GeoJsonPoint	[custom]*	[custom]**
DTS_LINE_GEOMETRY	com.alloy.dts.type.json. geojson.GeoJsonLineString	[custom]*	[custom]**
DTS_AREA_GEOMETRY	com.alloy.dts.type.json. geojson.GeoJsonPolygon	[custom]*	[custom]**
DTS_ANNOTATION_GEOMETRY	com.alloy.dts.type.json. geojson.GeoJsonPoint	[custom]*	[custom]**
DTS_MULTIPPOINT_GEOMETRY	com.alloy.dts.type.json. geojson.GeoJsonMultiPoint	[custom]*	[custom]**
DTS_MULTILINE_GEOMETRY	com.alloy.dts.type.json. geojson.GeoJsonMultiLineString	[custom]*	[custom]**
DTS_MULTIAREA_GEOMETRY	com.alloy.dts.type.json.	[custom]*	[custom]**

	geojson.GeoJsonMultiPolygon		
DTS_GEOGRAPHY	com.alloy.dts.type.json. geojson.GeoJson	[custom]*	[custom]**
DTS_PREDICATE	com.alloy.dts.model. DTSPredicate	[custom]***	[custom]***

**i (\*) Geometry Types have custom defined XSD Types, which are fully described in the DTS Service's [WxDL file](#)<sup>[195]</sup>.**

If the particular DTS service is a SOAP service, objects in these fields will only have the geoJsonString field populated with a JSON representation of the respective Java Type (see [Geometry Structure](#)<sup>[223]</sup>).

**i (\*\*) Geometry Types have custom defined OpenApi Types which are fully described in the DTS Service's [OpenApi file](#)<sup>[195]</sup>.**

This description will actually describe the respective Java Type in JSON format (see [Geometry Structure](#)<sup>[223]</sup>).

**i (\*\*\*) Predicates are represented in in XSD and OpenAPI as custom types, describing the structure of the Java Object in XML or JSON format.**

Read more about Predicate structure [here](#)<sup>[239]</sup>.

**i** Read more about XSD Types [here](#).

**i** Some OpenAPI Types will have format restrictions to reflect the original DTS Type. More info [here](#).

**i** To see how the DTS Types relate to a specific data source's native types, please see the Types section of the respective [Connector](#)<sup>[118]</sup>

## 6.4 Streams

DTS Streams are constructs that allow the transfer of data objects or records from available resources iteratively or in batches. They differ from record sets in that a stream does not contain any data records. Instead it is a token that permits requesting record sets and automatically advances as they are delivered. They are very much equivalent to Database Result Sets or File Output Streams in scope and operation. Indeed, a DTS Stream may actually use one or more Result Set(s) or Output Stream(s) to source the data objects it provides. Its purpose is to wrap such data source constructs in a form that is friendly to remote interaction.

### Streamable Resources

DTS provides streaming access to the following types of resources:

<b>Collections / Tables / Views</b>	Data from any storage construct accessible through DTS connectors can be streamed.
<b>Stream Result Routines</b>	Certain routines are considered to provide results in the form of Streams  e.g.: Table-Valued Functions, methods returning DTS record streams, etc.
<b>Aggregates</b>	DTS Aggregates can be streamed from as if they were a simple collection.
<b>Topics</b>	Upon subscription to a Topic, a DTS is created. Polling the topic can only be done using this stream.

## Stream Lifecycle and Operations

DTS Streams have a simple lifecycle, defined by the available operations:

<b>Create</b>	<b>Open</b>	Open a stream to a storage or aggregate resource. Can optionally use a DTSPredicate to perform a specific query. If no predicate is provided, the entire resource will be streamed (subject to any static filters put in place).
	<b>ExecuteStream Call</b>	Execute a Routine whose result is a stream.
	<b>Subscribe</b>	Subscribe to a Topic to consume it via a stream.
	<b>i</b> Regardless of how a Stream is created, the result is an Active Stream, represented by a Stream ID (in UUID format).	
<b>Active</b>	<b>GetRecords</b>	Requests a batch of records on the stream. Must provide a size for the batch. If enough records remain, then that number of records is returned, otherwise as many as are left.
	<b>HasMore</b>	Queries whether there are more records on the stream.
	<b>Poll</b>	Similar to GetRecords but wrapped for Topic use -> it does not take a specific size.
	<b>i</b> While a Stream is active, it takes up a slot in the Active Streams of the DTS Product Licence.	
<b>Close</b>	<b>Close</b>	Closes the stream, and frees associated resources.

**i** Active Streams that have had no operation performed on them for 2 minutes will time out and close automatically.

## Implementations

DTS Streams always adhere to the above specification, but the way they are effectively accessed depends on the Client API or Implementation.

Currently, the following are available:

- [Webservice Client](#)  187
- [Smallworld/Magik Client](#)  176
- [Java Client API](#)  257
- [.NET Client API](#)  266
- [Raw/JSON API](#)  274




## 6.5 Geometry

DTS uses a custom extension of the GeoJson standard format for encoding geometries internally (hereafter referred to as **DTSGeoJson**). This is also the format in which DTS encodes output and expects input geometry data for Redis and Web Service clients.

DTS endeavors to translate the various geometry formats provided by the supported datasources to and from DTSGeoJson with as much fidelity as possible. However, DTSGeoJson is also fully backwards compatible with the standard GeoJson format and ensures that clients which are not "aware" of the DTSGeoJson extension are still able to seamlessly communicate geometry data via DTS using reasonably accurate approximations.

This section details the additions DTSGeoJson makes to the standard GeoJson and how approximations work.

For details on the standard GeoJson format, please see [The GeoJson Specification](#).

- [Structure](#)  223
- [Corrections](#)  224
- [Examples](#)  226

## 6.5.1 Structure

Here is the general structure of a DTSGeoJson with all the fields it uses:

```
{
  "type": "",
  "coordinates": [],
  "orientation": [],
  "cs": "",
  "dimensions": 2,
  "corrections": [],
  "extras": {},
  "geoJsonString": ""
}
```

### Geometry data only

Please note that the structure above represents the content of the "geometry" field in a full GeoJson "Feature" object. DTS only uses this part, as the other feature elements are not relevant to the geometry and cannot be related to pure geometry structures in the various systems DTS connects.

<b>type</b>	Unchanged from the standard specification. DTS supports all GeoJson geometry types: Point, Multipoint, LineString, MultiLineString, Polygon, MultiPolygon and GeometryCollection. This field must be populated.
<b>coordinates</b>	Represents the coordinates of the geometry in the same format as the standard specification. If the geometry contains elements that are not supported by the standard specification (e.g. arcs, splines, etc.), the <b>coordinates</b> will contain a "flattened" approximation (a series of straight line segments), while the exact definition of the sector will be stored in <b>corrections</b> . This ensures that clients which only support the standard format can get a close approximation of these geometries from the expected field, while clients that support the DTSGeoJson extras can substitute the approximated sectors with the exact version. This field must be populated.
<b>orientation(s)</b>	Is only present for Point (orientation) and MultiPoint (orientations) and encodes the directions points are facing (e.g. building placements). It is encoded as a single coordinate for Points and as multiple coordinates for MultiPoints. The coordinate encoding is the same as in the coordinates field. Non-oriented points will have orientations with all ordinates = 0.0, or simply lack the field.
<b>cs</b>	Specifies the SRID of the coordinate system the geometry is defined in. This field is not required.

<b>dimensions</b>	Specifies the dimensionality of the geometry. Supported values are 2 and 3, but 3D geometries with certain sector types are not supported for approximation (see <a href="#">Known Limitations</a> <sup>[322]</sup> ). If left unpopulated, the value will be inferred from the <b>coordinates</b> .
<b>corrections</b>	Stores the exact representation of curve sectors that are not supported by the GeoJson standard specification. If not populated or empty, the <b>coordinates</b> are considered to be the exact representation of the geometry. The format of the corrections is detailed in a <a href="#">separate section</a> <sup>[224]</sup> .
<b>extras</b>	Stores extra information the geometry structure may contain which is only relevant to certain data sources (e.g. annotation text, justification, etc.). It presents as a key-value map (string-string) and the values are as provided/required by the datasource.
<b>geoJsonString</b>	A String representation of the entire DTSGeoJson object (in JSON format). This field is present as a workaround for including DTSGeoJson geometry representations in SOAP XML messages and only has relevance in that context. In any other circumstances it will be absent and/or ignored.

## 6.5.2 Corrections

DTSGeoJson corrections are encoded as an array of **Line Solutions**, each of which describes a sector's exact representation and its place within the [geometry structure's coordinates](#)<sup>[223]</sup>. A Line Solution must represent a continuous curve.

This is the structure of a Line Solution:

```
{
  "startIndex": 0,
  "length": 0,
  "exactRepresentation": [],
  "lineType": ""
}
```

<b>startIndex</b>	Represents the total index of the geometry structure's coordinates where the corrected sector starts. The total index represents the index across sub-arrays down to coordinate level.
<b>length</b>	Represents the length of the approximation in the geometry structure's coordinates that this correction replaces.

**exactRepresentation** an array of double[], each of which represents a coordinate. Each double[] must contain the same number of double values (ordinates), which can only be 2 or 3, however certain 3D corrections are not supported at this time (see [Known Limitations](#)<sup>322</sup>). These coordinates will be interpreted and translated to datasource-specific geometry encodings according to the **lineType**.

**lineType** Represents the type of curve this representation encodes. Possible values are:

Value	Interpretation
LINE_STRING	The coordinates represent a series of straight line segments, as they would in the geometry structure's coordinates. This correction type is redundant, but exists for specialized clients which can pass exact geometry representations using only the corrections field.
CP_ARCS	<p>The coordinates represent a series of Center Point Circle Arcs. Each arc is encoded by 3 coordinates [<b>start</b>, <b>middle</b>, <b>end</b>]. The middle coordinate can be any point on the arc. In geometries created by DTS, the middle coordinate is always the exact middle of the arc.</p> <p>The end coordinate of an arc also serves as the start coordinate of the next arc, so the total number of coordinates will be <b>2*N_ARCS+1</b>.</p> <p><b>⚠ Please note that elliptical / tangent arcs cannot be encoded with this lineType and need to be defined as NURBS.</b></p>
NURBS	<p>The coordinates represent a Non-Uniform Rational B-Spline as follows:  <code>[ [ D, N, K ], [ knot1, ..., knotK ], [ controlPoint1 ], ..., [ controlPointN ] ]</code></p> <p>Where:</p> <p><b>D</b>: the degree of the NURBS</p> <p><b>N</b>: the number of control points</p> <p><b>K</b>: the number of knots</p> <p><b>knot1, ..., knotK</b>: the knots (K double values)</p> <p><b>[controlPoint1], ..., [controlPointN]</b>: the weighted control points (N arrays of double values containing the coordinate for the respective control point and the weight [x, y, z, w])</p> <p><b>i A correctly defined NURBS will agree with <math>K = D + N + 1</math></b></p>
RECTANGLE	The coordinates represent the opposite corners of a rectangle. Serves as shorthand for the equivalent polygon which would use 5 coordinates instead of 2.

<b>CIRCLE</b>	The coordinates represent a circle using 2 coordinates: [ [centerCoord], [anyOtherPointOnCircle] ]
---------------	---

**i** When feeding geometry data to a system via DTS using DTSGeoJson, it is possible to set exact representations in the corrections field and not calculate any approximations for the coordinates field. For this to work, the targeted connector must be capable of using the DTSGeoJson extensions and the coordinates field must contain some minimal placeholder values that will be replaced by the exact corrections upon processing.

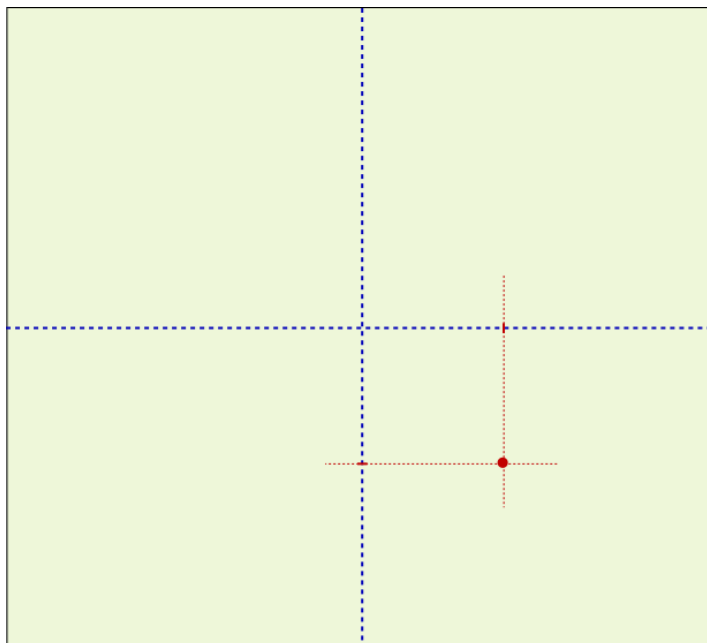
### 6.5.3 Examples

In this section we will explore some examples of geometries and how they would be encoded as DTSGeoJsons.

- [Simple Point](#)<sup>227</sup>
- [Oriented Point](#)<sup>227</sup>
- [Multi Point](#)<sup>228</sup>
- [Annotation](#)<sup>228</sup>
- [Simple Line String](#)<sup>229</sup>
- [Arc Line String](#)<sup>229</sup>
- [Compound Line String](#)<sup>230</sup>
- [NURBS](#)<sup>231</sup>
- [Multi Line String](#)<sup>232</sup>
- [Polygon](#)<sup>233</sup>
- [Compound Polygon](#)<sup>234</sup>
- [Rectangle](#)<sup>235</sup>
- [Circle](#)<sup>236</sup>
- [Multi Polygon](#)<sup>237</sup>
- [Geometry Collection](#)<sup>238</sup>

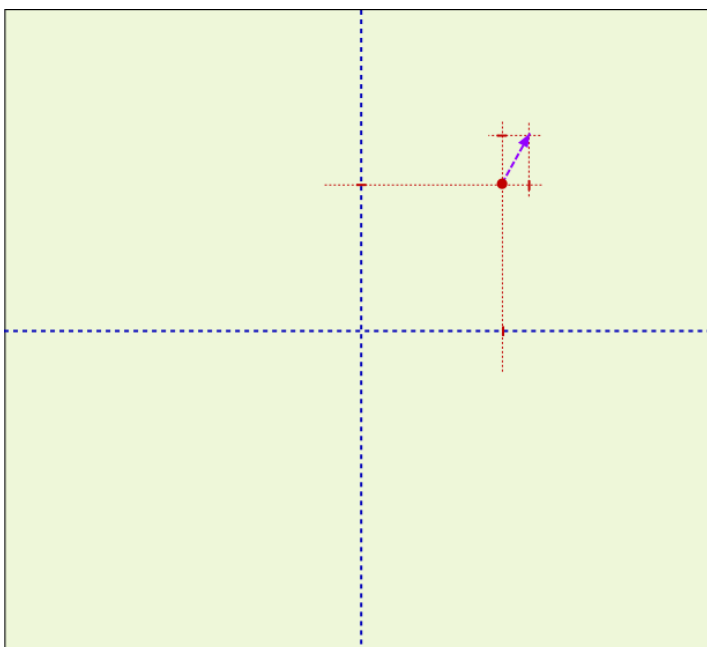
## 6.5.3.1 Simple Point

```
{
  "coordinates": [100.0, -100.0],
  "orientation": [0.0, 0.0],
  "type": "Point",
  "cs": "89000",
  "dimensions": 2,
  "extras": {}
}
```



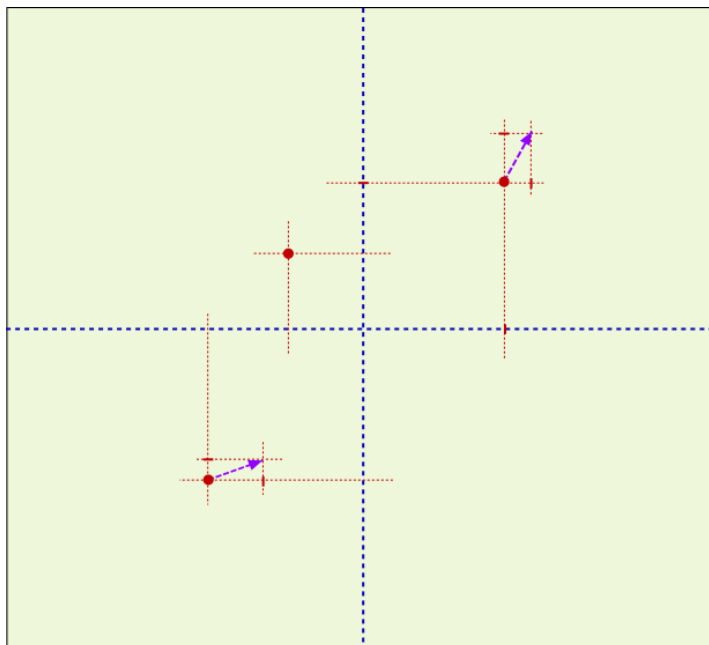
## 6.5.3.2 Oriented Point

```
{
  "coordinates": [100.0, 100.0],
  "orientation": [1.0, 2.0],
  "type": "Point",
  "cs": "89000",
  "dimensions": 2,
  "extras": {}
}
```



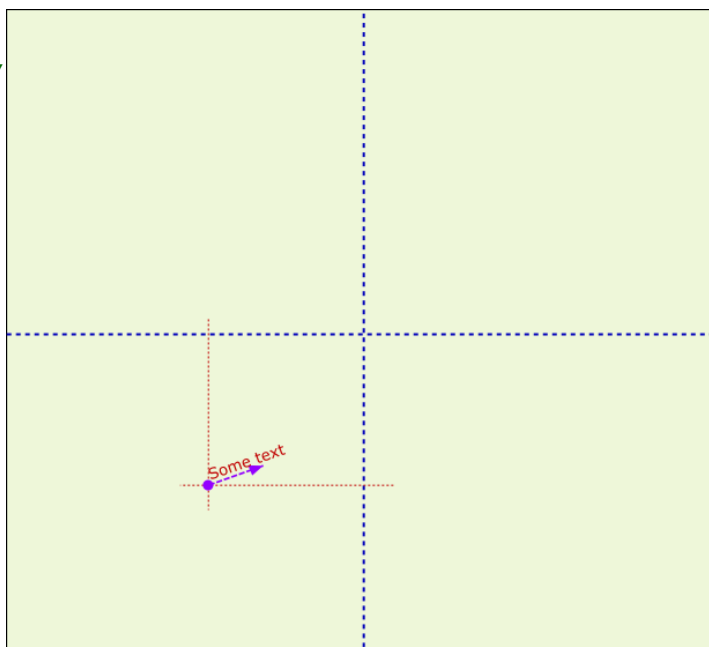
## 6.5.3.3 Multi Point

```
{
  "coordinates": [
    [100.0, 100.0],
    [-50.0, 50.0],
    [-100.0, -100.0]
  ],
  "orientations": [
    [1.0, 2.0],
    [0.0, 0.0],
    [3.0, 1.0]
  ],
  "type": "MultiPoint",
  "cs": "89000",
  "dimensions": 2,
  "extras": {}
}
```



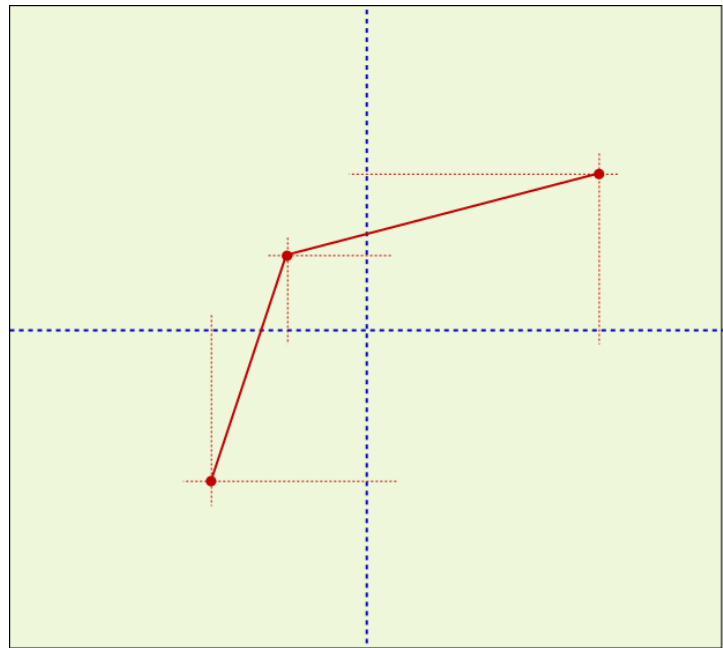
## 6.5.3.4 Annotation

```
{
  "coordinates": [-100.0, -100.0],
  "orientation": [3.0, 1.0],
  "type": "Point",
  "cs": "89000",
  "dimensions": 2,
  "extras": {
    "text": "Some text",
    "justification": "22"
  }
}
```



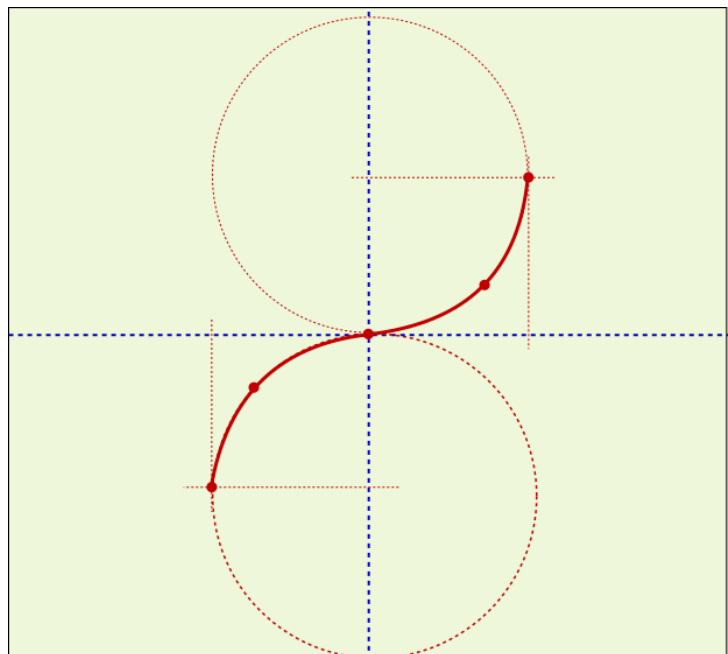
## 6.5.3.5 Simple Line String

```
{
  "coordinates": [
    [-100.0, -100.0],
    [-50.0, 50.0],
    [150.0, 100.0]
  ],
  "type": "LineString",
  "cs": "89000",
  "dimensions": 2,
  "extras": {}
}
```



## 6.5.3.6 Arc Line String

```
{
  "coordinates": [
    [-100.0, -100.0],
    [-99.879436, -95.093266],
    // ... intermediate steps
    [-74.094415, -32.844757],
    [-70.71, -29.29],
    [-67.155242, -25.905584],
    // ... intermediate steps
    [-4.906733, -0.120563],
    [0.0, 0.0],
    [4.906733, 0.120563],
    // ... intermediate steps
    [67.155242, 25.905584],
    [70.71, 29.29],
    [74.094415, 32.844757],
    // ... intermediate steps
    [99.879436, 95.093266],
    [100.0, 100.0]
  ],
  "type": "LineString",
  "cs": "89000",
}
```



```

"dimensions":2,
"corrections": [
  {
    "startIndex": 0,
    "length": 65,
    "exactRepresentation": [
      [-100.0, -100.0],
      [-70.71, -29.29],
      [0.0, 0.0],
      [70.71, 29.29],
      [100.0, 100.0]
    ],
    "lineType": "CP_ARCS"
  }
],
"extras":{}
}

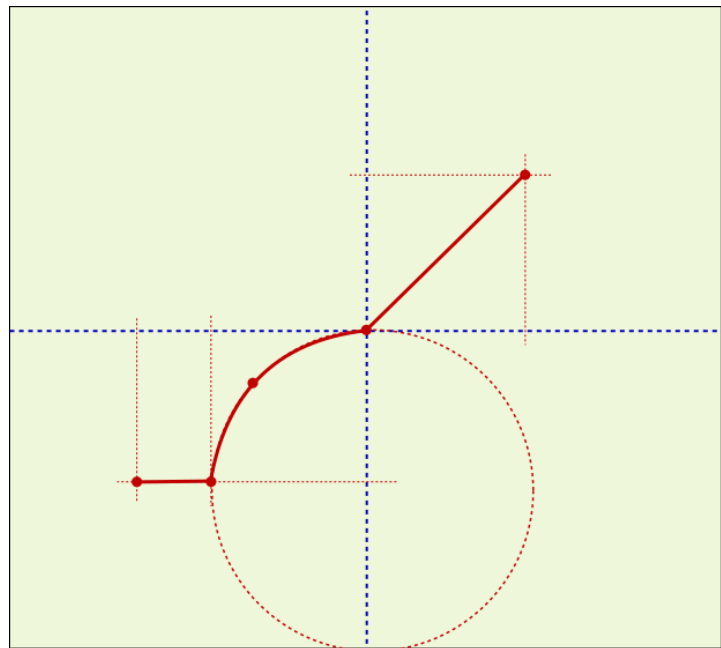
```

## 6.5.3.7 Compound Line String

```

{
  "coordinates":[
    [-150.0, -100.0],
    [-100.0,-100.0], // index=1
    [-99.879436,-95.093266],
    // ... intermediate steps
    [-74.094415,-32.844757],
    [-70.71,-29.29],
    [-67.155242,-25.9055844],
    // ... intermediate steps
    [-4.906733,-0.120563],
    [0.0,0.0], // index= 33
    [100.0,100.0]
  ],
  "type":"LineString",
  "cs":"89000",
  "dimensions":2,
  "corrections": [
    {
      "startIndex": 1,
      "length": 33,
      "exactRepresentation": [
        [-100.0, -100.0],
        [-70.71, -29.29],
        [0.0, 0.0]
      ]
    }
  ],
}

```



```

        "lineType": "CP_ARCS"
    }
],
"extras":{}
}

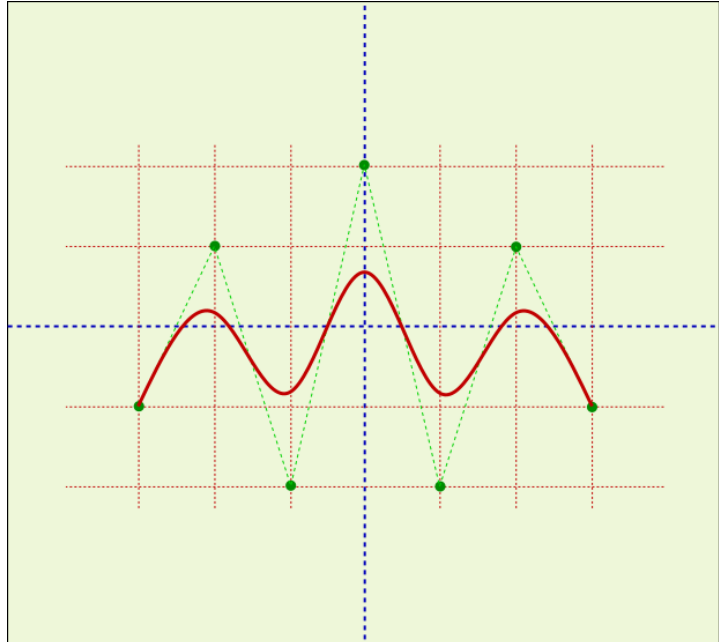
```

## 6.5.3.8 NURBS

```

{
  "coordinates":[
    [-150.0,-50.0],
    [-143.334820,-37.425259],
    [-136.967818,-26.881417],
    // ... intermediate steps
    [136.967818,-26.881417],
    [143.334820,-37.425259],
    [150.0,-50.0]
  ],
  "type":"LineString",
  "cs":"89000",
  "dimensions":2,
  "corrections":[
    {
      "startIndex":0,
      "length":89,
      "lineType":"NURBS",
      "exactRepresentation":[
        [3.0,7.0,11.0], // D, N,
        [ // knots:
          0.0,0.0,0.0,0.0,
          0.25,0.5,0.75,
          1.0,1.0,1.0,1.0
        ],
        // control points:
        [-150.0,-50.0,1.0],
        [-100.0,50.0,1.0],
        [-50.0,-100.0,1.0],
        [0.0,100.0,1.0],
        [50.0,-100.0,1.0],
        [100.0,50.0,1.0],
        [150.0,-50.0,1.0]
      ]
    }
  ],
  "extras":{}
}

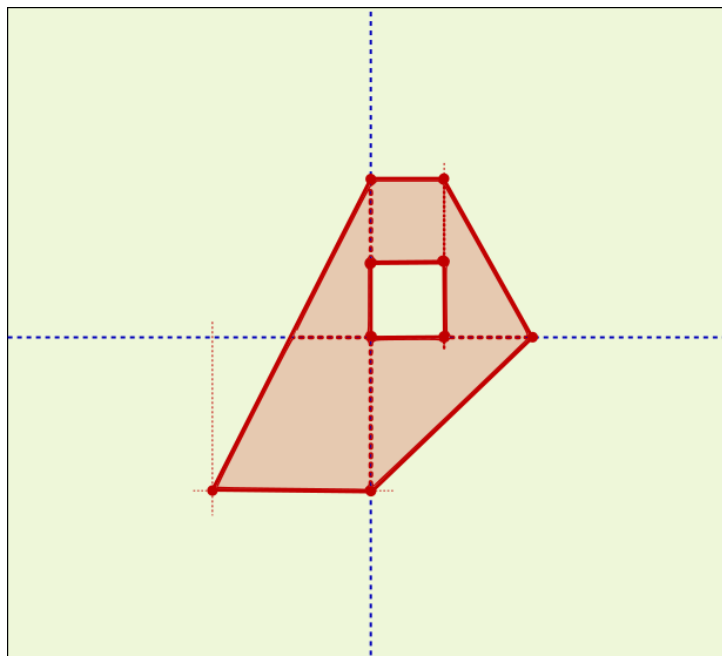
```





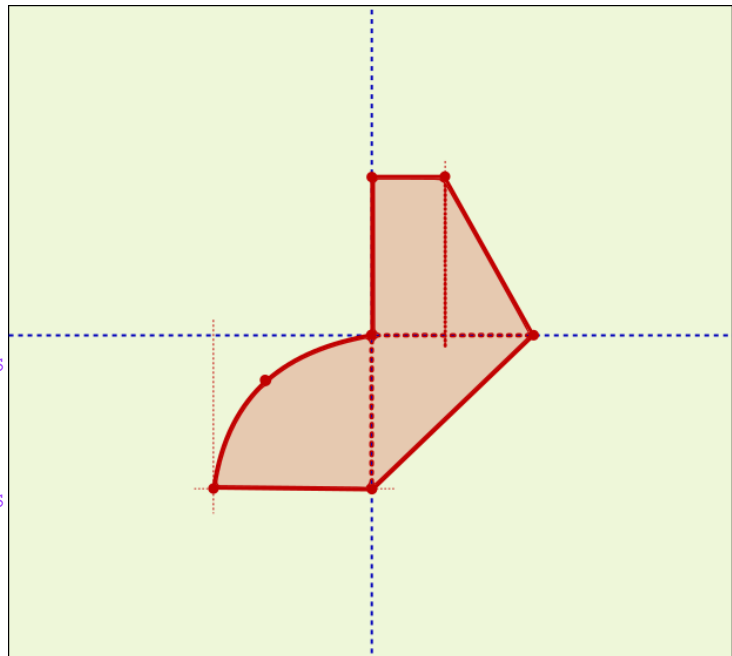
## 6.5.3.10 Polygon

```
{
  "coordinates": [
    [ // outer boundary:
      [-100.0, -100.0],
      [0.0, -100.0],
      [100.0, 0.0],
      [50.0, 100.0],
      [0.0, 100.0],
      [-100.0, -100.0]
    ],
    [ // hole:
      [0.0, 0.0],
      [50.0, 0.0],
      [50.0, 50.0],
      [0.0, 50.0],
      [0.0, 0.0]
    ]
  ],
  "type": "Polygon",
  "cs": "89000",
  "dimensions": 2,
  "corrections": [],
  "extras": {}
}
```



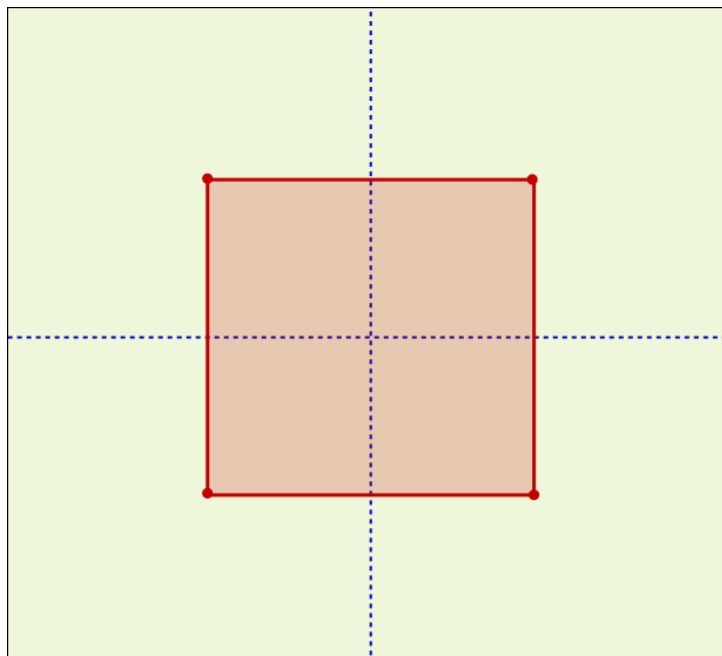
## 6.5.3.11 Compound Polygon

```
{
  "coordinates": [
    [
      [-100.0, -100.0],
      [0.0, -100.0],
      [100.0, 0.0],
      [50.0, 100.0],
      [0.0, 100.0],
      [0.0, 0.0],
      [-4.906733, -0.120563],
      // ... intermediate steps
      [-67.155242, -25.9055844],
      [-70.71, -29.29],
      [-74.094415, -32.844757],
      // ... intermediate steps
      [-99.879436, -95.093266],
      [-100.0, -100.0]
    ]
  ],
  "type": "Polygon",
  "cs": "89000",
  "dimensions": 2,
  "corrections": [
    {
      "startIndex": 5,
      "length": 33,
      "exactRepresentation": [
        [0.0, 0.0],
        [-70.71, -29.29],
        [-100.0, -100.0]
      ],
      "lineType": "CP_ARCS"
    }
  ],
  "extras": {}
}
```



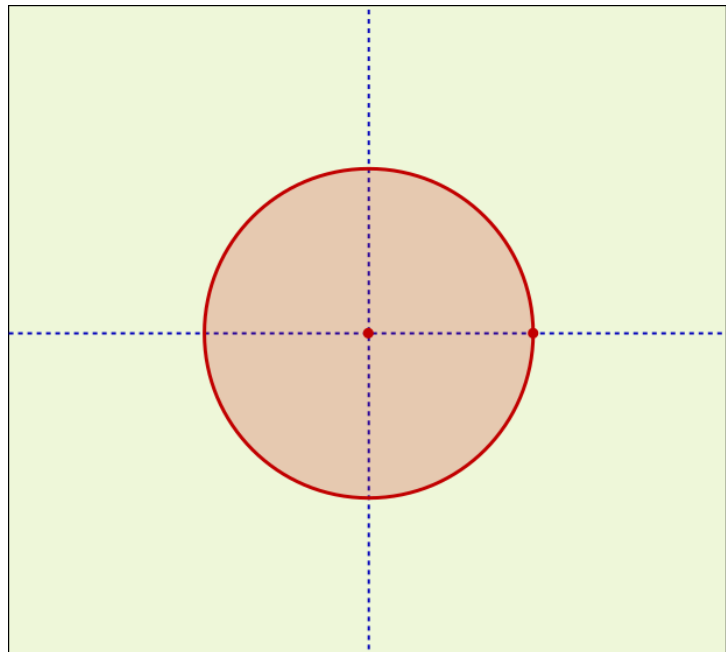
## 6.5.3.12 Rectangle

```
{
  "coordinates": [
    [
      [-100.0, -100.0],
      [100.0, -100.0],
      [100.0, 100.0],
      [-100.0, 100.0],
      [-100.0, -100.0]
    ]
  ],
  "type": "Polygon",
  "cs": "89000",
  "dimensions": 2,
  "corrections": [
    {
      "startIndex": 0,
      "length": 5,
      "exactRepresentation": [
        [-100.0, -100.0],
        [100.0, 100.0]
      ]
    }
  ],
  "lineType": "RECTANGLE"
},
"extras": {}
}
```



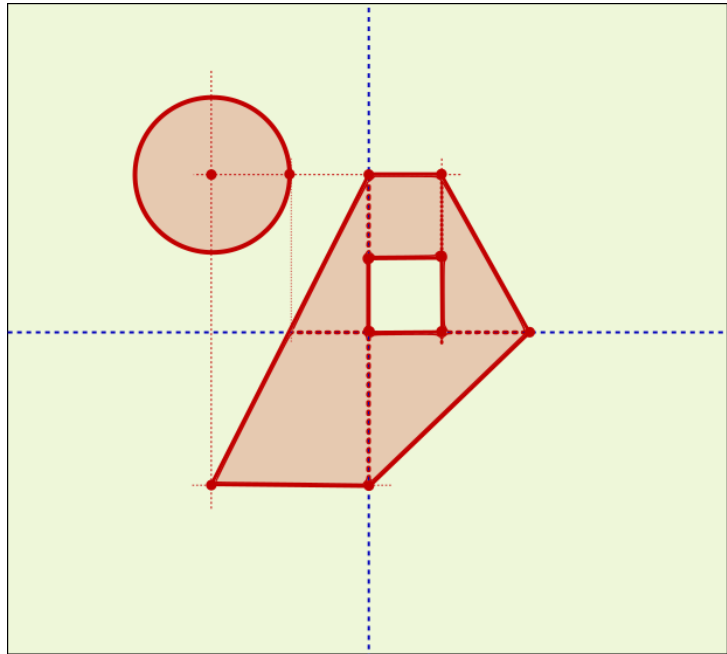
## 6.5.3.13 Circle

```
{
  "coordinates": [
    [
      [100.0,0.0],
      [99.518472,9.801714],
      // ... intermediate steps
      [9.801714,99.518472],
      [0.0,100.0],
      [-9.801714,99.518472],
      // ... intermediate steps
      [-99.518472,9.801714],
      [-100.0,0.0],
      [-99.518472,-9.801714],
      // ... intermediate steps
      [-9.801714,-99.518472],
      [0.0,-100.0],
      [9.801714,-99.518472],
      // ... intermediate steps
      [99.518472,-9.801714],
      [100.0,0.0]
    ]
  ],
  "type": "Polygon",
  "cs": "89000",
  "dimensions": 2,
  "corrections": [
    {
      "startIndex": 0,
      "length": 65,
      "exactRepresentation": [
        [0.0, 0.0],
        [100.0, 0.0]
      ],
      "lineType": "CIRCLE"
    }
  ],
  "extras": {}
}
```



## 6.5.3.14 Multi Polygon

```
{
  "coordinates": [
    [ poly1:
      [ // outer boundary:
        [-100.0, -100.0],
        [0.0, -100.0],
        [100.0, 0.0],
        [50.0, 100.0],
        [0.0, 100.0],
        [-100.0, -100.0]
      ],
      [ // hole:
        [0.0, 0.0],
        [50.0, 0.0],
        [50.0, 50.0],
        [0.0, 50.0],
        [0.0, 0.0]
      ]
    ],
    [ // poly2 (circle):
      [
        [-50.0, 100.0],
        // ... intermediate step
        [-100.0, 150.0],
        // ... intermediate step
        [-150.0, 100.0],
        // ... intermediate step
        [-100.0, 50.0],
        // ... intermediate step
        [-50.0, 100.0]
      ]
    ]
  ],
  "type": "MultiPolygon",
  "cs": "89000",
  "dimensions": 2,
  "corrections": [
    {
      "startIndex": 11,
      "length": 65,
      "exactRepresentation": [
        [-100.0, 100.0],
        [-50.0, 100.0]
      ],
      "lineType": "CIRCLE"
    }
  ]
}
```



```

    ],
    "extras": {}
  }
}

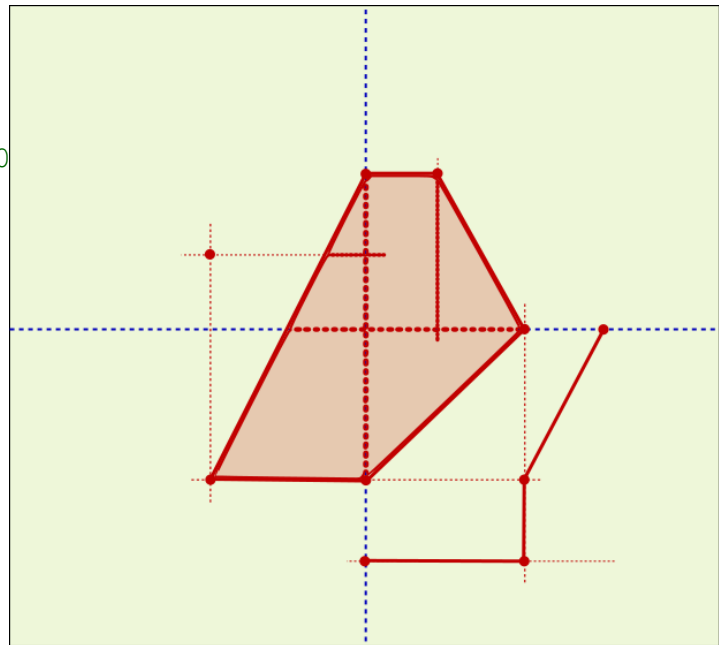
```

## 6.5.3.15 Geometry Collection

```

{
  "type": "GeometryCollection",
  "geometries": [
    { // point
      "coordinates": [-100.0, 50.0],
      "orientation": [0.0, 0.0],
      "type": "Point",
      "cs": "89000",
      "dimensions": 2,
      "extras": {}
    },
    { // polygon
      "coordinates": [
        [
          [-100.0, -100.0],
          [0.0, -100.0],
          [100.0, 0.0],
          [50.0, 100.0],
          [0.0, 100.0],
          [-100.0, -100.0]
        ],
        [
          [0.0, -150.0],
          [100.0, -150.0],
          [100.0, -100.0],
          [150.0, 0.0]
        ]
      ],
      "type": "Polygon",
      "cs": "89000",
      "dimensions": 2,
      "corrections": [],
      "extras": {}
    },
    { // line
      "coordinates": [
        [0.0, -150.0],
        [100.0, -150.0],
        [100.0, -100.0],
        [150.0, 0.0]
      ],
      "type": "LineString",
      "cs": "89000",
      "dimensions": 2,
      "extras": {}
    }
  ]
}

```



```
    ]
}
```

## 6.6 Predicate

Predicates are data structures that allow complex queries to be passed through DTS towards various [Connectors](#)<sup>[118]</sup> in order to extract records or create streams. All record requests through DTS will internally resolve to Predicates, though the direct interface may sometimes wrap the request in simpler parameters for ease of use.

Predicates are also used by DTS to define fundamental Collection Filters and to design relationships within [Aggregates](#)<sup>[242]</sup>.

### Structure

DTS Predicates can be found in various endpoints either as Java Objects or JSON or XML data structures. We will explore their structure in JSON format for simplicity:

```
{
  "operatorName": "",
  "attributeName": "",
  "attributeValue": "",
  "negated": false,
  "leftPredicate": {
    // another predicate
  },
  "rightPredicate": {
    // another predicate
  }
}
```

<b>operatorName</b>	The name of the operator to be used either between attributeName and attributeValue, or between leftPredicate and rightPredicate, depending on the operator.
<b>attributeName</b>	The name of the attribute that the Predicate will create a clause for
<b>attributeValue</b>	The comparative value for attributeName
<b>negated</b>	Whether the Predicate should be negated
<b>leftPredicate</b>	Used in compound Predicates. Specifies the first predicate that should be combined using operatorName.

**rightPredicate** Used in compound Predicates. Specifies the second predicate that should be combined using operatorName.

**i** The types of all of the Predicate's fields is String, with the exception of "negated", which is a boolean

To pass a number-type attributeValue, into the predicate, simply use the standard String representation (e.g.: 34 -> "34", 100.29 -> "100.29", etc.)

**i** Certain expressions can also be passed through the attributeName and attributeValue fields, in order to provide another layer of query customization

The exact limits of this feature depend on the targeted system. You can find a general description of it below and individual implementation limits in the "Limitations" section of each [Connector](#)<sup>118</sup> category.

## Operators

The operators, specified using the operatorName field of the Predicate, control the actual function of the Predicate. The following operators are known to DTS:

operatorName	Function	Details
eq	Equals	Creates a clause which checks if <b>attributeName</b> <i>equals</i> <b>attributeValue</b>
like	Matches	Creates a clause which checks if <b>attributeName</b> <i>matches</i> <b>attributeValue</b> (may be a regex or some other kind of matching pattern, depending on the targeted system)
gt	Greater Than	Creates a clause which checks if <b>attributeName</b> <i>is greater than</i> <b>attributeValue</b>
gteq	Greater Or Equal Than	Creates a clause which checks if <b>attributeName</b> <i>is greater than or equals</i> <b>attributeValue</b>
lt	Lesser Than	Creates a clause which checks if <b>attributeName</b> <i>is lesser than</i> <b>attributeValue</b>
lteq	Lesser or Equal Than	Creates a clause which checks if <b>attributeName</b> <i>is lesser than or equals</i> <b>attributeValue</b>
and	And	Combines <b>leftPredicate</b> and <b>rightPredicate</b> using an AND operator
or	Or	Combines <b>leftPredicate</b> and <b>rightPredicate</b> using an OR operator

**i** The negated field value determines whether the entire result of the Predicate should be negated, thus making the creation of complementary operators possible:

eq + negated = not-Equals  
 like + negated = not-Matches  
 and + negated = NAND  
 or + negated = NOR

## Passing Expressions

### Examples

```
{
  "operatorName": "eq",
  "attributeName": "SURNAME",
  "attributeValue": "Smith",
  "negated": false
}
```

This Predicate will create a clause that will be true for all records in the target collection where the value of the **SURNAME** field *is exactly equal to* **"Smith"**

```
{
  "operatorName": "like",
  "attributeName": "SURNAME",
  "attributeValue": "%son",
  "negated": false
}
```

This Predicate will create a clause that will be true for all records in the target collection where the value of the **SURNAME** field *ends with* **"son"** (e.g. "Johnson", "Robertson", etc.)

**i** The wildcard character % is common in SQL-type queries

```
{
  "operatorName": "lteq",
  "attributeName": "AGE",
  "attributeValue": "34",
  "negated": false
}
```

This Predicate will create a clause that will be true for all records in the target collection where the value of the **AGE** field *is less than or equal to* **34**

```
{
  "operatorName": "eq",
  "attributeName": "CITY",
  "attributeValue": "New York",
  "negated": true
}
```

This Predicate will create a clause that will be true for all records in the target collection where the value of the CITY field *is not equal to* **"New York"**

```
{
  "operatorName": "and",
  "negated": false,
  "leftPredicate": {
    "operatorName": "eq",
```

This Predicate will create a clause that will be true for all records in the target collection where the value of the **SURNAME** field *is exactly equal*

```

        "attributeName": "SURNAME",
        "attributeValue": "Smith",
        "negated": false
    },
    "rightPredicate": {
        "operatorName": "lteq",
        "attributeName": "AGE",
        "attributeValue": "34",
        "negated": false
    }
}

{
    "operatorName": "and",
    "negated": false,
    "leftPredicate": {
        "operatorName": "eq",
        "attributeName": "CITY",
        "attributeValue": "New York",
        "negated": true
    },
    "rightPredicate": {
        "operatorName": "like",
        "attributeName": "TO_CHAR(DOB)",
        "attributeValue": "%-12-%",
        "negated": false
    }
}

```

to **"Smith"** and the value of the **AGE** field is less than or equal to **34**

This Predicate will create a clause that will be true for all records in the target collection where the value of the **CITY** field is not equal to **"New York"** and the value of the **DOB** field after transforming it to a String, contains **"-12-"** within it (i.e. people that aren't from New York, but were born in December).

**i** This is a simple use of expressions in the **attributeName** field, where the Oracle function **TO\_CHAR** was used to create a match clause on a **DATE**-type column

## 6.7 Aggregation

Aggregation is the feature that allows DTS to seamlessly unify resources from disparate data sources based on predefined relationships and present them for consumption via [Streams](#)<sup>[220]</sup>.

The configuration unit (project artifact) for Aggregation is called an Aggregate (stored in the [AGGREGATE collection](#)<sup>[306]</sup>). Multiple Aggregates can be defined for any DTS project using the [Aggregates Page](#)<sup>[59]</sup> in the Web UI.

The DTS component responsible for resolving aggregate data requests is called an [Aggregator](#)<sup>[215]</sup>.

## Aggregate Elements

Each well-defined Aggregate must contain the following elements:

<b>Main Source</b>	The starting point for the Aggregate. The main stream will be created on this data source and its results will be the root for requesting data from the secondary sources.
<b>Secondary Sources</b>	The other data sources for the Aggregate. Queries performed on these sources depend on results from their parent sources (as defined through the Relationships).
<b>Relationships</b>	Simple clauses that define equality relationships between parent and child sources. Each is modeled by feeding the value of an attribute from a parent's result as the value of another attribute in a child's query.
<b>Filters</b>	Each source (main or secondary) can have a fundamental filter. <ul style="list-style-type: none"> <li>• On Main Sources, the filter will restrict what base records will pass through to those matching the filter.</li> <li>• On Secondary Collection Sources, the filter will inject static/constant parameters in the queries used to find data pertaining to an aggregate record, with the Relationships providing the variable parameters.</li> <li>• On Secondary Routine Sources, the filter is used to set constant values for routine inputs.</li> </ul>

Each source has a list of **Attributes** (items which can be included in the Aggregate result) and **Query Parameters** (items which can be used to create relationship queries targeting the respective source).

**i Sources can be Collections or Routines.**

Collection have their fields serve as both Attributes and Query Parameters.

Routines have their inputs (arguments) serve as QueryParameters and their outputs (results) serve as Attributes.

**i Each source (main or secondary) can also have a fundamental filter defined, which will restrict results from that source to the ones that match the filter.**

**i The list of Attributes from each source that will be included in the aggregation process is fully customizable, as are the names with which the respective attributes will be tagged in the resulting records.**

**⚠ It falls to the user to ensure that no two Attributes across an Aggregate have the same name. The name customization feature mentioned above must be used when duplicate Attribute names are present.**

## Rules and Limitations

The following set of rules governs how aggregates function and it is not expected to change:

- Only Sources that have been included in the project are available for use in Aggregates. For Collection Sources, only the fields that are included in the project will be available for use in Aggregates.
- Each Secondary Source must have at least one relationship. Secondary Sources without relationships are ignored.
- Only the Attributes from a parent source that have been included in the Aggregate are available for relationships with child sources.
- All Query Parameters from a child source can be used in its relationships, regardless of their inclusion status in the aggregate.
- Circular references cannot be created using Aggregate Relationships.

The following set of limitations exists for the current version, but is subject to possible changes in future releases:

- The Main Source of an Aggregate must be a Collection.
- Secondary Sources cannot be Routines that return Streams.
- Only the Main Source's Query Parameters can be used to make queries on the Aggregate.
- All relationships are treated like "Outer Joins". If no results are found in a particular source for a particular query, the resulting aggregate record will still be delivered, but with the values for that source's attributes missing.
- If more than one result is generated by a particular relationship query, only the first record is used.

## Example

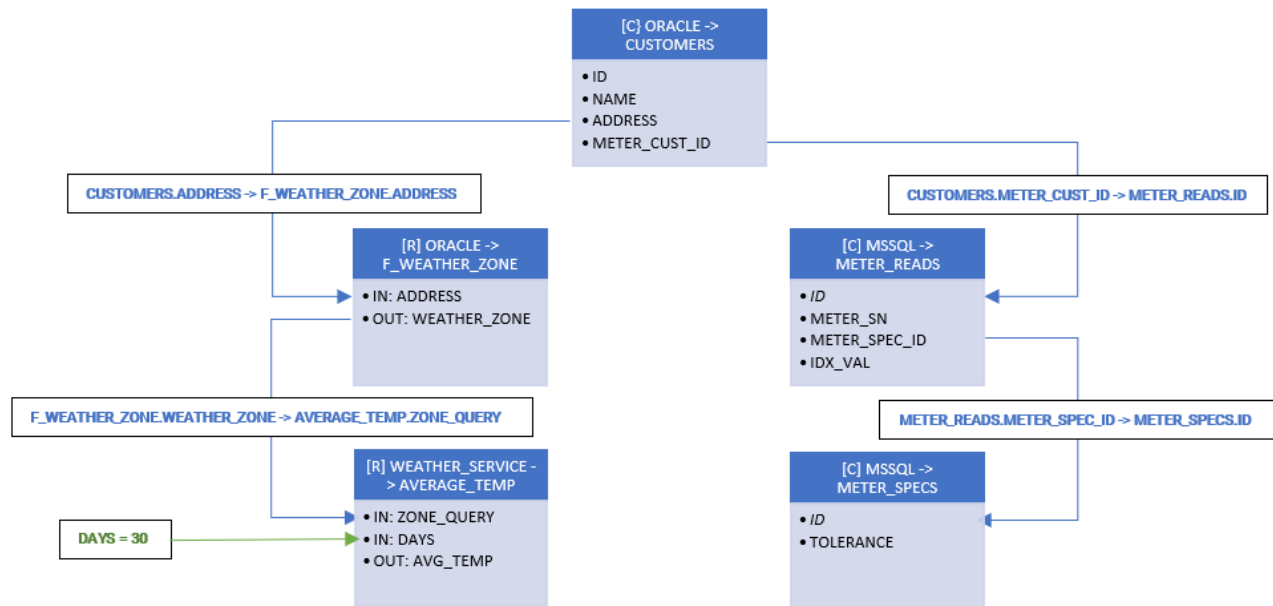
Let's now look at an example Aggregate definition and see how it functions and what it outputs.

We'll assume we have a project with the following Sources included:

- A connector named ORACLE, with a CUSTOMERS table and a F\_WEATHER\_ZONE function;
- A connector named WEATHER\_SERVICE with a AVERAGE\_WEATHER function;
- A connector named MSSQL with two tables called METER\_READS and METER\_SPECS;


We'll also assume that all of the fields used in the Aggregate exist and have been included in the project.

Here is our Aggregate:



Let's do an inventory of the elements:

Role	Element	Type	Details
<b>Main Source</b>	ORACLE -> CUSTOMERS	Collection	Our main source. A table of customers for our service. It contains data about the customers, including their addresses and the ids for their meters.
<b>Secondary Sources</b>	ORACLE -> F_WEATHER_ZONE	Routine	A stored function that creates a webservice-friendly weather zone query based on an address.
	WEATHER_SERVICE -> AVERAGE_TEMP	Routine	A webservice operation that returns the average temperature in a zone, over a number of days.  We have a <b>filter</b> on it which serves to set a static value for the number of days parameter.
	MSSQL -> METER_READS	Collection	A table containing readings from meters.  <b>i</b> We have not included the ID field since it will duplicate the METER_CUST_ID value. It also has a name coincidence with the field in the main source. If we would want to include it, we would have to set a custom name for the attribute.
	MSSQL -> METER_SPECS	Collection	A table containing specifications for meters.

			 We have not included the ID field since it will duplicate the METER_SPEC_ID value. It also has a name coincidence with the field in the main source. If we would want to include it, we would have to set a custom name for the attribute.
<b>Relationships</b>	ADDRESS -> WEATHER_ZONE	Routine Input	A relationship that feeds a customer's address into the function that determines the respective weather zone.
	WEATHER_ZONE -> ZONE_QUERY	Routine Input	A relationship that feeds a weather zone into the web operation for the average temperature.
	CUST_ID -> ID	Foreign Key	A relationship modeling a Foreign Key for meter ids.
	METER_SPEC_ID -> ID	Foreign Key	A relationship modeling a Foreign Key for meter specification ids.

Queries and responses on the aggregate will work the same as with any DTS collection, for example, for a particular CUSTOMER.ID (=123456), we would get a record of the form:

```
{
  "ID": 123456,
  "NAME": "Smith, John",
  "ADDRESS": "1 Street Lane, Townsville, Countryland",
  "METER_CUST_ID": 654321,
  "WEATHER_ZONE": "townsville,co",
  "AVG_TEMP": 301.4,
  "METER_SN": 112233445566,
  "METER_SPEC_ID": "acme-4",
  "IDX_VAL": 9000.04,
  "TOLERANCE": 0.03
}
```

The record will contain only the attributes we have included in the aggregate, all brought together in a single record and extracted based on our relationships, filters and query.

## 6.8 Security

The DTS architecture allows various components of the system to function remotely. This implies that the possibility of transferring sensitive data over improperly channels cannot be ignored.




Moreover, the remote operations capability opens the door for malicious actors to interfere with the various DTS subsystems in destructive ways.

To address these concerns, DTS implements its own security stack which ensures authentication of all components in the system and end-to-end encryption of all inter-module communication.

DTS uses three industry-standard technology stacks in its security implementation.


- a) Authentication and authorization of components is achieved using **X509** certificates. This allows the use of certificate chaining as a means of authentication, is inherently secure by virtue of the underlying encryption technology and provides easy control over validity periods and other types of authorization control.
  - The certificates used by DTS implement SHA256 hashing and RSA keys (configurable between 1024 and 4096 bits)
- b) General communication encryption is handled by an **AES** Galois Counter Mode (GCM) 256 bit mechanism. This one of the most secure variants of AES when implemented correctly while also preserving fast encryption performance for larger data volumes.
  - DTS uses new random initialization vectors for each message for maximum security
- c) Initial handshakes and key exchanges are secured using an **RSA** ECB – SHA256 mechanism.
  - The RSA key pairs used here are the ones associated with the X509 certificates at point (a).


For more information on the DTS Security implementation, please see:

- [Security Outline](#)  <sup>247</sup>
- [Registration and Authentication](#)  <sup>248</sup>
- [Security Setup](#)  <sup>252</sup>

## 6.8.1 Outline

The DTS security implementations uses the previously listed technology stacks as follows:

- A certificate authority signs a root certificate (**CTRL CERT**) for the DTS [Controller](#)  <sup>210</sup> module. This certificate contains the Controller's public key (**CTRL PUB KEY**). This certificate is also accompanied by its complementary private key (**CTRL PRIV KEY**).

 Each DTS deployment is capable of generating and maintaining a certificate authority. Alternatively, a 3rd party CA can be involved.

- Each connected component then receives a component certificate (**COMP CERT**) signed using **CTRL CERT**, as well as two RSA keys: The Controller's public key (**CTRL PUB KEY**) and the private key complementary to the component certificate (**COMP PRIV KEY**).

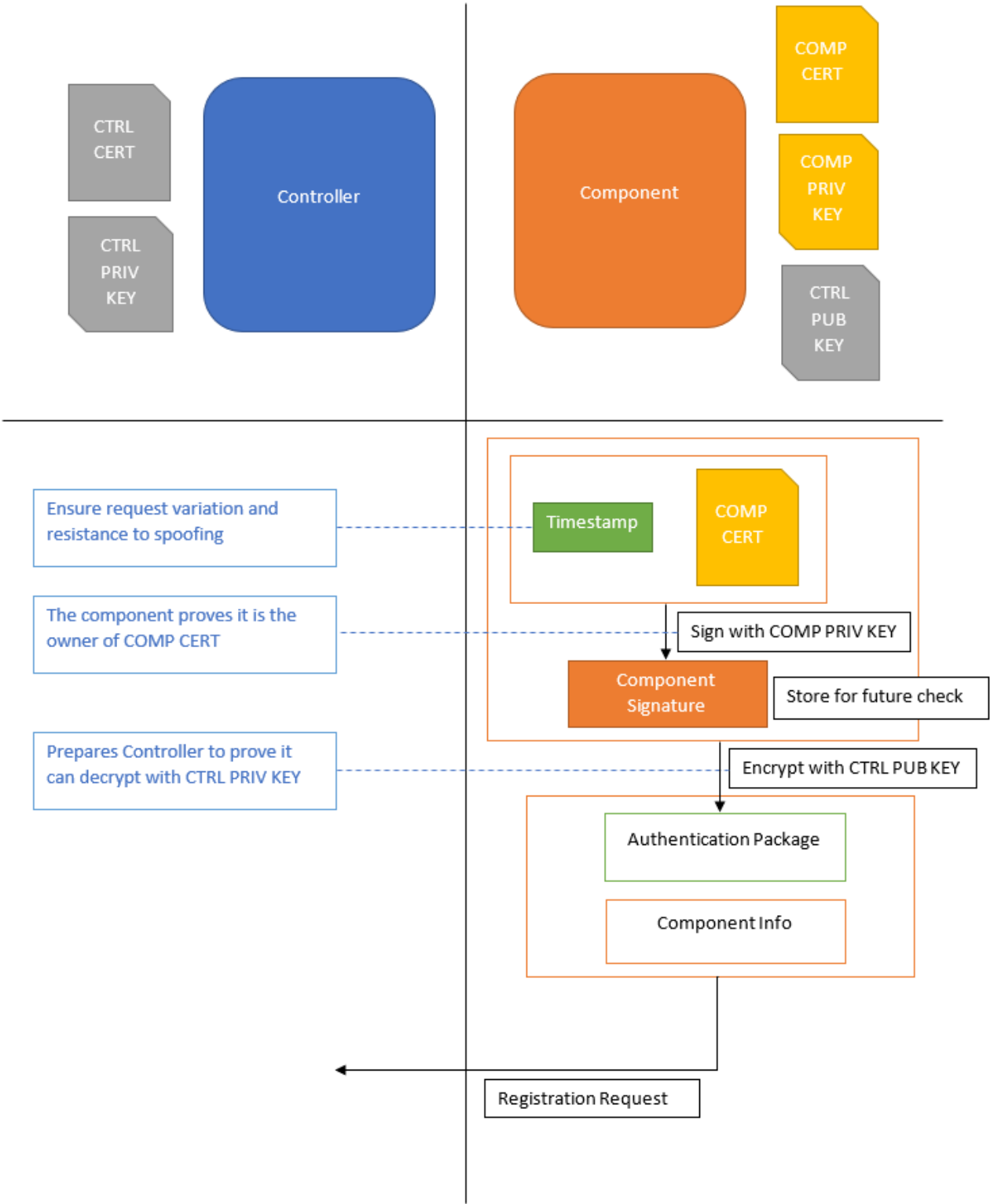
- When the component registers with the controller, it provides its **COMP CERT** to be verified using **CTRL CERT**. True ownership of the certificates is proven using the complementary private keys to encrypt parts of the registration request and reply.
- This registration exchange also provides the registered component with the AES keys which will be used in further communication on the initial channels. Each component, as well as the controller have their own main communication channel and other channels can be opened later depending on circumstances. Each channel has its own AES key. The initial AES key transfer is secured using the **COMP PUB/PRIV KEY** pair.
- Subsequent communication on the main channels will be encrypted with the previously exchanged AES keys.
- Whenever a new channel is put in use, its availability is communicated to the components concerned and the AES key is included. This communication is always done on an already secured channel (usually the component's main channel).

## 6.8.2 Registration and Authentication

Registration and Authentication of DTS Components to the system is achieved using an initial handshake process which ensures that all the actors involved are genuine and authorized DTS components.

The following diagrams present the entire initial handshake process in detail.

- The first step is for the Component that wants to register to compose its Registration Request.

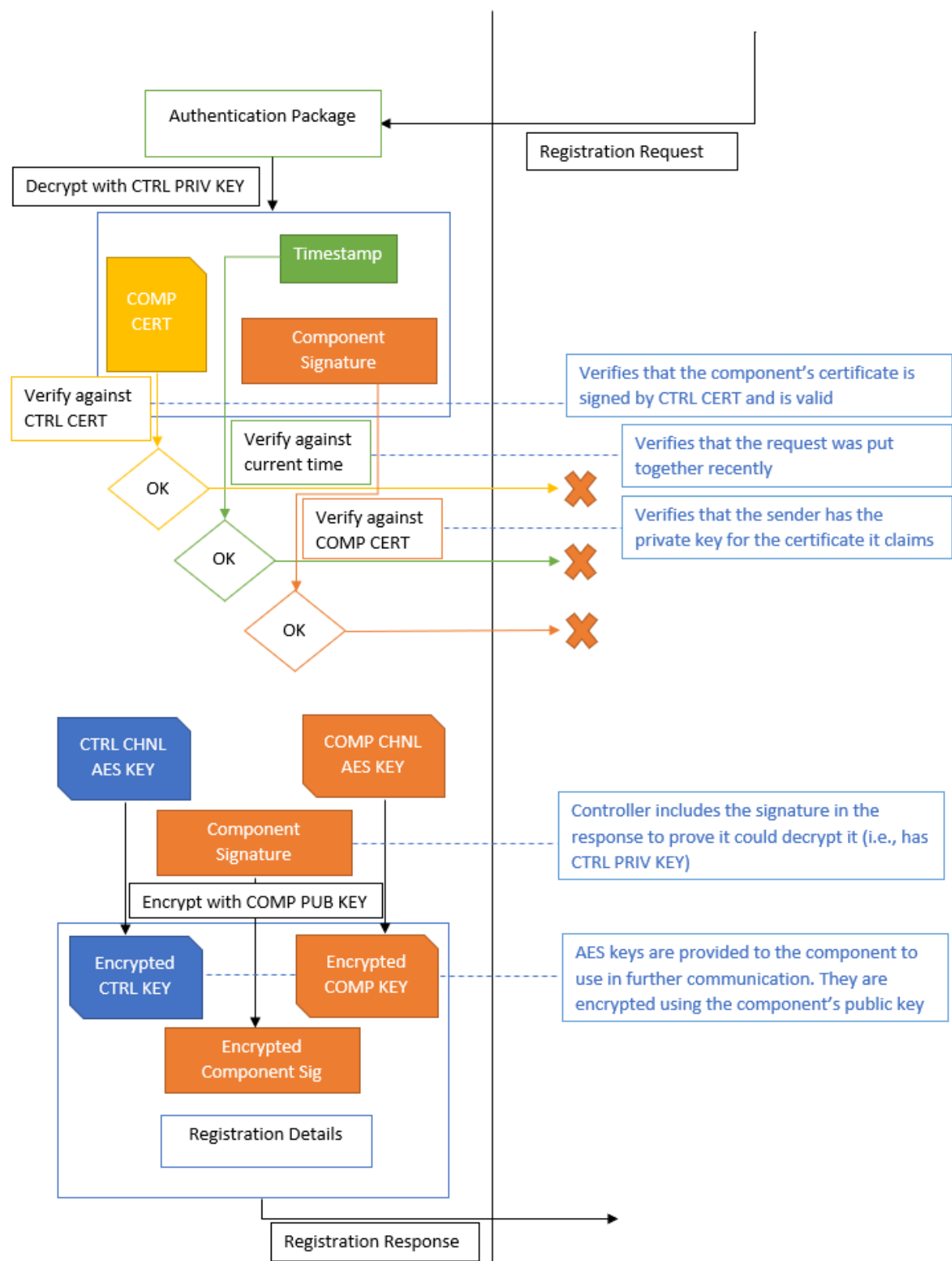


The Authentication Package has exactly the following binary format:

Bytes	Value
0 to 7	Little Endian representation of the current timestamp (as a 64-bit signed integer (long) representing milliseconds since epoch)
8 to 15	Little Endian representation of the byte length of COMP CERT (as a 64-bit signed integer (long) - [COMP_CERT_length] )
16 to [COMP_CERT_length] + 15	The Component's certificate encoded as a byte array - COMP CERT
Rest	The component signature as a byte array

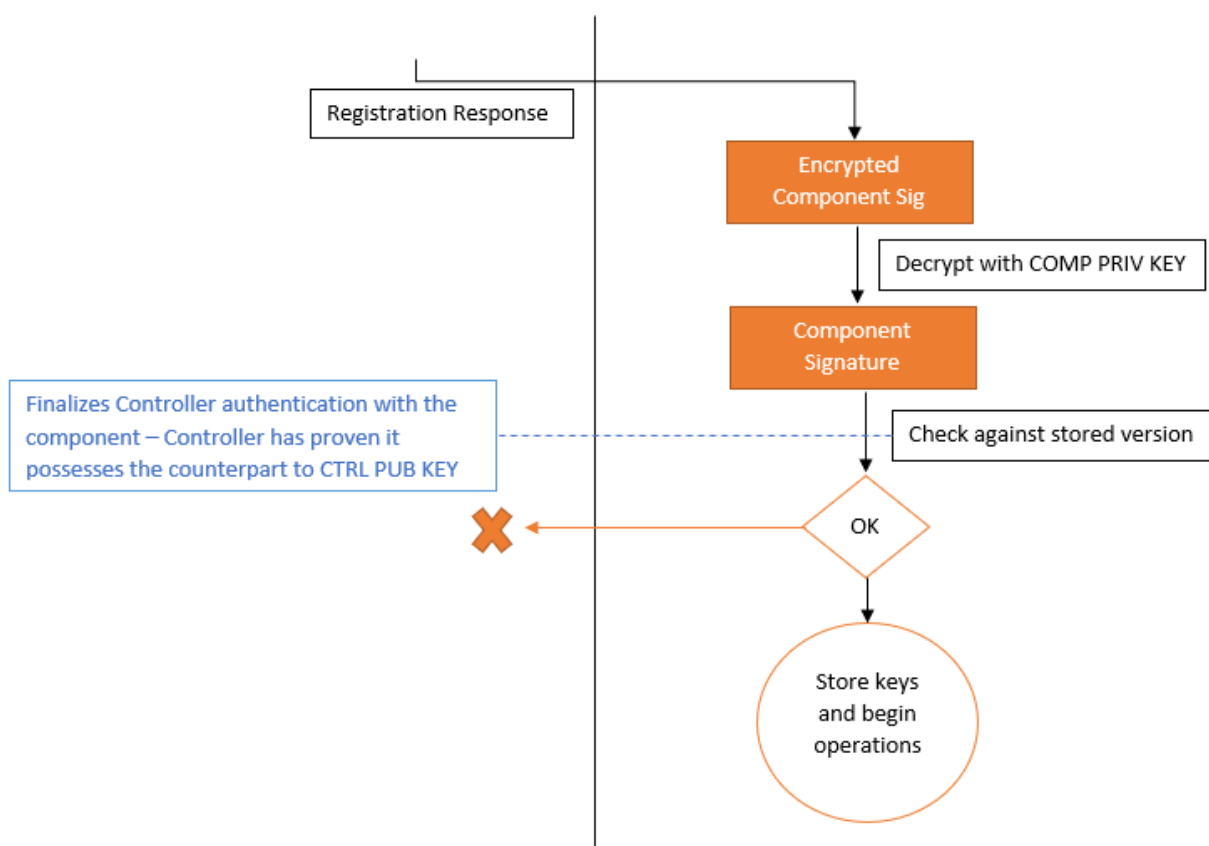
The resulting byte array is encoded as a Base64 String and included in the Registration Request.

- Next, the [Controller](#)<sup>210</sup> verifies the request and composes a response that will authenticate itself to the Component while also providing the encryption keys for further communication.



The encrypted channel keys and component signature are encoded as Base64 strings for inclusion in the registration response.

- Finally, the Component verifies the Controller's response and begins normal operations.



## 6.8.3 Setup

To setup the DTS security system, a certificate authority is required. This certificate authority will sign the DTS deployment's root certificate (CTRL CERT), which, in turn, will sign the certificates for the connected components (COMP CERT).

While 3rd party CAs can be used, DTS also provides an easy-to-use script (**x509/setup.sh**) which generates a CA and a root certificate (CTRL CERT) using OpenSSL, as well as key files in the required formats (CTRL PRIV KEY, CTRL PUB KEY).

The next step is to enable security for the Controller modules by setting the following environment variables on dts-controller and dts-gui-controller:

```
DTS_ENABLE_SECURITY = true
DTS_X509_CERTIFICATE_PATH = [Path to CTRL CERT]
DTS_RSA_PRIVATE_KEY_PATH = [Path to CTRL PRIV KEY in DER format]
```

From here on, only secured components will be able to connect to DTS.

To secure a component, a certificate must be generated for it. This certificate (COMP CERT) must be signed by CTRL CERT.

DTS also provides a script to facilitate this action (**x509/create-cert.sh**), provided the system is secured using a CA created using the previous script and not a 3rd party or custom one. This script generates and signs the certificate (COMP CERT) and produces key files in the required formats (COMP PRIV KEY, COMP PUB KEY). These files will need to be made available to the component for opening in its local environment, together with the Controller's public key file (CTRL PUB KEY).

Finally, security can now be enabled on the component by setting its following environment variables:

```
DTS_ENABLE_SECURITY=true
DTS_X509_CERTIFICATE_PATH = [Path to COMP CERT]
DTS_RSA_PRIVATE_KEY_PATH = [Path to COMP PRIV KEY in DER format]
DTS_CTRL_PUBLIC_KEY_PATH = [Path to CTRL PUB KEY in DER format]
```





# Development

## 7 Development

This section is aiming to guide you through using DTS's APIs to connect your applications to the DTS middleware or build specific custom endpoints for your DTS environment.

Some examples of possible development relating to DTS include:

- Creating a custom data pump (requesting data from various data sources through DTS, then processing and pumping it into some other system).
- Linking an existing application to DTS data (e.g. feed a complex asset management UI with data from otherwise incompatible systems).
- Creating a bespoke DTS connector library for a specialized data source.

Depending on the purpose of the development, only one of the two facets of DTS extensibility is usually used:

- [Client Development](#)<sup>[256]</sup>
- [Producer Development](#)<sup>[288]</sup>

For any DTS related development and even for some administrative jobs, it is very helpful to be familiar with the formats used in the DTS Repository (Mongo).

- [Project Artifacts](#)<sup>[288]</sup>

### 7.1 Client

---

In order to build or adapt an application to consume data from various data sources via DTS, some level of extension of DTS's Client functionality usually needs to be developed.

DTS Client functionality can be accessed in the following ways (in ascending order of customization possibilities, feature set, but also implementation difficulty) :

- Using one of the [pre-built client implementations](#)<sup>[257]</sup> providing simplified access in a certain environment: Smallworld Client, Webservice Client
- Implementing the fully featured API library: [Java](#)<sup>[257]</sup>
- Hooking in directly communication backbone API: [Redis/JSON](#)<sup>[274]</sup>

The first two options allow using high level APIs to connect to DTS, access projects and make requests, while the last option opens up DTS to direct management via internal commands that can be invoked from any programming environment for which a Redis driver exists, but requires more micromanagement of operations.

### 7.1.1 Pre-Built

DTS provides some out-of-the-box Client implementations for easy use in specific environments. Each Pre-Built Client implementation is purpose-built for the targeted environment and caters to its native workflow patterns.

The currently available Client implementations are:

- [The Smallworld Client](#)<sup>[176]</sup>
  - Connects to DTS using the [Java Client Library](#)<sup>[257]</sup>.
  - Provides access to DTS resources from the Magik environment.
  - Translates all artifacts into native Smallworld objects that can be directly used in the environment.
- [The Webservice Client](#)<sup>[186]</sup>
  - Connects to DTS using the [Java Client Library](#)<sup>[257]</sup>.
  - Provides access to DTS resources via HTTP Requests.
  - Must be generated using the [Web UI](#)<sup>[70]</sup> where it can be greatly customized without coding.
  - Translates all artifacts into standardized Web objects.

### 7.1.2 Java Library

DTS provides a fully featured Client library for Java with its **dts-client** package. This section outlines how the library can be used for connecting to DTS and making requests and provides some insight into implementing custom DTS clients.

#### Library package

The Client library package is currently only available as a set of JARs which includes all dependencies. This means that you can import it into your project as is, or you can set it up in a custom repository (Maven, Gradle, etc.), depending on your development environment. For reference, the external dependencies are (Gradle format):

- redis.clients:jedis:3.2.0
- com.google.code.gson:gson:2.8.6
- org.ow2.asm:asm-all:5.2
- commons-codec:commons-codec:1.15
- com.sun.mail:javax.mail:1.6.2

The Client library package is built and tested using Oracle JDK 8, which is the minimum version supported for development using the library.

## Environment

While your implementation can make use of any environment variables it requires, the following are leveraged by the DTS client libraries.

Variable	Default	Function
<code>DTS_REDIS_HOST_NAME</code>	localhost	The host name or IP address of the DTS Redis server ( <a href="#">Internal Communication Bus</a> <sup>218</sup> )
<code>DTS_REDIS_PORT</code>	6379	The port of the Redis server
<code>DTS_CLIENT_NAME</code>	anonymous	A name given to the client that will be used to tag it administrative purposes. This name will be suffixed with "@[hostname]".
<code>DTS_DEBUG_LOGGING</code>	false	Flag to toggle debug logging
<code>DTS_ENABLE_SECURITY</code>	false	Flag to toggle internal <a href="#">security</a> <sup>246</sup> (if the central DTS deployment is secured, clients are also required to be)
<code>DTS_X509_CERTIFICATE</code>	null	The local path to the X509 certificate file that was generated for this client (see <a href="#">Security Setup</a> <sup>252</sup> )
<code>DTS_RSA_PRIVATE_KEY</code>	null	The local path to the file containing the RSA Private Key corresponding to the X509 certificate in DER format.
<code>DTS_CTRL_PUBLIC_KEY</code>	null	The local path to the file containing the RSA Public Key of the DTS Controller in DER format.

In a standard secured DTS environment, the values for the Client environment variables may look something like this:

```
DTS_REDIS_HOST_NAME=dts-server # Host name for the machine running the Docker pl
DTS_REDIS_PORT=7877 # The default port the DTS contained Redis server is mapped
```

```

DTS_CLIENT_NAME=test1 # Assuming the host name of the machine running the client
DTS_DEBUG_LOGGING=true
DTS_ENABLE_SECURITY=true
DTS_X509_CERTIFICATE_PATH=/home/dts/client/security/test1.crt
DTS_RSA_PRIVATE_KEY_PATH=/home/dts/client/security/test1.key.der
DTS_CTRL_PUBLIC_KEY_PATH=/home/dts/client/security/ca.pubkey.der

```

## Usage

The usage of the Java Client Library is best illustrated by example. Please see:

- [Direct Usage Example](#)<sup>259</sup>
- [Custom Implementation Example](#)<sup>263</sup>

The library is also accompanied by Javadoc describing the details of all classes, constants, methods, etc.

### 7.1.2.1 Direct Usage Example

In this example, we use the DTSCClient class as-is and go through the initialization process and a few requests.

The code example assumes the following project structure is available:

Project	Connector	Asset	Type	Parameters
myproject	myOraConnector	DTSDemo.NICE_TABLE	Collection	N/A
		DTSDemo.NICE_FUNCTION	Routine	INT32, STRING
		DTSDemo.NICE_STREAMY_FUNCTION	Streaming Routine	INT32
	kafka	test-topic	Topic	N/A
	N/A	nice_aggregate	Aggregate	N/A

 To see how to create projects, connectors and configure assets, please see the [Web UI Section](#)<sup>261</sup>

```
import com.alloy.dts.DTSException;
import com.alloy.dts.client.DTSClient;
import com.alloy.dts.client.DTSDatasource;
import com.alloy.dts.client.DTSEndpoint;
import com.alloy.dts.client.DTSAggregationEndpoint;
import com.alloy.dts.model.DTSPredicate;
import com.alloy.dts.record.IValuesContainer;
import com.alloy.dts.record.StreamRecord;

import java.util.Arrays;
import java.util.List;
import java.util.Map;
import java.util.concurrent.CountDownLatch;
import java.util.concurrent.TimeUnit;

public class DirectDTSClientExample {

    public static void main (String [] args) throws Exception {

        // First create an instance of DTSClient
        DTSClient dtsClient = new DTSClient();

        // Open the DTS project in the client
        // This sends a request to the DTS Controller to make the project available to the Client
        // The Controller will respond immediately with some information regarding the project,
        // and orders the necessary producers to be started if they aren't already
        // this means that the process is asynchronous and data endpoints will not become available
        // the instant the open() method returns.
        dtsClient.open("myproject");

        // Retrieve the datasource created for the opened project
        DTSDatasource ds = dtsClient.getDatasource("myproject");

        // Make sure the datasource is configured to contain our connector
        ds.checkEndpointName("myOraConnector");

        // We must now wait for the endpoint for our connector to become available
        ds.waitForAllEndpoints(60 * 1000);

        // We get a handle on the endpoint for the connector
        DTSEndpoint myEndpoint = ds.getEndpoint("myOraConnector");
        // We open a stream to the table we want using a certain query predicate
        String streamId = myEndpoint.openRecordStream("DTSDEMO.NICE_TABLE",
            DTSPredicate.eq("value", "100"));
        // We fully consume the stream
        while(myEndpoint.streamHasMore(streamId)) {
            // 100 records at a time
            List<? extends StreamRecord> records = myEndpoint.getRecordsFromStream(streamId, 100);
            for (StreamRecord rec : records) {
                // record fields can be accessed using reflection or transforming them into Maps
                // here, we use maps
            }
        }
    }
}
```

```

        Map<String, Object> recordMap = rec.asMap();
        /*
        do something with the record
        */
    }
}
// Stream is spent, we close it
myEndpoint.closeRecordStream(streamId);

// Let's also invoke a stored function that receives 2 arguments
IValuesContainer results = myEndpoint.executeCall("DTS_DEMO.NICE_FUNCTION", new Object[] {12,
"active"});
// And access the single result using reflection this time
Class resultClass =
dtsClient.getPayloadFactory().getTypesCluster().getClassForTypeName("DTS_DEMO.NICE_FUNCTION.RESU
LTS");
Object theResult = resultClass.getField("res1").get(results);
/*
do something with it
*/

// We'll now get a record from an aggregate

// We get a handle on the aggregation endpoint. There is just one per project datasource.
DTSAggregationEndpoint aggEndpoint = ds.getAggregationEndpoint();

// We open a stream on the aggregate as if it were a regular collection
streamId = aggEndpoint.openRecordStream("nice_aggregate", DTSPredicate.eq("id", "3"));

// And we pick up our aggregate record
StreamRecord record = aggEndpoint.getRecordsFromStream(streamId, 1).get(0);
/*
do something with it
*/

// Stream is not needed anymore, we close it
aggEndpoint.closeRecordStream(streamId);

// Now let's use a remote call that returns a record stream
streamId = myEndpoint.executeStreamCall("DTS_DEMO.NICE_STREAMY_FUNCTION", new Object[]
{1234});

// We fully consume the stream
while(myEndpoint.streamHasMore(streamId)) {
    // 100 records at a time
    List<? extends StreamRecord> records = myEndpoint.getRecordsFromStream(streamId, 100);
    for (StreamRecord rec : records) {
        // record fields can be accessed using reflection or transforming them into Maps
        // here, we use maps
        Map<String, Object> recordMap = rec.asMap();
        /*

```

```

        do something with the record
        */
    }
}
// Stream is spent, we close it
myEndpoint.closeRecordStream(streamId);

// Let's also do some work on a topic
DTSEndpoint kafkaEndpoint = ds.getEndpoint("kafka");

// First, we can subscribe to it -> which gives us a stream we can consume
// We provide a group ID so that Kafka remembers where we leave off
streamId = kafkaEndpoint.subscribeToTopic("test-topic", "dts-java");

// Now we can poll some entries from the topic
List<? extends StreamRecord> records = kafkaEndpoint.pollTopic(streamId, 5000);
for (StreamRecord rec : records) {
    Map<String, Object> recordMap = rec.asMap();
    // topic records will always contain the key and msg attributes
    // however, their types can vary
    System.out.println(recordMap.get("key") + " -> " + recordMap.get("msg"));
}

// We unsubscribe to release the stream
kafkaEndpoint.unsubscribeFromTopic(streamId);

// We can also do some pushing
// First we do a single string/string push
kafkaEndpoint.pushToTopic("test-topic", "some_key", "some_message");
// Now let's push a record with a binary message
IValuesContainer aRecord = (IValuesContainer) dtsClient.getPayloadFactory().getTypesCluster()
    .getClassForTypeName("test-topic.RECORD").newInstance();
aRecord.readValues(new Object[] { "some_other_key", new byte[] {67,83,28,91,4,77,82,12}});
kafkaEndpoint.pushToTopic("test-topic", aRecord);
// Let's also push multiple records at once
kafkaEndpoint.pushManyToTopic(
    "test-topic",
    new Object[] { "key1", "key2", "key3"},
    new Object[] { "msg1", "msg2", "msg3"});

// Finally, let's shut down the client
// This will ensure all streams are closed and all resources are released
dtsClient.closeAllAndShutdown();
}
}

```

### 7.1.2.2 Custom Implementation Usage

This example shows a more complex implementation of a DTSCClient, where we extend the class and customize elements to suit our purposes.

First, the Client (CustomClient.java):

```
import com.alloy.dts.DTSException;
import com.alloy.dts.client.*;
import com.alloy.dts.model.DTSPredicate;
import com.alloy.dts.record.StreamRecord;
import com.alloy.dts.type.DTSTypesClusterManager;

import java.util.List;
import java.util.concurrent.CountDownLatch;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;

import static com.alloy.dts.type.DTSTypes.*;

public class CustomClient extends DTSCClient {

    // Our client will be a singleton for ease of use throughout the application
    private static CustomClient instance ;
    public static CustomClient getInstance() {
        if (instance == null) {
            instance = new CustomClient();
        }
        return instance ;
    }

    // We will only use one collection, so let's keep it here
    DTSRemoteCollection collection;

    public CustomClient() {
        super();
        // we also want to override some root types in our client
        initSpecialTypes();
    }

    private void initSpecialTypes() {
        // overriding types is as simple as calling this static method on DTSTypesClusterManager
        // but be aware that you will most likely need to define and implement
        // custom Serialization/Deserialization routines
        DTSTypesClusterManager.overrideRootType (GEOMETRY_TYPE , CustomGeometry.class);
        DTSTypesClusterManager.overrideRootType (POINT_TYPE , CustomGeometry.class);
        DTSTypesClusterManager.overrideRootType (LINE_TYPE , CustomGeometry.class);
        DTSTypesClusterManager.overrideRootType (AREA_TYPE , CustomGeometry.class);
    }

    // We override the open() method to automatically wait for our endpoint and set our collection
    @Override
```

```

public void open(String projectName) throws TimeoutException, DTSEException{
    super.open(projectName);
    DTSDatasource ds = getDatasource(projectName);
    if (!ds.waitForEndpoints (Arrays.asList("myOnlyConnector"), 60 * 1000)) {
        throw new DTSEException("No endpoint", ex);
    }
    collection = ds.getEndpoint("myOnlyConnector").getCollection("public.all_things");
}

// We decide we want our client to react in some way when a stream times out
@Override
public void onClosedStream(String streamId) {
    System.out.println("Stream " + streamId + " has timed out!");
    /*
    do something
    */
}

// and here is the point of our implementation - a nice shortcut to getting the geometric center of an object
public double[] getObjectCenter(String objectId) throws Exception {
    // we only need 1 record, but we still need a stream
    // we'll use another way of managing it this time
    // first we get the DTSRemoteRecordStream object using createStream on the collection (which gets fed our query
    predicate)
    DTSRemoteRecordStream stream = collection.createStream(DTSPredicate.eq("id", objectId));
    // then we advance the stream by 1 so that the record we're looking for makes its way into the buffer
    stream.advance(1);
    // then we grab the record from the buffer
    StreamRecord object = (StreamRecord) stream.getCurrentRecordContainer().getRecords().get(0);
    // we use the getFieldValue() method to extract the geometry (which will be in our CustomGeometry format
    because we overrode the types)
    CustomGeometry geom = (CustomGeometry) object.getFieldValue("geom");
    // and finally call a method we created on CustomGeometry that calculates its center
    // while this particular result could easily be obtained without overriding root types and defining custom
    deserializers
    // we are using it to illustrate a point and it is easy to see how this can be a powerful tool in more complex
    scenarios
    return geom.getCenter();
}
}

```

The CustomGeometry class we're using (CustomGeometry.java):

```

import com.alloy.dts.DTSEException;
import com.alloy.dts.type.json.geojson.GeoJson;
import com.alloy.dts.type.json.geojson.GeoJsonLineString;
import com.alloy.dts.type.json.geojson.GeoJsonPoint;
import com.alloy.dts.type.json.geojson.GeoJsonPolygon;

import java.util.ArrayList;

public class CustomGeometry {

    // We'll hold the GeoJson that comes through
    private GeoJson originalGeom;

```

```

public GeoJson getOriginalGeom() {
    return originalGeom;
}

public void setOriginalGeom(GeoJson originalGeom) {
    this.originalGeom = originalGeom;
}

// And calculate the center when needed
public double[] getCenter() throws DTSEException {
    if (getOriginalGeom() instanceof GeoJsonPoint) {
        // trivial...
        GeoJsonPoint point = (GeoJsonPoint) getOriginalGeom();
        return point.getCoordinates();
    } else if (getOriginalGeom() instanceof GeoJsonLineString) {
        // naive...
        GeoJsonLineString line = (GeoJsonLineString) getOriginalGeom();
        ArrayList<double[]> coords = line.getCoordinates();
        return new double[] {
            (coords.get(0)[0] - coords.get(coords.size()-1)[0]) / 2,
            (coords.get(0)[1] - coords.get(coords.size()-1)[1]) / 2
        };
    } else if (getOriginalGeom() instanceof GeoJsonPolygon) {
        // eh...
        throw new DTSEException("Not worth it...");
    }
    throw new DTSEException("Unsupported geometry");
}
}

```

And the custom Deserializer (CustomGeometryDeserializer.java):

```

// the package is important here! In order for the deserializer to register, it needs to be in this package!
package com.alloy.dts.type.json;

import com.mypackage.CustomGeometry;
import com.alloy.dts.type.json.geojson.GeoJson;
import com.google.gson.*;

import java.lang.reflect.Type;

// Our deserializer needs to implement JsonSerializer<the_class_we_want_to_output>
public class CustomGeometryDeserializer implements JsonSerializer<CustomGeometry> {

    public static void registerWithGsonBuilder(GsonBuilder builder){
        builder.registerTypeAdapter(CustomGeometry.class, new CustomGeometryDeserializer());
    }

    // And here is where we implement our deserialization
    @Override
    public CustomGeometry deserialize(JsonElement json, Type typeOfT, JsonDeserializationContext context) throws
    JsonParseException {
        // We simply use the deserialization context to deserialize the object to the class it was meant to be
        GeoJson geoJson = context.deserialize(json, GeoJson.class);
    }
}

```

```
// The we create our custom object and place the original geometry inside it
CustomGeometry geom = new CustomGeometry();
geom.setOriginalGeom(geoJson);

return geom;
}
```

### 7.1.3 .NET Library

DTS provides a fully featured Client library for Microsoft® .NET™ with its **dts-client-dotnet** package. This section outlines how the library can be used for connecting to DTS and making requests and provides some insight into implementing custom DTS clients.

#### Library package

The .NET Client library package is currently only available as a nuget package which includes the base library and references all required dependencies. This means that you can import it into your VS project as is, or you can extract it and include the DLLs manually and resolve the dependencies. For reference, the external dependencies are:

- Microsoft.CodeAnalysis.Common (4.2.0)
- Microsoft.CodeAnalysis.CSharp (4.2.0)
- ServiceStack.Redis (6.1.0)
- System.CodeDom (6.0.0)

The .NET Client library package is built and tested using the .NET 6.0 Framework.

 If you require a build in a different .NET Framework, please contact us for a custom build.

#### Environment

While your implementation can make use of any environment variables it requires, the following are leveraged by the DTS client libraries.

Variable	Default	Function
<b>DTS_REDIS_HOST_NAME</b>	localhost	The host name or IP address of the DTS Redis server ( <a href="#">Internal Communication Bus</a> <sup>218</sup> )
<b>DTS_REDIS_PORT</b>	6379	The port of the Redis server

<b>DTS_CLIENT_NAME</b>	anonymous	A name given to the client that will be used to tag it administrative purposes. This name will be suffixed with "[hostname]".
<b>DTS_DEBUG_LOGGING</b>	false	Flag to toggle debug logging
<b>DTS_ENABLE_SECURITY</b>	false	Flag to toggle internal <a href="#">security</a> (if the central DTS deployment is secured, clients are also required to be)
<b>DTS_X509_CERTIFICATE_PATH</b>	null	The local path to the X509 certificate file that was generated for this client (see <a href="#">Security Setup</a> )
<b>DTS_RSA_PRIVATE_KEY_PATH</b>	null	The local path to the file containing the RSA Private Key corresponding to the X509 certificate in DER format.
<b>DTS_CTRL_PUBLIC_KEY_PATH</b>	null	The local path to the file containing the RSA Public Key of the DTS Controller in DER format.

In a standard secured DTS environment, the values for the Client environment variables may look something like this:

```
DTS_REDIS_HOST_NAME=dts-server # Host name for the machine running the Docker platform
DTS_REDIS_PORT=7877 # The default port the DTS contained Redis server is mapped to
DTS_CLIENT_NAME=test1 # Assuming the host name of the machine running the client
DTS_DEBUG_LOGGING=true
DTS_ENABLE_SECURITY=true
DTS_X509_CERTIFICATE_PATH=/home/dts/client/security/test1.crt
DTS_RSA_PRIVATE_KEY_PATH=/home/dts/client/security/test1.key.der
DTS_CTRL_PUBLIC_KEY_PATH=/home/dts/client/security/ca.pubkey.der
```

## Usage

The usage of the .NET Client Library is best illustrated by example. Please see:

- [Direct Usage Example](#)
- [Custom Implementation Example](#)

### 7.1.3.1 Direct Usage Example

In this example, we use the DTSCClient class as-is and go through the initialization process and a few requests.

The code example assumes the following project structure is available:

Project	Connector	Asset	Type	Parameters
myproject	myOraConnector	DTSDemo.NICE_TABLE	Collection	N/A
		DTSDemo.NICE_FUNCTION	Routine	INT32, STRING
		DTSDemo.NICE_STREAMY_FUNCTION	Streaming Routine	INT32
	N/A	nice_aggregate	Aggregate	N/A

 To see how to create projects, connectors and configure assets, please see the [Web UI Section](#) <sup>261</sup>

```

using Alloy.DTS.Base.Model;
using Alloy.DTS.Base.Record;
using Alloy.DTS.Client;

namespace Alloy.DTS.ClientExamples
{
    public class DirectDTSClientExample
    {
        public static void Main(string[] args)
        {
            // First create an instance of DTSCient
            DTSCient dtsClient = new DTSCient();

            // Open the DTS project in the client
            // This sends a request to the DTS Controller to make the project available to
            the Client
            // The Controller will respond immediately with some information regarding the
            project,
            // and orders the necessary producers to be started if they aren't already
            // this means that the process is asynchronous and data endpoints will not
            become available
            // the instant the open() method returns.
            dtsClient.Open("myproject");

            // Retrieve the datasource created for the opened project
            DTSDatasource ds = dtsClient.GetDatasource("myproject");

            // Make sure the datasource is configured to contain our connector
            ds.CheckEndpointName("myOraConnector");

            // We must now wait for the endpoint for our connector to become available
            ds.WaitForAllEndpoints(60 * 1000);
        }
    }
}

```

```

// We get a handle on the endpoint for the connector
DTSEndpoint myEndpoint = ds.GetEndpoint("myOraConnector");
// We open a stream to the table we want using a certain query predicate
string streamId = myEndpoint.OpenRecordStream("DTSDEMO.NICE_TABLE",
    DTSPredicate.Eq("value", "100"));
// We fully consume the stream
while (myEndpoint.StreamHasMore(streamId))
{
    // 100 records at a time
    List <IValuesContainer> records = myEndpoint.GetRecordsFromStream(streamId,
100);

    foreach (IValuesContainer rec in records)
    {
        // record fields can be accessed using reflection or transforming them
        // here, we use maps
        Dictionary<string, object> recordMap = rec.AsMap();
        /*
        do something with the record
        */
    }
}
// Stream is spent, we close it
myEndpoint.CloseRecordStream(streamId);

// Let's also invoke a stored function that receives 2 arguments
IValuesContainer results = myEndpoint.ExecuteRoutine("DTS_DEMO.NICE_FUNCTION",
new Object[] { 12, "active" });
// And access the single result using reflection this time
Type resultClass =
dtsClient.PayloadFactory.TypesCluster.GetSystemType("DTS_DEMO.NICE_FUNCTION.RESULTS");
object theResult = resultClass.GetProperty("res1").GetValue(results);
/*
do something with it
*/

// We'll now get a record from an aggregate

// We get a handle on the aggregation endpoint. There is just one per project
datasource.
DTSAggregationEndpoint aggEndpoint = ds.AggregationEndpoint;

// We open a stream on the aggregate as if it were a regular collection
streamId = aggEndpoint.OpenRecordStream("nice_aggregate", DTSPredicate.Eq("id",
"3"));

// And we pick up our aggregate record
IValuesContainer record = aggEndpoint.GetRecordsFromStream(streamId, 1)[0];
/*
do something with it
*/

```

```

        // Stream is not needed anymore, we close it
        aggEndpoint.CloseRecordStream(streamId);

        // Finally, let's use a remote call that returns a record stream
        streamId = myEndpoint.ExecuteStreamRoutine("DTS_DEMO.NICE_STREAMY_FUNCTION",
new Object[] { 1234 });

        // We fully consume the stream
        while (myEndpoint.StreamHasMore(streamId))
        {
            // 100 records at a time
            List <IValuesContainer> records =
myEndpoint.GetRecordsFromStream(streamId, 100);
            foreach (IValuesContainer rec in records)
            {
                // record fields can be accessed using reflection or transforming them
                // here, we use maps
                Dictionary<string, object> recordMap = rec.AsMap();
                /*
                do something with the record
                */
            }
        }
        // Stream is spent, we close it
        myEndpoint.CloseRecordStream(streamId);
    }
}

```

### 7.1.3.2 Custom Implementation Example

This example shows a more complex implementation of a DTSCClient, where we extend the class and customize elements to suit our purposes.

First, the Client (CustomClient.cs):

```

using Alloy.DTS.Base.Model;
using Alloy.DTS.Base.Record;
using Alloy.DTS.Base.Type;
using Alloy.DTS.Client;

namespace Alloy.DTS.ClientTest
{
    public class CustomClient : DTSCClient
    {

```

```

// Our client will be a singleton for ease of use throughout the application
private static CustomClient instance;

public static CustomClient Instance
{
    get {
        if (instance == null)
        {
            instance = new CustomClient();
        }
        return instance;
    }
}

// We will only use one collection, so let's keep it here
DTSRemoteCollection? _collection;

public CustomClient() : base()
{
    // we also want to override some root types in our client
    initSpecialTypes();
}

private void initSpecialTypes()
{
    // overriding types is as simple as calling this static method on
    DTSTypesClusterManager
    // but be aware that you will most likely need to define and implement
    // custom Serialization/Deserialization routines
    DTSTypesCluster.OverrideRootType(DTSTypes.GEOMETRY_TYPE,
    typeof(CustomGeometry));
    DTSTypesCluster.OverrideRootType(DTSTypes.POINT_TYPE, typeof(CustomGeometry));
    DTSTypesCluster.OverrideRootType(DTSTypes.LINE_TYPE, typeof(CustomGeometry));
    DTSTypesCluster.OverrideRootType(DTSTypes.AREA_TYPE, typeof(CustomGeometry));
    // Register the CustomGeometry serializer
    JsonEngine.Options.Converters.Add(new CustomGeometrySerializer());
}

// We override the open() method to automatically wait for our endpoint and set our
collection
public override void Open(string projectName) {
    base.Open(projectName);
    DTSDatasource ds = GetDatasource(projectName);
    if (!ds.WaitForEndpoints(new List<string> { "myOnlyConnector" }, 60 * 1000))
        throw new Exception("No endpoint");

    _collection =
ds.GetEndpoint("myOnlyConnector").Collections["public.all_things"];
}

// We decide we want our client to react in some way when a stream times out

```

```

protected override void OnClosedStream(string streamId)
{
    Console.WriteLine("Stream " + streamId + " has timed out!");
    /*
    do something
    */
}

// and here is the point of our implementation - a nice shortcut to getting the
geometric center of an object
public double[] GetObjectCenter(string objectId)
{
    // we only need 1 record, but we still need a stream
    // we'll use another way of managing it this time
    // first we get the DTSRemoteRecordStream object using createStream on the
collection (which gets fed our query predicate)
    DTSRemoteRecordStream stream = _collection.CreateStream(DTSPredicate.Eq("id",
objectId));
    // then we advance the stream by 1 so that the record we're looking for makes
its way into the buffer
    stream.Advance(1);
    // then we grab the record from the buffer
    IValuesContainer obj = stream.RecordContainer.Records[0];
    // we use the getFieldValue() method to extract the geometry (which will be in
our CustomGeometry format because we overrode the types)
    CustomGeometry geom = (CustomGeometry)obj.AsMap()["geom"];
    // and finally call a method we created on CustomGeometry that calculates its
center
    // while this particular result could easily be obtained without overriding
root types and defining custom deserializers
    // we are using it to illustrate a point and it is easy to see how this can be
a powerful tool in more complex scenarios
    return geom.GetCenter();
}
}
}

```

The CustomGeometry class we're using (CustomGeometry.cs):

```

using Alloy.DTS.Base.Model.Geometry;

namespace Alloy.DTS.ClientTest
{
    public class CustomGeometry
    {
        // We'll hold the GeoJson that comes through
        public GeoJson OriginalGeom { get; set; }

        public CustomGeometry(GeoJson geoJson)
        {

```

```

        OriginalGeom = geoJson;
    }

    // And calculate the center when needed
    public double[] GetCenter()
    {
        if (OriginalGeom is GeoJsonPoint) {
            // trivial...
            GeoJsonPoint point = (GeoJsonPoint)OriginalGeom;
            return point.Coordinates;
        } else if (OriginalGeom is GeoJsonLineString) {
            // naive...
            GeoJsonLineString line = (GeoJsonLineString)OriginalGeom;
            List<double[]> coords = line.Coordinates;
            return new double[] {
                (coords[0][0] - coords[coords.Count-1][0]) / 2,
                (coords[0][1] - coords[coords.Count-1][1]) / 2
            };
        } else if (OriginalGeom is GeoJsonPolygon) {
            // eh...
            throw new Exception("Not worth it...");
        }
        throw new Exception("Unsupported geometry");
    }
}

```

And the custom Deserializer (CustomGeometrySerializer.cs):

```

using Alloy.DTS.Base.Model;
using Alloy.DTS.Base.Model.Geometry;
using System.Text.Json;
using System.Text.Json.Serialization;

namespace Alloy.DTS.ClientTest
{
    public class CustomGeometrySerializer : JsonConverter<CustomGeometry>
    {
        // We will need some clean JsonSerializerOptions as well
        private JsonSerializerOptions _cleanSerializerOptions = new JsonSerializerOptions
        {
            WriteIndented = true,
            Converters =
            {
                new ResourceIdentifierSerializer(),
                new GeoJsonSerializer()
            }
        };
    }
}

```

```

        public override CustomGeometry? Read(ref Utf8JsonReader reader, Type typeToConvert,
        JsonSerializerOptions options)
        {
            // We simply use the serializer options to deserialize the object to the class
            it was meant to be
            GeoJson geoJson = (GeoJson)JsonSerializer.Deserialize(
                JsonDocument.ParseValue(ref reader).RootElement.GetRawText(),
                typeof(GeoJson),
                _cleanSerializerOptions);
            // The we create our custom object and place the original geometry inside it
            CustomGeometry geom = new CustomGeometry(geoJson);

            return geom;
        }

        public override void Write(Utf8JsonWriter writer, CustomGeometry value,
        JsonSerializerOptions options)
        {
            throw new NotImplementedException();
        }
    }
}

```

#### 7.1.4 Redis/JSON

All DTS components communicate to the Core via the Internal Communication Bus (ICB), which is a Redis Server. Messages exchanged on the Redis channels are all in JSON format, following a specific protocol.

In this section we'll illustrate how an application can register as a DTS Client, make requests and parse responses using direct ICB communication.

## Messages

All messages exchanged by DTS components have the following standard form:

```

{
  "_id" : "",
  "_correlationId" : "",
  "replyToChannel" : "",
  "replyEncryptionKey" : "",
  "payload" : {
    "payloadMetadata" : {
      "payloadClassKey" : "",
      "arrayPayload" : false
    }
  }
}

```

```

    },
    "payloadJSON" : ""
  },
  "error" : {
    "name" : "",
    "message" : "",
    "type" : "",
    "traceback" : ""
  }
}

```

Where the attributes represent the following:

Attribute	Value	Note
<code>_id</code>	A unique identifier for the message - generally a random UUID	The component creating the message must generate it and ensure uniqueness
<code>_correlationId</code>	The <code>_id</code> of another message that this message relates to - generally used for responses to reference requests	A component responding to a certain message must use the <code>_id</code> of the request as the <code>_correlationId</code> of its response
<code>replyToChannel</code>	The name of the Redis channel on which replies to the message are expected	A component responding to a certain message must send the response on the <code>replyToChannel</code> of the request. Conversely, when making a request, a component must set the <code>replyToChannel</code> where they expect the response.
<code>replyEncryptionKey</code>	The AES encryption key corresponding to <code>replyToChannel</code>	<p>This parameter is only used when communications are secured. Its purpose is for the requesting party to communicate to the responder how to encrypt messages that will be sent on a newly opened channel. It only needs to be included for newly opened channels or when a change of encryption key is desired.</p> <p>Any component receiving a request that includes the <code>replyEncryptionKey</code> must keep it stored and always use it for messages sent on <code>replyToChannel</code>.</p>

payload			Wrapper for the actual payload of the message	
	payloadMeta		Wrapper for the metadata of the message's payload	
		payloadTypeKey	The DTS Type Key of the payload	This is the key by which DTS components know what class to use for deserializing the payload.
		payloadIsArray	Flag that communicates whether the payload is a single object or an array	
	payloadJSON		The actual payload in JSON format	The payload json is actually wrapped in a string. This means that all " (double quote) characters need to be escaped.
error			Wrapper for an error object	Should only exist on responses when an error happened while processing the request.
	name		The name of the error	
	message		The complete error message / description	
	type		The type of error	
	traceback		The stack trace of the error	

 If a certain attribute is unknown or not applicable for a given message, it must be omitted. Including an attribute that is an empty string can have unforeseen consequences.

## Client Commands

Client Commands are Request-Response entities that facilitate all the operations that a Client can perform within DTS. Requests and Responses are marshalled over the ICB wrapped in messages in the format above where they occupy the **payload** -> **payloadJSON** field.

The commands relevant to the Client are:

- [Registration Command](#)  278
- [Project Command](#)  280
- [Connector Command](#)  282

- [Record Stream Command](#)<sup>[284]</sup>
- [Execute Remote Command](#)<sup>[286]</sup>

## Standard Sequence

The standard sequence for a Client to connect to the DTS Core and make requests is the following:

### • Register

- [Secure Only] Build an *Authentication Package* using the component certificate, component private key and controller public key (see [Registration and Authentication](#)<sup>[248]</sup>).
- Send a **Registration Command Request** (including the *Authentication Package* if needed) with a temporary *replyToChannel*.
- Read the **Registration Command Response (RCResp)** from the *replyToChannel*.
- [Secure Only] Decrypt the signature and keys provided in the **RCResp** using the component private key.
- [Secure Only] Verify that the signature in the **RCResp** matches the signature sent in the *Authentication Package*.
- Register the Redis channel provided by the **RCResp** (hereafter referred to as *componentChannel*) and start a reader process on it.
- [Secure Only] assign the *controllerKey* from **RCResp** to be used to encrypt all messages sent on **DTS:MIDDLEWARE:CHANNEL**.
- [Secure Only] assign the *componentKey* from **RCResp** to be used to decrypt all messages received on the channel provided by **RCResp**.
- [Optional] Setup a handler for unprompted **Record Stream Command Responses** received on the *componentChannel* (stream timeouts).

### • Open a project

- Send a **Project Command Request** to *Open* the desired project (use *componentChannel* as *replyToChannel* for convenience).
- Read the **Project Command Response** from the *replyToChannel*.
- [Optional] Store or verify the list of connector names

### • Receive connector entries

- Handle each **Connector Command Request** received on the *componentChannel* by constructing any structures you need based on the metadata the requests provide.
- Send **Connector Command Responses** to the request's *replyToChannel* to acknowledge each connector entry. Use the *error* attribute of the response message if something went wrong on the client end.

- [Optional] Wait for the required connectors to become available and/or implement some timeouts.

- **Use**

- Send **Record Stream Command Requests** and/or **Execute Remote Command Requests** as desired.
- **Records Stream Command Responses** for *Open* Stream Commands will include the channel where *Advance* commands for the opened stream should be sent ([Secure Only] they will also include the corresponding AES key). Make sure to use these when sending *Advance* Stream Requests.

- **Shutdown**

- Send a **Project Command Request** to *Close* any open projects.

### 7.1.4.1 Registration Command

The Registration Command is the first message any DTS component must send to the Controller. Its purpose is to make the component known to the Controller, to request further communication parameters and to authenticate if the DTS environment is secured.

#### Request

*Outgoing*

*Target channel*                      "DTS:MIDDLEWARE:CHANNEL"

*Payload Class Key*                "DTS\_REGISTER\_COMPONENT\_REQ\_COMMAND"

```
{
  "projectName" : "",
  "sessionName" : "",
  "authentication" : "",
  "component" : {
    "componentType" : "",
    "category" : "",
    "variety" : "",
    "connectorName" : "",
    "threadCount" : "",
    "metadataProducer" : false
  }
}
```

Attribute	Value	Note
-----------	-------	------

<b>projectName</b>	The name of the project this component will serve	Not applicable for Client Registration
<b>sessionName</b>	The human readable name of this session	Standard for clients is "[given_name]@[hostname]"
<b>authentication</b>	The authentication package	Only required for secured deployments - content must be in the format shown in <a href="#">Registration and Authentication</a> <sup>248</sup>
<b>component</b>	Wrapper for the component description	
	<b>componentType</b>	The type of component being registered
	<b>category</b>	For clients, this is "DTS : CLIENT : CONSUMER"
	<b>variety</b>	For clients, this is "DTS : INTERNAL"
	<b>connectorName</b>	Not applicable for Client Registration
	<b>threadCount</b>	Not applicable for Client Registration
	<b>metadataProd</b>	Not applicable for Client Registration

## Response

Incoming

Target channel: request replyToChannel

Payload Class Key: "DTS\_REGISTER\_COMPONENT\_RESPONSE"

```
{
  "id" : "",
  "channel" : "",
  "controllerKey" : "",
  "componentKey" : "",
  "securitySignature" : ""
}
```

Attribute	Value	Note
<b>id</b>	The id the Controller assigns to the Component	
<b>channel</b>	The Redis channel the Controller assigns to the component	This is the channel that the Controller will use to send requests to the component from here on
<b>controllerKey</b>	The AES key for the controller channel (DTS:MIDDLEWARE:CHANNEL) - encrypted using the component's public key and Base64 encoded	Only for secured deployments. All messages sent by the client on the controller channel will need to be encrypted using this key
<b>componentKey</b>	The AES key for the component's channel (i.e. channel) - encrypted using the component's public key and Base64 encoded	Only for secured deployments. All messages sent to the client on its channel will be encrypted using this key
<b>securitySignature</b>	The RSA signature included in the request's authentication package - encrypted using the component's public key and Base64 encoded	Only for secured deployments. The purpose of returning the security signature in this way is for the component to know it is dealing with an authentic Controller

#### 7.1.4.2 Project Command

Project Commands are perform operations on DTS projects. Clients can use Project Commands to Open and Close projects.

- The Open Project Command is sent by the Client to the Controller to request access to resources belonging to a specific DTS project. It also triggers the Controller to initialize producers and setup support structures for managing and serving requests for that project.
- The Close Project Command is sent by the Client to the Controller to inform it that it no longer needs a certain project. The Controller will only discard the project resources if no Clients require them for a set amount of time.

## Request

*Outgoing*

Target channel                    **"DTS:MIDDLEWARE:CHANNEL"**

Payload Class Key                **"DTS\_PRJ\_COMMAND\_REQ"**

```
{  
    "name" : "",  
    "projectName" : ""  
}
```

Attribute	Value	Note
<b>name</b>	The name of the command	For Clients, either "DTS_PRJ_OPEN_COMMAND" or "DTS_PRJ_CLOSE_COMMAND"
<b>projectName</b>	The name of the project the command targets	

## Response

*Incoming*

Target channel:                  request *replyToChannel*

Payload Class Key:              **"STRING"** or array of **"STRING"**

Open Project Commands are responded to with an array of strings representing the names of the connectors the project contains. This does not imply that the connectors are now ready for usage, simply that they are recognized and are being initialized if they are not yet available. Each connector's availability is announced to the client by an individual Connector Command, outlined below.

Close Project Commands are simply responded to with the **"DONE"** acknowledgment string.

For error responses, the payload will be **null** and the error details will be included in the response message's *error* attribute.

### 7.1.4.3 Connector Command

Connector Commands are used throughout DTS to activate and deactivate connectors in components. From a Client's perspective, it will receive Connector Commands to signal the Activation or Deactivation of connectors targeted by the projects currently open in the Client. It may also receive Kick Connector Commands to be notified that it no longer has access to a certain project.

Activation commands will contain the full description of the Connector, including descriptors for all collections, routines and types involved.

#### Request

*Incoming*

<i>Target channel</i>	channel assigned to Client with the Registration Command Response - <i>componentChannel</i>
<i>Payload Class Key</i>	"DTS_CONNECTOR_COMMAND"

```
{
  "name" : "",
  "projectName" : "",
  "connectorEntry" : {
    "name" : "",
    "type" : "",
    "channel" : "",
    "replicaCount" : "",
    "parametersContainer" : {
      "development" : {},
      "testing" : {},
      "production" : {}
    },
    "collections" : [
      <CollectionResources>
    ],
    "remotes" : [
      <RemoteCallResources>
    ],
    "types" : [
      <TypeResources>
    ]
  },
  "reconnect" : false,
  "reason" : ""
}
```

Attribute		Value	Note
<b>name</b>		The name of the command	"ACTIVATE_CONNECTOR", "DEACTIVATE_CONNECTOR" or "KICK"
<b>projectName</b>		The name of the project that is the subject of the command	
<b>connectorEntry</b>		Wrapper for the descriptor of the connector that is the subject of the command	Not used for KICK commands
	<b>name</b>	The name of the connector	
	<b>type</b>	The type code of the connector	
	<b>channel</b>	Channel assigned to the connector	Not applicable to Client
	<b>replicaCount</b>	Maximum number of producers	Not applicable to Client
	<b>parametersContainer</b>	Wrapper for the connection parameters to a connector's data source.	Not applicable to Client
	<b>collections</b>	The list of collections that the connector can serve as <a href="#">CollectionResource</a> <sup>[288]</sup> objects	These are full descriptions of the collections served, with fields, types and constraints (only present on activation)
	<b>remotes</b>	The list of remote calls that the connector can serve as <a href="#">RemoteCallResource</a> <sup>[288]</sup> objects	These describe the details of the native routines that can be accessed with full descriptions of inputs and outputs (only present on activation)
	<b>types</b>	The list of non-trivial data types that the connector relies on as <a href="#">TypeResource</a> <sup>[288]</sup> objects	These are full descriptions of any structure types present in any fields or parameters served by the connector (only present on activation)
<b>reconnect</b>		Only for KICK commands - whether or not a reconnect should be attempted	
<b>reason</b>		Only for KICK commands - the reason for being kicked	

## Response

*Outgoing*

*Target channel* Request *replyToChannel*

*Payload Class Key* "STRING"

The response the client must provide to a Connector Command Request is the "DONE" acknowledgment string. If the request generated errors, the response should be `null` and the error should be included in the response message's *error* attribute.

### 7.1.4.4 Record Stream Command

Record Stream Commands are used to manage and consume DTS record streams. Record Stream Commands emitted by the Client can Open, Close or Advance a stream.

## Request

*Outgoing*

*Target channel* "DTS:MIDDLEWARE:CHANNEL" for Open and Close / the stream producer's channel for Advance

*Payload Class Key* "DTS\_RECORD\_REQ\_COMMAND"

```
{
  "projectName" : "",
  "connectorEntry" : {
    "name" : ""
  },
  "collectionKey" : "",
  "operation" : 0,
  "filter" : <DTSPredicate>,
  "streamKey" : "",
  "batchSize" : 0
}
```

Attribute	Value	Note
-----------	-------	------

<b>projectName</b>	The name of the project containing the resource	
<b>connectorEntry -&gt; name</b>	The name of the connector serving the resource	
<b>collectionKey</b>	The name of the collection resource	
<b>operation</b>	The operation to execute (1 - Open, 3 - Close, 5 - Advance)	
<b>filter</b>	The <a href="#">DTSPredicate</a> <sup>239</sup> to use for building the query that generates the record stream	
<b>streamKey</b>	The stream's identification key (not used for Open)	The streamKey is generated by the stream producers after opening the stream and is communicated to the Client in the response to the Open request.
<b>batchSize</b>	The number of records to be served (not used for Close)	

## Response

Incoming

Target channel

Request *replyToChannel*

Payload Class Key

**"DTS\_STREAM\_RECORD\_CONTAINER"**

```
{
  "streamId" : "",
  "moreToGet" : false,
  "channel" : "",
  "channelKey" : "",
  "records" : [
    <StreamRecord>
  ]
}
```

Attribute	Value	Note
-----------	-------	------

<b>streamId</b>	The key or id of the stream	
<b>moreToGet</b>	Whether the stream has more records after these	It is <b>true</b> or <b>false</b> for responses to Open and Advance Commands and it is <b>null</b> to signify a completed Close command
<b>channel</b>	The stream producer's channel to send subsequent Advance commands to (only for Open command responses)	
<b>channelKey</b>	The AES encryption key for <i>channel</i>	Only for secured deployments
<b>records</b>	The list of requested records	Will be of the length set in the request's <i>batchSize</i> attribute or all that is left in the stream if less than that.  The format of the returned records is described by the respective <a href="#">CollectionResource</a> <sup>288</sup>

**i** Record Stream Command Responses can be received even without making an explicit request. This happens whenever a stream times out and is closed.

#### 7.1.4.5 Execute Remote Command

Execute Remote Commands are used to invoke routines. The Client addresses the requests to the Controller and the responses come from the serving producers.

##### Request

Outgoing

Target channel **"DTS:MIDDLEWARE:CHANNEL"**

Payload Class Key **"DTS\_EXECUTE\_REMOTE\_COMMAND"**

```
{
  "projectName" : "",
  "connectorEntry" : {
    "name" : ""
  }
}
```

```

    },
    "remoteKey" : "",
    "parameters" : {
        "payloadMetadata" : {
            "payloadClassKey" : "",
            "arrayPayload" : false
        },
        "payloadJSON" : ""
    }
}

```

Attribute		Value	Note
<b>projectName</b>		The name of the project containing the resource	
<b>connectorEntry -&gt; name</b>		The name of the connector serving the resource	
<b>remoteKey</b>		The name of the remote call resource	
<b>parameters</b>		A payload container to wrap the remote call's input parameters	
	<b>payloadMetadata</b>	The metadata for the payload	
	<b>payloadClassKey</b>	The type key of the remote's arguments wrapper class	The TypeDescriptor for this type key can be found in the <a href="#">RemoteCallDescriptor</a> <sup>288</sup> corresponding to the remote call.
	<b>arrayPayload</b>	Always false as the arguments are wrapped in a single object	
	<b>payloadJSON</b>	The JSON representing the arguments wrapper object in the format referenced by <i>payloadClassKey</i>	The payload json is actually wrapped in a string. This means that all " (double quote) characters need to be escaped.

## Response

Incoming

Target channel

Request *replyToChannel*

*Payload Class Key*

The type key of the remote call's results wrapper (described in the [RemoteCallDescriptor](#)<sup>[288]</sup>) or `"DTS_STREAM_RECORD_CONTAINER"`

If it is a regular routine, the response will directly contain the outputs of the routine, wrapped in an object with each output as a named attribute.

If the routine returns a stream, the response will be the same as a [Record Stream Command](#)<sup>[284]</sup> Response, containing the details of the resulting stream.

## 7.2 Producer

Custom producer development is not yet ready for public access - we're still working on our APIs.

However, if you have a particular data source that you need DTS to access, please contact us for a solution.

## 7.3 Project Artifacts

DTS stores a number of artifacts used to describe structures used in its projects in its internal repository (Mongo). When developing extensions to DTS or tracing certain issues during administration, it is useful to have a reference on the formats and purposes of these artifacts.










DTS uses the DTS database in its internal Mongo deployment to store the following artifact collections:

### • Foundation Artifacts





- [CONNECTOR\\_CATEGORY](#)<sup>[290]</sup>
- [CONNECTOR\\_TYPE](#)<sup>[291]</sup>
- [NOTIFICATION\\_SENDER](#)<sup>[293]</sup>
- [WEBSERVICE\\_DEPLOYER](#)<sup>[294]</sup>

### • Design Artifacts









- [PROJECT\\_WIP](#)<sup>[295]</sup>

- [PROJECT\\_RESOURCE](#)  296
- [PROJECT\\_OPERATION](#)  297
- [CONNECTOR](#)  298
- [COLLECTION\\_RESOURCE](#)  300
- [COLLECTION\\_DETAILS](#)  301
- [REMOTE\\_CALL\\_RESOURCE](#)  302
- [REMOTE\\_CALL\\_DETAILS](#)  303
- [AGGREGATE](#)  306
- [WEBSERVICE](#)  307

#### • Runtime Artifacts

- [PROJECT](#)  308
- [PROJECT\\_STATUS](#)  309
- [PRODUCER\\_STATUS](#)  310
- [CONSUMER\\_STATUS](#)  311

Certain substructures are used in multiple collections and are detailed separately:

- [NotificationsConfig](#)  313
- [NotificationTarget](#)  313
- [AttributeDescriptor](#)  314
- [ConstraintDefinition](#)  315
- [TypeResource](#)  316
- [AggregateSource](#)  316
- [AggregateRelationship](#)  317
- [WebserviceResource](#)  318


 Some of these structures or their substructures are also passed within commands between DTS components.

### 7.3.1 CONNECTOR\_CATEGORY

The `CONNECTOR_CATEGORY` collection holds the base (built-in) definitions for DTS connectors. Its content is predefined and maintained by the DTS Core and should not be altered.

```
{
  "category" : "",
  "imageName" : "",
  "bladeJar" : "",
  "build" : "",
  "requiredJars" : [],
  "environment" : {},
  "volumes" : {}
}
```

Attribute	Value	Note
<b>category</b>	The category of the connector type, also the type of data source. Must be one of the supported data source types.  e.g.: "Oracle"	The category is used to identify base connection parameters and to find specific connector implementation libraries.
<b>imageName</b>	The name of the Docker image to use as a base for producers of this connector type.  e.g.: "registry/dts-producer-2021.1"	Should be the tag by which the image is known in the Docker engine.
<b>bladeJar</b>	The path of the main jar for this connector type's implementation library (blade) relative to the blades directory  e.g.: "Oracle/dts-blade-oracle-2021.1.jar"	If the blade is contained in the Docker image, this is omitted (e.g. the Smallworld connector types).
<b>requiredJars</b>	A list of all the jars that bladeJar requires as paths relative to the blades directory.  e.g.: ["Oracle/dts-blade-oracle-2021.1.jar", "Oracle/ojdbc8-19.3.0.0.jar",	

	<code>"Oracle/orai18n-19.3.0.0.jar", "Oracle/jdbc"]</code>	
<b>environment</b>	A map of environment variables to pass into the producer's Docker container as key-value pairs.  e.g: <code>{"DTS_DEBUG_LOGGING" : "true", "SOMETHING_ELSE" : "1234"}</code>	
<b>volumes</b>	A map of volumes between the Docker host and the producer container as key-value pairs where the key is the host directory and the value is the container directory.  e.g.: <code>={"/dts/stuff" : "/usr/local/stuff"}</code>	 <b>Mongo does not accept the period (.) character as a map key and it cannot be escaped. As such, DTS does not support mapping directories whose path on the Docker host contains a period (.) character.</b>


### 7.3.2 CONNECTOR\_TYPE

The `CONNECTOR_TYPE` collection holds all connector type definitions for a DTS deployment. This is where the Type choices when [adding a connector in the WebApp](#)<sup>[44]</sup> come from. Connector Types can be defined using the [Connector Types dialog](#)<sup>[104]</sup> or by directly interacting with documents in the collection.

```
{
  "category" : "",
  "variety" : "",
  "type" : "",
  "imageName" : "",
  "bladeJar" : "",
  "build" : "",
  "requiredJars" : [],
  "environment" : {},
  "volumes" : {},
  "parameters" : []
}
```

Attribute	Value	Note
-----------	-------	------

<b>category</b>	The category of the connector type, also the type of data source. Must be one of the supported data source types.  e.g.: "Oracle"	The category is used to identify base connection parameters and to find specific connector implementation libraries.
<b>variety</b>	The name of the variation of <i>category</i> , if any.  e.g.: "PDB"	
<b>type</b>	The name of the connector type as it will appear in the WebApp.  e.g.: "Oracle:PDB"	Convention is to use <i>category:variety</i>
<b>imageName</b>	The name of the Docker image to use as a base for producers of this connector type.  e.g.: "registry/dts-producer-2021.1"	Should be the tag by which the image is known in the Docker engine.
<b>bladeJar</b>	The path of the main jar for this this connector type's implementation library (blade) relative to the blades directory  e.g.: "Oracle/dts-blade-oracle-2021.1.jar"	If the blade is contained in the Docker image, this is omitted (e.g. the Smallworld connector types).
<b>requiredJars</b>	A list of all the jars that bladeJar requires as paths relative to the blades directory.  e.g.: ["Oracle/dts-blade-oracle-2021.1.jar", "Oracle/ojdbc8-19.3.0.0.jar", "Oracle/orai18n-19.3.0.0.jar", "Oracle/jdbc"]	
<b>environment</b>	A map of environment variables to pass into the producer's Docker container as key-value pairs.	

	e.g: <code>{"DTS_DEBUG_LOGGING" : "true", "SOMETHING_ELSE" : "1234"}</code>	
<b>volumes</b>	A map of volumes between the Docker host and the producer container as key-value pairs where the key is the host directory and the value is the container directory.  e.g.: <code>={"/dts/stuff" : "/usr/local/stuff"}</code>	 <b>Mongo does not accept the period (.) character as a map key and it cannot be escaped. As such, DTS does not support mapping directories whose path on the Docker host contains a period (.) character.</b>

 The `category`, `imageName`, `bladeJar`, `requiredJars`, `environment` and `volumes` attributes are directly inherited from the referenced [CONNECTOR\\_CATEGORY](#)<sup>290</sup>.

 The `blades` directory in a producer container is by default `/usr/local/blades`, but can be customized by setting the `DTS_BLADES_PATH` environment variable.

### 7.3.3 NOTIFICATION\_SENDER

The `NOTIFICATION_SENDER` collection holds the SMTP parameters of all configured accounts that will be used for sending DTS notifications. Notification senders can be defined using the [Notification Senders dialog](#)<sup>109</sup> in the Web UI.

```
{
  "address" : "",
  "host" : "",
  "port" : 0,
  "username" : "",
  "password" : "",
  "encryption" : ""
}
```

Attribute	Value	Note
<b>address</b>	The full email address of the sender.	This value will be used to identify the sender when configuring a project's notifications.
<b>host</b>	The email server host name.  e.g.: <code>"smtp.gmail.com"</code>	

<b>port</b>	The email server port. e.g.: 587	
<b>username</b>	The SMTP username for the email account.	
<b>password</b>	The SMTP password for the email account.	
<b>encryption</b>	The type of encryption to be used. "none" / "SSL" / "STARTTL"	

### 7.3.4 WEBSERVICE\_DEPLOYER

The WEBSERVICE\_DEPLOYER collection stores definitions for DTS WebService deployment methods. The deployers are created and edited using the [Webservice Deployers](#)<sup>[97]</sup> dialog in the Web UI.

```
{
  "name" : "",
  "type" : "",
  "hostname" : "",
  "port" : "",
  "hostUsername" : "",
  "hostPassword" : "",
  "copyPath" : "",
  "appservUsername" : "",
  "appservPassword" : ""
}
```

Attribute	Value	Note
<b>name</b>	The name given to the deployer.	It is the unique identifier used for the deployer throughout DTS.
<b>type</b>	The type of deployment. "SCP" / "Samba" / "JBossCLI" / "TomcatHTTP" / "Weblogic" / "Websphere" / "Local"	Each deployment type has a specific method and requires a specific subset of the other attributes.

<b>hostname</b>	The host name for the deployment target (usually the Application Server).	Required by all deployment types, except "Local"
<b>port</b>	The port used for invoking the Application Server's deployment API.	Required only for "JBossCLI" and "TomcatHTTP"
<b>hostUsername</b>	The username to connect to the deployment target host (usually the Application Server).	Required for "SCP", "Samba", "TomcatHTTP", "Weblogic", "Websphere"
<b>hostPassword</b>	The password for <b>hostUsername</b>	Same as <b>hostUsername</b>
<b>copyPath</b>	The path to copy the DTS WARs to on <b>hostname</b> .	Required for all deployment types, except "JBossCLI"
<b>appservUsername</b>	An admin user name for the Application Server.	Required for "JBossCLI", "TomcatHTTP", "Weblogic", "Websphere"
<b>appservPassword</b>	The password for <b>appservUsername</b>	Same as <b>appservUsername</b>

 The precise methods each deployer type uses are documented in the [Webservice Deployers](#) <sup>97</sup> section.

### 7.3.5 PROJECT\_WIP

The PROJECT\_WIP collection holds stubs for all DTS projects in design phase. Published projects retain their design stub as well, to be used for future editing.

```
{
  "_id" : "",
  "name" : "",
  "createdAt" : 1970-01-01 00:00:00.000z,
  "version" : 0,
  "wip" : 0,
  "notifications" : []
}
```

Attribute	Value	Note
<b>_id</b>	A unique identifier for the design project.	Artifacts belonging to this project stored in other collections will use this id to reference the project.
<b>name</b>	The given name of the project	
<b>createdAt</b>	The time when the project was created	
<b>version</b>	The major version of the project	Increments after the project is published
<b>wip</b>	The minor (current design) version of the project.	Increments when changes are made to the design project and resets after the project is published.
<b>notifications</b>	A list of notification recipients for this project, in the form of <a href="#">NotificationsConfigs</a> <sup>313</sup>	

### 7.3.6 PROJECT\_RESOURCE

The PROJECT\_RESOURCE collection holds references to all resources currently enabled in existing design projects ([PROJECT\\_WIP](#)<sup>295</sup>). Enabling/disabling a collection or routine in the WebApp directly adds/removes entries from this collection.

```
{
  "_id" : "",
  "nativeIdentifier" : "",
  "projectId" : "",
  "connectorId" : "",
  "type" : "",
  "fields" : []
}
```

Attribute	Value	Note
<b>_id</b>	A unique identifier for the project resource.	
<b>nativeIdentifier</b>	The full native name of the resource	

<b>projectId</b>	Reference to the design project ( <a href="#">PROJECT_WIP</a> <sup>[295]</sup> entry) this resource belongs to.	
<b>connectorId</b>	Reference to the connector ( <a href="#">CONNECTOR</a> <sup>[298]</sup> entry) this resource is accessed through.	
<b>type</b>	The type of resource	"collection" (collection, table, view, etc.)  "api" (function, method, procedure, etc.)
<b>fields</b>	A list containing the nativeIdentifiers of all the fields in the resource that are included in the Project.	Only for "collection" resources.

### 7.3.7 PROJECT\_OPERATION

The PROJECT\_OPERATION collection contains a log of all operations performed on DTS projects in the form of the request headers and the resolution.

```
{
  "_id" : "",
  "key" : "",
  "projectName" : "",
  "payload" : {
    "payloadRecord" : {},
    "payloadMetadata" : {
      "payloadClassKey" : "",
      "arrayPayload" : false
    }
  },
  "status" : "",
  "createdAt" : 1970-01-01 00:00:00.000Z,
  "completedAt" : 1970-01-01 00:00:00.000Z
}
```

Attribute	Value	Note
<b>_id</b>	A unique identifier for the project operation.	

<b>key</b>	The <code>_id</code> of the request message for this operation.	Also the <code>_correlationId</code> of the response.
<b>payload</b>	An abridged version of the request message's payload	
<b>payloadRecord</b>	An abridged version of the actual request command (indentification info only)	
<b>payloadMetadata</b>	The metadata for the request payload	
<b>payloadClass</b>	The class key for the request command	Identifies the operation command
<b>arrayPayload</b>	Whether or not the payload was an array	<code>False</code> for all project operations
<b>status</b>	The resolution status of the command	
<b>createdAt</b>	The timestamp of the request	
<b>completedAt</b>	The timestamp of the response	

### 7.3.8 CONNECTOR

The `CONNECTOR` collection holds references to all connectors defined for current design projects ([PROJECT\\_WIP](#)<sup>295</sup>).

```
{
  "_id" : "",
  "name" : "",
  "type" : "",
  "projectId" : "",
  "createdAt" : 1970-01-01 00:00:00.000Z,
  "parametersContainer" : {
    "development" : {},
    "testing" : {},
    "production" : {}
  }
}
```

Attribute	Value	Note
<b>_id</b>	A unique identifier for the connector.	
<b>name</b>	The full native name of the resource	
<b>type</b>	The type of connector. Must match an entry from <a href="#">CONNECTOR_TYPE</a> <sup>[294]</sup> .  e.g: "Oracle:PDB"	
<b>projectId</b>	Reference to the design project ( <a href="#">PROJECT_WIP</a> <sup>[295]</sup> entry) this connector belongs to.	
<b>createdAt</b>	The time when the connector was created	
<b>parametersContain</b>	Sets of connection parameters for the connector (credentials, target schemas, etc.)  e.g.: { "username" : "dtstest", "password" : "test", "connectionString" : "jdbc:oracle:thin:@172.16 .10.212:1521/DTSTESTPDB", "schemas" : "dtstest,dtsdemo" }	Has 3 separate sections for the types of deployment.

## Published Project Usage

Objects of the CONNETOR form will also be used to describe connectors in published projects ([PROJECT](#)<sup>[308]</sup>). When employed in this way, they will also include the following attributes:

Attribute	Value	Note
<b>collections</b>	The list of all collection resources from this connector that are included in the published project, in the published form of <a href="#">COLLECTION_DETAILS</a> <sup>[301]</sup> .	The data for these items is extracted by joining design artifacts from <a href="#">PROJECT_RESOURCE</a> <sup>[296]</sup> with artifacts from <a href="#">COLLECTION_DETAILS</a> <sup>[301]</sup> .

<b>remotes</b>	The list of all remote call resources from this connector that are included in the published project, in the form of <a href="#">REMOTE_CALL_DETAILS</a> <sup>[303]</sup> .	The data for these items is extracted by joining design artifacts from <a href="#">PROJECT_RESOURCE</a> <sup>[296]</sup> with artifacts from <a href="#">REMOTE_CALL_DETAILS</a> <sup>[303]</sup> .
----------------	---	---

### 7.3.9 COLLECTION\_RESOURCE

The COLLECTION\_RESOURCE collection holds references to all collections (tables, views, etc.) identified by the connectors of existing design projects.

```
{
  "_id" : "",
  "name" : "",
  "nativeIdentifier" : "",
  "projectId" : "",
  "connectorId" : ""
}
```

Attribute	Value	Note
<b>_id</b>	A unique identifier for the collection resource.	
<b>name</b>	The given name for the collection resource	By default, it is the simple name of the datasource collection
<b>nativeIdentifier</b>	The full native name of the collection	Generally of the form <container>.<name> e.g.: "public.employees"
<b>projectId</b>	Reference to the design project ( <a href="#">PROJECT_WIP</a> <sup>[295]</sup> entry) this collection belongs to.	
<b>connectorId</b>	Reference to the connector ( <a href="#">CONNECTOR</a> <sup>[298]</sup> entry) this collection is accessed through.	

## 7.3.10 COLLECTION\_DETAILS

The COLLECTION\_DETAILS collection holds the structural details of all collections that are enabled in current design projects or have otherwise been requested.

**i** A COLLECTION\_DETAILS entry always has a corresponding COLLECTION\_RESOURCE entry and their name, nativeIdentifier, projectId and connectorId attributes always match.

```
{
  "_id" : "",
  "name" : "",
  "nativeIdentifier" : "",
  "projectId" : "",
  "connectorId" : "",
  "fields" : [],
  "constraints" : []
}
```

Attribute	Value	Note
<b>_id</b>	A unique identifier for the collection resource.	
<b>name</b>	The given name for the collection resource	By default, it is the simple name of the datasource collection
<b>nativeIdentifier</b>	The full native name of the collection	Generally of the form <container>.<name> e.g.: "public.employees"
<b>projectId</b>	Reference to the design project ( <a href="#">PROJECT_WIP</a> <sup>[295]</sup> entry) this collection belongs to.	
<b>connectorId</b>	Reference to the connector ( <a href="#">CONNECTOR</a> <sup>[298]</sup> entry) this collection is accessed through.	
<b>fields</b>	A list of all the fields available on the collection in the <a href="#">AttributeDescriptor</a> <sup>[314]</sup> format.	
<b>constraints</b>	A list of all the PK and FK constraints on the collection in the <a href="#">ConstraintDefinition</a> <sup>[315]</sup> format	

<b>predicate</b>	A <a href="#">DTSPredicate</a> <sup>[239]</sup> that defines the fundamental filter applied to this collection	Only records that match the filter will be allowed through the system. If the record request contains a query, it will be compounded (using AND) with the fundamental filter defined here.
------------------	--	--

## Published Project Usage

Objects of the COLLECTION\_DETAILS form will also be used to describe collections in the connectors of published projects.

### 7.3.11 REMOTE\_CALL\_RESOURCE

The REMOTE\_CALL\_RESOURCE collection holds references to all routines (function, procedures, methods, etc.) identified by the connectors of existing design projects.

```
{
  "_id" : "",
  "name" : "",
  "nativeIdentifier" : "",
  "projectId" : "",
  "connectorId" : ""
}
```

Attribute	Value	Note
<b>_id</b>	A unique identifier for the remote call resource.	
<b>name</b>	The given name for the remote call resource	By default, it is the simple name of the routine
<b>nativeIdentifier</b>	The full native name of the routine	Generally of the form <container>.<name>  e.g.: "public.calculate_impact"
<b>projectId</b>	Reference to the design project ( <a href="#">PROJECT_WIP</a> <sup>[295]</sup> entry) this routine belongs to.	

<b>connectorId</b>	Reference to the connector ( <a href="#">CONNECTOR</a> <sup>298</sup> entry) this routine is accessed through.	
--------------------	--	--

### 7.3.12 REMOTE\_CALL\_DETAILS

The REMOTE\_CALL\_DETAILS collection holds the structural details of all routines that are enabled in current design projects or have otherwise been requested.

**i** A REMOTE\_CALL\_DETAILS entry always has a corresponding REMOTE\_CALL\_RESOURCE entry and their name, nativeIdentifier, projectId and connectorId attributes always match.

```
{
  "_id" : "",
  "name" : "",
  "nativeIdentifier" : "",
  "projectId" : "",
  "connectorId" : "",
  "inTypeResource" : {},
  "outTypeResource" : {},
  "streamy" : false
}
```

Attribute	Value	Note
<b>_id</b>	A unique identifier for the remote call resource.	
<b>name</b>	The given name for the remote call resource	By default, it is the simple name of the routine
<b>nativeIdentifier</b>	The full native name of the routine	Generally of the form <container>.<name>  e.g.: <code>"public.calculate_impact"</code>
<b>projectId</b>	Reference to the design project ( <a href="#">PROJECT_WIP</a> <sup>295</sup> entry) this routine belongs to.	
<b>connectorId</b>	Reference to the connector ( <a href="#">CONNECTOR</a> <sup>298</sup> entry) this routine is accessed through.	

<b>inTypeResource</b>	A descriptor for the routine's inputs (arguments) in the form of a <a href="#">TypeResource</a> <sup>316</sup>	DTS models routines so that their arguments and results are wrapped in structured objects (each argument and result being an attribute in the respective wrapper). These are the TypeResources that describe those models.
<b>outTypeResource</b>	A descriptor for the routine's outputs (results) in the form of a <a href="#">TypeResource</a> <sup>316</sup>	
<b>streamy</b>	A flag showing if the remote call returns a stream or not.	

### 7.3.13 TOPIC\_RESOURCE

The TOPIC\_RESOURCE collection holds references to all topics identified by the connectors of existing design projects.

```
{
  "_id" : "",
  "name" : "",
  "nativeIdentifier" : "",
  "projectId" : "",
  "connectorId" : ""
}
```

Attribute	Value	Note
<b>_id</b>	A unique identifier for the topic resource.	
<b>name</b>	The given name for the topic resource	By default, it is the same as the nativeIdentifier
<b>nativeIdentifier</b>	The full native name of the topic	
<b>projectId</b>	Reference to the design project ( <a href="#">PROJECT_WIP</a> <sup>295</sup> entry) this collection belongs to.	
<b>connectorId</b>	Reference to the connector ( <a href="#">CONNECTOR</a> <sup>298</sup> entry) this collection is accessed through.	

## 7.3.14 TOPIC\_DETAILS

The TOPIC\_DETAILS collection holds the structural details of all topics that are enabled in current design projects or have otherwise been requested.

**i** A TOPIC\_DETAILS entry always has a corresponding TOPIC\_RESOURCE entry and their name, nativeIdentifier, projectId and connectorId attributes always match.

```
{
  "_id" : "",
  "name" : "",
  "nativeIdentifier" : "",
  "projectId" : "",
  "connectorId" : "",
  "partitions" : [],
  "keyType" : "",
  "messageType" : "",
  "readEnabled" : false,
  "writeEnabled" : false,
  "properties" : {},
  "recordTypeDescriptor" : {}
}
```

Attribute	Value	Note
<b>_id</b>	A unique identifier for the topic resource.	
<b>name</b>	The given name for the topic resource	By default, it is the same as the nativeIdentifier
<b>nativeIdentifier</b>	The full native name of the topic	
<b>projectId</b>	Reference to the design project ( <a href="#">PROJECT_WIP</a> <sup>295</sup> entry) this collection belongs to.	
<b>connectorId</b>	Reference to the connector ( <a href="#">CONNECTOR</a> <sup>298</sup> entry) this collection is accessed through.	
<b>partitions</b>	A list of the partition ids in the Topic (int array)	
<b>keyType</b>	The selected (de)serialization type for the topic's keys	

<b>messageType</b>	The selected (de)serialization type for the topic's messages	
<b>readEnabled</b>	Whether reading from the Topic via DTS is enabled	
<b>writeEnabled</b>	Whether writing to the Topic via DTS is enabled	
<b>properties</b>	A map of properties specified for the Topic	
<b>recordTypeDescription</b>	A full <a href="#">TypeResource</a> <sup>316</sup> describing the records that will be streamed for the Topic	

## Published Project Usage

Objects of the TOPIC\_DETAILS form will also be used to describe collections in the connectors of published projects.

### 7.3.15 AGGREGATE

The AGGREGATE collection holds the Aggregate definitions for DTS Projects. They can be created in the WebApp, using the [Aggregates Page](#)<sup>59</sup>.

 For more information on this functionality, please see [Aggregation](#)<sup>242</sup>.

```
{
  "name" : "",
  "mainSourceIdentifier" : "",
  "mainSourceName" : "",
  "mainSourceConnector" : "",
  "projectId" : "",
  "sources" : [],
  "relationships" : []
}
```

Attribute	Value	Note
<b>name</b>	The given name for the collection resource	By default, it is the simple name of the datasource collection

<b>mainSourceIdentif</b>	The full native name of the main source	Generally of the form <container>.<name>  e.g.: "public.employees"
<b>mainSourceName</b>	The DTS name for the main source	
<b>mainSourceConnect</b>	The name of the connector definition where the main source is from	
<b>projectId</b>	Reference to the design project ( <a href="#">PROJECT_WIP</a> <sup>295</sup> entry) this collection belongs to.	
<b>sources</b>	The list of sources included in the Aggregate, in the form of <a href="#">AggregateSources</a> <sup>316</sup>	
<b>relationships</b>	The list of relationships between the Aggregate's sources, in the form of <a href="#">AggregateRelationships</a> <sup>317</sup>	

### 7.3.16 WEBSERVICE

The WEBSERVICE collection holds definitions for webservices designed to serve DTS projects. The reflect the work in the [Webservices section](#)<sup>70</sup> of the Web UI.

```
{
  "webserviceName" : "",
  "mode" : "",
  "projectId" : "",
  "connectors" : []
}
```

Attribute	Value	Note
<b>webserviceName</b>	The given name for the webservice.	
<b>mode</b>	The type of webservice this models.  "REST" / "SOAP"	

<b>projectId</b>	The id of the project this webservice definition belongs to.	
<b>connectors</b>	The list of resources included in the webservice, in the form of <a href="#">WebserviceResources</a> <sup>318</sup>	

### 7.3.17 PROJECT

The PROJECT collection houses all published (i.e. usable) projects. While design artifact are spread over multiple collections, each PROJECT entry brings together all the artifacts relevant to a given project in a single document.

```
{
  "_id" : "",
  "wid" : "",
  "version" : 0,
  "wip" : 0,
  "name" : "",
  "connectors" : [],
  "aggregates" : [],
  "webservices" : [],
  "notifications" : []
}
```

Attribute	Value	Note
<b>_id</b>	A unique identifier for the published project	Not the same as the design project <i>_id</i>
<b>wid</b>	The <i>_id</i> of the corresponding design project ( <a href="#">PROJECT_WIP</a> <sup>295</sup> )	
<b>version</b>	The major version of the published project	The values of <i>version</i> and <i>wip</i> the design project was on before publishing.
<b>wip</b>	The minor version of the published project	
<b>name</b>	The name of the published project	The same as the name of the design project or a variation denoting a Webservice-specific publishing (*).

<b>connectors</b>	The list of connectors in the project in the published form of <a href="#">CONNECTOR</a> <sup>298</sup>	
<b>aggregates</b>	The list of aggregates defined in the project, in the published form of <a href="#">AGGREGATE</a> <sup>306</sup>	
<b>webservices</b>	The list of webservices defined in the project, in the published form of <a href="#">WEBSERVICE</a> <sup>307</sup>	
<b>notifications</b>	The list of notification routes defined in the project, in the published form of <a href="#">NOTIFICATIONS</a> <sup>313</sup>	

**i (\*)** Each time a Webservice WAR is generated, a snapshot of the Project it belongs to is saved in the PROJECT collection.

This is due to the fact that the generated WAR contains hardcoded resources referencing project artifacts. These can change in the base PROJECT entry when a new version is published, so a snapshot is kept so that the Webservice does not risk becoming inoperative.

### 7.3.18 PROJECT\_STATUS

The PROJECT\_STATUS collection houses real-time information on the operating status of published projects ([PROJECT](#)<sup>308</sup>).

```
{
  "projectName" : "",
  "active" : false,
  "started" : "",
  "version" : 0,
  "totalCalls" : 0,
  "totalStreams" : 0,
  "connectorNames" : []
}
```

Attribute	Value	Note
<b>projectName</b>	The name of the published project this status references	

<b>active</b>	Whether or not the project is currently active	A project becomes active when a consumer asks for it or it is manually activated and it becomes inactive after a certain time with no consumers or if it is manually stopped.
<b>started</b>	The time at which the project was last activated	
<b>version</b>	The version of the project that is currently activated	This may differ from the published project version if the project has been active while a new version was published.
<b>totalCalls</b>	The total number of calls the project has served.	
<b>totalStreams</b>	The total number of streams the project has served.	
<b>connectorNames</b>	A list of the names of all the connectors in this project for the active version.	This list is used in the Dashboard to keep track of producers for connectors that have possibly been removed or renamed in versions of the project newer than the active one.

**i** Together with [PRODUCER\\_STATUS](#)<sup>[310]</sup> and [CONSUMER\\_STATUS](#)<sup>[311]</sup>, this collection is what feeds real-time information in the WebApp's Dashboard.

### 7.3.19 PRODUCER\_STATUS

The PRODUCER\_STATUS collection houses real-time information on the operating status of currently running producers.

```
{
  "componentId" : "",
  "projectName" : "",
  "connectorName" : "",
  "sessionName" : false,
  "started" : "",
  "status" : 0,
  "isAggregator" : false,
  "activeCalls" : "",
```

```

    "activeStreams" : "",
    "totalCalls" : 0,
    "totalStreams" : 0
}

```

Attribute	Value	Note
<b>componentId</b>	The component identifier assigned to this producer	
<b>projectName</b>	The name of the published project this producer serves	
<b>connectorName</b>	The name of the connector the producer belongs to.	
<b>sessionName</b>	The self-generated name of the producer's session	Typically, this is the name of the producer's Docker container.
<b>started</b>	The time at which the producer was started	
<b>status</b>	The current status of the producer	
<b>isAggregator</b>	A flag that says if this status entry is for a connector producer, or for an aggregator.	
<b>activeCalls</b>	The number of remote calls currently being served	
<b>activeStreams</b>	The number of streams currently being served	
<b>totalCalls</b>	The total number of calls the producer has served.	
<b>totalStreams</b>	The total number of streams the producer has served.	

### 7.3.20 CONSUMER\_STATUS

The CONSUMER\_STATUS collection houses real-time information on the operating status of currently running consumers (clients).

```

{
    "componentId" : "",
    "projectId" : "",
    "sessionName" : false,
    "started" : "",
    "status" : 0,
    "isAggregator" : false,
    "activeCalls" : "",
    "activeStreams" : "",
    "totalCalls" : 0,
    "totalStreams" : 0
}

```

Attribute	Value	Note
<b>_id</b>	A unique identifier for this status entry	
<b>componentId</b>	The component identifier assigned to this consumer	
<b>projectId</b>	The name of the published project this consumer targets	
<b>sessionName</b>	The self-generated name of the consumer's session	Typically, this is of the form <given_name>@<hostname>
<b>started</b>	The time at which the consumer was registered	
<b>status</b>	The current status of the consumer	
<b>isAggregator</b>	A flag that says if this status entry is for a regular client, or for an aggregator's client.	
<b>activeCalls</b>	The number of remote calls currently being accessed	
<b>activeStreams</b>	The number of streams currently being accessed	
<b>totalCalls</b>	The total number of calls the consumer has accessed	
<b>totalStreams</b>	The total number of streams the consumer has accessed	

### 7.3.21 NotificationConfig

NotificationConfigs model notification configurations for registered projects. They are the direct result of the Project Notifications page in the Web UI.

```
{
  "senderId" : "",
  "projectName" : "",
  "targets" : []
}
```

Attribute	Value	Note
<b>senderId</b>	The identifier for the notification sender account.	This is the <i>address</i> field of the respective entry in <a href="#">NOTIFICATION_SENDER</a> <sup>293</sup> .
<b>projectName</b>	The name of the project these notification settings are for.	
<b>targets</b>	A list of recipients of notifications for this project, in the form of <a href="#">NotificationTargets</a> <sup>313</sup> .	

### 7.3.22 NotificationTarget

NotificationTargets model recipients for project notifications and their settings.

```
{
  "address" : "",
  "componentFailureEnabled" : false,
  "agentFailureEnabled" : false,
  "licenceLimitReachedEnabled" : false
}
```

Attribute	Value	Note
<b>address</b>	The email address of the notification recipient.	

<b>componentFailureEnabled</b>	Whether the recipient will receive notifications when a component (producer or aggregator) fails.	
<b>agentFailureEnabled</b>	Whether the recipient will receive notifications when a the DTS agent fails.	
<b>licenceLimitReached</b>	Whether the recipient will receive notifications when the limit of concurrent calls or streams set by the licence is reached.	

### 7.3.23 AttributeDescriptor

AttributeDescriptors model the details of resource attributes - either collection fields (table columns), routine arguments/results or structured type members.

```
{
  "nativeIdentifier" : "",
  "name" : "",
  "nativeTypeIdentifier" : "",
  "type" : "",
  "array" : false,
  "extra" : {}
}
```

Attribute	Value	Note
<b>nativeIdentifier</b>	The full native name of the attribute	
<b>name</b>	The given name for the attribute	By default, it is the same as the <i>nativeIdentifier</i>
<b>nativeTypeIdentifier</b>	The native data type of the attribute	
<b>type</b>	The DTS type of the attribute	
<b>array</b>	Whether or not the data type is an array or a list	
<b>extra</b>	A map of extra information only relevant for certain attributes of	

	resources from certain connectors	
--	-----------------------------------	--

### 7.3.24 ConstraintDefinition

ConstraintDefinitions model the details of table constraints defined in a data source.

**i** Only Primary Key and Foreign Key constraints are stored as ConstraintDefinitions.

**i** ConstraintDefinitions only exist for collection resources in relational databases.

**i** Non-SQL datasources may be assigned ConstraintDefinitions with slightly different interpretations. The table below shows the interpretation for Smallworld in the Note column.

```
{
  "constraintName" : "",
  "constraintType" : "",
  "tableName" : "",
  "columnNames" : [],
  "referencedTableName" : "",
  "referencedColumnNames" : []
}
```

Attribute	Value	Note
<b>constraintName</b>	The name of the constraint	For Smallworld, this is a generated PK name or the name of a join field
<b>constraintType</b>	The type of constraint ("DTS_PRIMARY_KEY" or "DTS_FOREIGN_KEY")	
<b>tableName</b>	The name of the collection/table the constraint is defined on	For Smallworld, the collection with the join field being modeled, or the intermediate table that governs the join
<b>columnNames</b>	The list of columns the constraint is defined on	For Smallworld, the list of underlying physical fields on the <i>tableName</i> side of the join
<b>referencedTableName</b>	The table that the constraint references	For Smallworld, the collection on the opposite side of the join

<b>referencedColumnNames</b>	The list of columns that are referenced by the constraint (the order corresponds to that of <i>columnNames</i> )	For Smallworld, the list of underlying physical fields on the <i>referencedTableName</i> side of the join
------------------------------	--	---

### 7.3.25 TypeResource

TypeResources model the details of a structured entity. Generally used to describe the structure of complex types or wrapper entities.

```
{
  "name" : "",
  "type" : "",
  "attributes" : []
}
```

Attribute	Value	Note
<b>name</b>	The name of the entity or type	
<b>type</b>	The type of the entity	For complex types, same as <i>name</i>
<b>attributes</b>	The list of attributes the type or entity has - in <a href="#">AttributeDescriptor</a> <sup>314</sup> format	

### 7.3.26 AggregateSource

AggregateSources model data resources used in Aggregate definitions.

 For more information on this functionality, please see [Aggregation](#)<sup>242</sup>.

```
{
  "name" : "",
  "nativeIdentifier" : "",
  "connectorId" : "",
  "connectorName" : "",
  "type" : "",
  "fields" : {},
  "predicate" : {}
}
```

```
}
```

Attribute	Value	Note
<b>name</b>	The DTS name of the resource.	
<b>nativeIdentifier</b>	The native identifier of the resource.	
<b>connectorId</b>	The <i>_id</i> of the connector the resource belongs to.	
<b>connectorName</b>	The name of the connector the resource belongs to.	
<b>type</b>	The type of resource. "COLLECTION" / "ROUTINE"	
<b>fields</b>	A map of the attributes from the resource that are included in the Aggregate. The keys represent the attributes' nativeIdentifiers, while the values represent the names by which they are known inside the aggregate.	
<b>predicate</b>	The <a href="#">DTSPredicate</a> <sup>239</sup> that defines the universal filter used on this resource, if any.	

### 7.3.27 AggregateRelationship

AggregateRelationships model relationships between Aggregate sources. Each relationship is always between two sources and is asymmetrical - one source is the parent and one is the child.

 For more information on this functionality, please see [Aggregation](#)<sup>242</sup>.

```
{
  "parentConnectorId" : "",
  "parentIdentifier" : "",
  "childConnectorId" : "",
  "childIdentifier" : "",
  "predicate" : {}
}
```

Attribute	Value	Note
<b>parentConnectorId</b>	The <i>_id</i> of the connector the parent source belongs to.	
<b>parentIdentifier</b>	The native identifier of the parent source.	
<b>childConnectorId</b>	The <i>_id</i> of the connector the child source belongs to.	
<b>childIdentifier</b>	The native identifier of the child source.	
<b>predicate</b>	The <a href="#">DTSPredicate</a> <sup>239</sup> that models the relationship between the parent and the child.	The predicate will always have <i>operatorName</i> = "eq", the <i>attributeName</i> represents the child's Query Parameter and the <i>attributeValue</i> represents the parent's Attribute.

### 7.3.28 WebserviceResource

WebserviceResources model configurations for DTS resources as it pertains to their inclusion in designed Webservices.

```
{
  "connector" : "",
  "remoteName" : "",
  "name" : "",
  "streamPathPrefix" : "",
  "methsMetadata" : {}
}
```

Attribute	Value	Note
<b>connector</b>	The <i>_id</i> of the connector the resource belongs to.	
<b>remoteName</b>	The native identifier of the resource.	
<b>name</b>	The custom name given to the resource inside the webservice, if	Only applies to remote call resources.

	any.	For REST webservices it is reflected in the operation's URL, while for SOAP webservices it becomes the operation's name.
<b>streamPathPrefix</b>	The URL prefix that will be used for streaming operations on the resource.	Only applies to streaming resources in REST webservices.
<b>methsMetadata</b>	An map of the settings for streaming operations on this resource.	Only applies to streaming resources.

## Streaming Operation Settings

The **methsMetadata** attribute contains a map whose keys represent the identifiers for the possible streaming operations on a resource. The presence of an item implies its inclusion in the Webservice.

- open\_stream\_w\_inline** Open a stream with URL query parameters (REST only)  
Has a *fields* attribute which lists all fields which will be available as query parameters in the resulting GET operation.
- open\_stream\_w\_predicate** Open a stream using a predicate (SOAP & REST).  
When used in a REST webservice, it results in a POST operation.
- get\_stream\_records** Get records from a stream (SOAP & REST).  
Mandatory if any Open Stream operation is enabled.
- get\_stream\_records\_w\_inline** Get a set of records with URL query parameters (REST only)  
Has a *fields* attribute which lists all fields which will be available as query parameters in the resulting GET operation.
- get\_stream\_records\_w\_predicate** Get a set of records using a predicate (SOAP & REST).  
When used in a REST webservice, it results in a POST operation.
- get\_stream\_record\_w\_key** Get a single record using a unique field (SOAP & REST).  
Has a *fields* attribute which will contain a single element representing the chosen field's name.

All entries can have a *name* attribute which sets a custom operation name (SOAP) or a custom URL suffix (REST).

 For more information on Webservice streaming operations, please see [Webservices - Access](#) <sup>195</sup>.





# Known Limitations

## 8 Known Limitations

<b>3D Geometries</b>	While simple 3D geometries are supported throughout DTS, approximations for complex curves (arcs, splines and any combination thereof) are only available in 2D.
<b>Aggregate Queries</b>	Aggregates can only be queried relative to the fields of the Main Source. Fields from other sources can only be delivered or used in static filters and relationships.
<b>Collections on Upgrade</b>	When upgrading from an older version of DTS, you may notice some collections included in various Projects have all their fields become excluded. This only happens when there was no manual interaction with the fields list of the collection and all fields are enabled by default. The simple workaround is to exclude and re-include the collection in the Project, which will bring back all the fields.
<b>Other Upgrade Considerations</b>	When upgrading from an older version of DTS, for the safest possible transition, all Projects should be republished and all Webservices should be redeployed.
<b>Individual Connector Limitations</b>	Each individual <a href="#">Connector</a> <sup>118</sup> may have its own set of limitations. They are listed in each Connector's Limitations section.



# Licenses













## 9 Licenses

This section concentrates all licensing information regarding the DTS product and its dependencies.

- [3rd Party Licenses](#) 

### 9.1 3rd Party Licenses

The DTS distribution packages can include any of the following 3rd Party software products and libraries, each under its respective license:

3rd Party Software	Version	Owner	Function	License
asm	5.2	OW2	Bytecode Library	<a href="#">BSD 3-clause</a> 
camel	2.25.0	Apache Software Foundation	Enterprise Integration	<a href="#">Apache 2.0</a> 
commons	1.15	Apache Software Foundation	General Purpose Library	<a href="#">Apache 2.0</a> 
cxfr	3.3.5	Apache Software Foundation	Services Framework	<a href="#">Apache 2.0</a> 
docker-java	3.2.7	Marcus Linke	Docker Interop Library	<a href="#">Apache 2.0</a> 
gson	2.8.6	Google	JSON Library	<a href="#">Apache 2.0</a> 
jackson	2.10.1	FasterXML	JSON Library	<a href="#">Apache 2.0</a> 
javax.mail	1.6.2	Oracle	Email Library	<a href="#">CDDL 1.1</a> 
jaxws-api	2.3.1	Oracle	Webservice Annotations	<a href="#">CDDL 1.1</a> 
jcifs	2.1.6	JCifs	Samba Library	<a href="#">LGPL 2.1</a> 
jedis	3.2.0	Redis Labs	Redis Client	<a href="#">MIT</a> 
jsch	0.1.55	JCraft	SSH Library	<a href="#">BSD 3-clause</a> 

kubernetes-client	4.13.0	Fabric8	Kubernetes Java Client	<a href="#">Apache 2.0</a> <sup>325</sup>
mongo-java-driver	3.12.1	Mongo DB	MongoDB Client	<a href="#">Apache 2.0</a> <sup>325</sup>
mssql-jdbc	8.2.2.jre8	Microsoft	JDBC Driver	<a href="#">MIT</a> <sup>348</sup>
mysql-connector-java	8.0.30	Oracle Corporation	JDBC Driver	<a href="#">GPL 2.0</a> <sup>336</sup>
ngdbc	2.9.12	SAP	SAP HANA Driver	<a href="#">SAP DLA</a> <sup>354</sup>
ojdbc	19.3.0.0	Oracle Corporation	JDBC Driver	<a href="#">OTNLA</a> <sup>349</sup>
postgis-jdbc	2.5.0	OSGeo	JDBC Driver Extension	<a href="#">LGPL 2.1</a> <sup>341</sup>
postgresql	42.2.17	PostgreSQL	JDBC Driver	<a href="#">BSD 2-clause</a> <sup>329</sup>
reactive-streams	1.0.3			<a href="#">CC0 1.0</a> <sup>330</sup>
spring	5.2.3.RELEASE	Pivotal Software	Web Framework	<a href="#">Apache 2.0</a> <sup>325</sup>

DTS uses each of the above software products in accordance with its license and, upon distribution, extends the obligations regarding their respective assets specified in each license to its users.

### 9.1.1 Apache 2.0

## Apache License

Version 2.0, January 2004

<http://www.apache.org/licenses/>

## TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

### 1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

## 2. Grant of Copyright License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

### 3. Grant of Patent License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

### 4. Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

1. You must give any other recipients of the Work or Derivative Works a copy of this License; and
2. You must cause any modified files to carry prominent notices stating that You changed the files; and
3. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
4. If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

## **5. Submission of Contributions.**

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

## **6. Trademarks.**

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

## **7. Disclaimer of Warranty.**

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

## **8. Limitation of Liability.**

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

## **9. Accepting Warranty or Additional Liability.**

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims

asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

## END OF TERMS AND CONDITIONS

### 9.1.2 BSD 2-clause

<https://opensource.org/licenses/BSD-2-Clause>

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

### 9.1.3 BSD 3-clause

<https://opensource.org/licenses/BSD-3-Clause>

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

#### 9.1.4 CCO 1.0

<https://creativecommons.org/publicdomain/zero/1.0/>

#### 9.1.5 CDDL 1.1

### COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL)

<https://spdx.org/licenses/CDDL-1.1>

#### Version 1.1

- 1. Definitions.
  - 1.1. "Contributor" means each individual or entity that creates or contributes to the creation of Modifications.
  - 1.2. "Contributor Version" means the combination of the Original Software, prior Modifications used by a Contributor (if any), and the Modifications made by that particular Contributor.
  - 1.3. "Covered Software" means (a) the Original Software, or (b) Modifications, or (c) the combination of files containing Original Software with files containing Modifications, in each case including portions thereof.
  - 1.4. "Executable" means the Covered Software in any form other than Source Code.

- 1.5. "Initial Developer" means the individual or entity that first makes Original Software available under this License.
- 1.6. "Larger Work" means a work which combines Covered Software or portions thereof with code not governed by the terms of this License.
- 1.7. "License" means this document.
- 1.8. "Licensable" means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently acquired, any and all of the rights conveyed herein.
- 1.9. "Modifications" means the Source Code and Executable form of any of the following:
  - A. Any file that results from an addition to, deletion from or modification of the contents of a file containing Original Software or previous Modifications;
  - B. Any new file that contains any part of the Original Software or previous Modification; or
  - C. Any new file that is contributed or otherwise made available under the terms of this License.
- 1.10. "Original Software" means the Source Code and Executable form of computer software code that is originally released under this License.
- 1.11. "Patent Claims" means any patent claim(s), now owned or hereafter acquired, including without limitation, method, process, and apparatus claims, in any patent Licensable by grantor.
- 1.12. "Source Code" means (a) the common form of computer software code in which modifications are made and (b) associated documentation included in or with such code.
- 1.13. "You" (or "Your") means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License. For legal entities, "You" includes any entity which controls, is controlled by, or is under common control with You. For purposes of this definition, "control" means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.
- 2. License Grants.
  - 2.1. The Initial Developer Grant.

Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, the Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license:

- (a) under intellectual property rights (other than patent or trademark) Licensable by Initial Developer, to use, reproduce, modify, display, perform, sublicense and distribute the Original Software (or portions thereof), with or without Modifications, and/or as part of a Larger Work; and

- (b) under Patent Claims infringed by the making, using or selling of Original Software, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Software (or portions thereof).
  - (c) The licenses granted in Sections 2.1(a) and (b) are effective on the date Initial Developer first distributes or otherwise makes the Original Software available to a third party under the terms of this License.
  - (d) Notwithstanding Section 2.1(b) above, no patent license is granted: (1) for code that You delete from the Original Software, or (2) for infringements caused by: (i) the modification of the Original Software, or (ii) the combination of the Original Software with other software or devices.
- 2.2. Contributor Grant.

Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:

- (a) under intellectual property rights (other than patent or trademark) Licensable by Contributor to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof), either on an unmodified basis, with other Modifications, as Covered Software and/or as part of a Larger Work; and
  - (b) under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: (1) Modifications made by that Contributor (or portions thereof); and (2) the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination).
  - (c) The licenses granted in Sections 2.2(a) and 2.2(b) are effective on the date Contributor first distributes or otherwise makes the Modifications available to a third party.
  - (d) Notwithstanding Section 2.2(b) above, no patent license is granted: (1) for any code that Contributor has deleted from the Contributor Version; (2) for infringements caused by: (i) third party modifications of Contributor Version, or (ii) the combination of Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or (3) under Patent Claims infringed by Covered Software in the absence of Modifications made by that Contributor.
- 3. Distribution Obligations.
    - 3.1. Availability of Source Code.

Any Covered Software that You distribute or otherwise make available in Executable form must also be made available in Source Code form and that Source Code form must be distributed only under the terms of this License. You must include a copy of this License with every copy of the Source Code form of the Covered Software You distribute or otherwise make available. You must inform recipients of any such Covered Software in Executable form as to how they can obtain such

Covered Software in Source Code form in a reasonable manner on or through a medium customarily used for software exchange.

- 3.2. Modifications.

The Modifications that You create or to which You contribute are governed by the terms of this License. You represent that You believe Your Modifications are Your original creation(s) and/or You have sufficient rights to grant the rights conveyed by this License.

- 3.3. Required Notices.

You must include a notice in each of Your Modifications that identifies You as the Contributor of the Modification. You may not remove or alter any copyright, patent or trademark notices contained within the Covered Software, or any notices of licensing or any descriptive text giving attribution to any Contributor or the Initial Developer.

- 3.4. Application of Additional Terms.

You may not offer or impose any terms on any Covered Software in Source Code form that alters or restricts the applicable version of this License or the recipients' rights hereunder. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However, you may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear that any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

- 3.5. Distribution of Executable Versions.

You may distribute the Executable form of the Covered Software under the terms of this License or under the terms of a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable form does not attempt to limit or alter the recipient's rights in the Source Code form from the rights set forth in this License. If You distribute the Covered Software in Executable form under a different license, You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.

- 3.6. Larger Works.

You may create a Larger Work by combining Covered Software with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Software.

- 4. Versions of the License.

- 4.1. New Versions.

Oracle is the initial license steward and may publish revised and/or new versions of this License from time to time. Each version will be given a distinguishing version number. Except as provided in Section 4.3, no one other than the license steward has the right to modify this License.

- 4.2. Effect of New Versions.

You may always continue to use, distribute or otherwise make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. If the Initial Developer includes a notice in the Original Software prohibiting it from being distributed or otherwise made available under any subsequent version of the License, You must distribute and make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. Otherwise, You may also choose to use, distribute or otherwise make the Covered Software available under the terms of any subsequent version of the License published by the license steward.

- 4.3. Modified Versions.

When You are an Initial Developer and You want to create a new license for Your Original Software, You may create and use a modified version of this License if You: (a) rename the license and remove any references to the name of the license steward (except to note that the license differs from this License); and (b) otherwise make it clear that the license contains terms which differ from this License.

- 5. DISCLAIMER OF WARRANTY.

COVERED SOFTWARE IS PROVIDED UNDER THIS LICENSE ON AN "AS IS" BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED SOFTWARE IS FREE OF DEFECTS, MERCHANTABILITY, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED SOFTWARE IS WITH YOU. SHOULD ANY COVERED SOFTWARE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED SOFTWARE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

- 6. TERMINATION.

- 6.1. This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.
- 6.2. If You assert a patent infringement claim (excluding declaratory judgment actions) against Initial Developer or a Contributor (the Initial Developer or Contributor against whom You assert such claim is referred to as "Participant") alleging that the Participant Software (meaning the Contributor Version where the Participant is a Contributor or the Original Software where the Participant is the Initial Developer) directly or indirectly infringes any patent, then any and all rights granted directly or indirectly to You by such Participant, the Initial Developer (if the Initial Developer is not the Participant) and all Contributors under Sections 2.1 and/or 2.2 of this License shall, upon 60 days notice from Participant terminate prospectively and automatically at the expiration of such 60 day notice period, unless if within such 60 day period You withdraw Your claim with respect to the Participant Software against such Participant either unilaterally or pursuant to a written agreement with Participant.

- 6.3. If You assert a patent infringement claim against Participant alleging that the Participant Software directly or indirectly infringes any patent where such claim is resolved (such as by license or settlement) prior to the initiation of patent infringement litigation, then the reasonable value of the licenses granted by such Participant under Sections 2.1 or 2.2 shall be taken into account in determining the amount or value of any payment or license.
- 6.4. In the event of termination under Sections 6.1 or 6.2 above, all end user licenses that have been validly granted by You or any distributor hereunder prior to termination (excluding licenses granted to You by any distributor) shall survive termination.
- 7. LIMITATION OF LIABILITY.

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL YOU, THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED SOFTWARE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO ANY PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTY'S NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

- 8. U.S. GOVERNMENT END USERS.

The Covered Software is a "commercial item," as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of "commercial computer software" (as that term is defined at 48 C.F.R. § 252.227-7014(a)(1)) and "commercial computer software documentation" as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Software with only those rights set forth herein. This U.S. Government Rights clause is in lieu of, and supersedes, any other FAR, DFAR, or other clause or provision that addresses Government rights in computer software under this License.

- 9. MISCELLANEOUS.

This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall be governed by the law of the jurisdiction specified in a notice contained within the Original Software (except to the extent applicable law, if any, provides otherwise), excluding such jurisdiction's conflict-of-law provisions. Any litigation relating to this License shall be subject to the jurisdiction of the courts located in the jurisdiction and venue specified in a notice contained within the Original Software, with the losing party responsible for costs, including, without limitation, court costs and reasonable attorneys' fees and expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be

construed against the drafter shall not apply to this License. You agree that You alone are responsible for compliance with the United States export administration regulations (and the export control laws and regulation of any other countries) when You use, distribute or otherwise make available any Covered Software.

- 10. RESPONSIBILITY FOR CLAIMS.

As between Initial Developer and the Contributors, each party is responsible for claims and damages arising, directly or indirectly, out of its utilization of rights under this License and You agree to work with Initial Developer and Contributors to distribute such responsibility on an equitable basis. Nothing herein is intended or shall be deemed to constitute any admission of liability.

#### NOTICE PURSUANT TO SECTION 9 OF THE COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL)

The code released under the CDDL shall be governed by the laws of the State of California (excluding conflict-of-law provisions). Any litigation relating to this License shall be subject to the jurisdiction of the Federal Courts of the Northern District of California and the state courts of the State of California, with venue lying in Santa Clara County, California.

### 9.1.6 GPL 2.0

## GNU GENERAL PUBLIC LICENSE

<https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html>

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.

51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

## Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program

whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

## TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an

appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-

readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

## NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

## END OF TERMS AND CONDITIONS

### 9.1.7 LGPL 2.1

## GNU LESSER GENERAL PUBLIC LICENSE

<https://www.gnu.org/licenses/old-licenses/lgpl-2.1.en.html>

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc.

51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

## Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it;

that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C

Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

## TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a) The modified work must itself be a software library.
- b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
- c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
- d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

- a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in

the Library will not necessarily be able to recompile the application to use the modified definitions.)

- b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.
- c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.
- d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.
- e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

- a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.
- b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the

free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

## NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

## END OF TERMS AND CONDITIONS

### 9.1.8 MIT

<https://opensource.org/licenses/MIT>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE

AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

### 9.1.9 OTNLA

## Oracle Technology Network License Agreement

<https://www.oracle.com/downloads/licenses/distribution-license.html>

Oracle is willing to authorize Your access to software associated with this License Agreement (“Agreement”) only upon the condition that You accept that this Agreement governs Your use of the software. By selecting the “Accept License Agreement” button or box (or the equivalent) or installing or using the Programs You indicate Your acceptance of this Agreement and Your agreement, as an authorized representative of Your company or organization (if being acquired for use by an entity) or as an individual, to comply with the license terms that apply to the software that You wish to download and access. If You are not willing to be bound by this Agreement, do not select the “Accept License Agreement” button or box (or the equivalent) and do not download or access the software.

### Definitions

"Oracle" refers to Oracle America, Inc. "You" and "Your" refers to (a) a company or organization (each an “Entity”) accessing the Programs, if use of the Programs will be on behalf of such Entity; or (b) an individual accessing the Programs, if use of the Programs will not be on behalf of an Entity. “Contractors” refers to Your agents and contractors (including, without limitation, outsourcers). "Program(s)" refers to Oracle software provided by Oracle pursuant to this Agreement and any updates, error corrections, and/or Program Documentation provided by Oracle. “Program Documentation” refers to Program user manuals and Program installation manuals, if any. If available, Program Documentation may be delivered with the Programs and/or may be accessed from [www.oracle.com/documentation](http://www.oracle.com/documentation). “Associated Product” refers to the Oracle product(s), if any, and as identified in the Programs documentation or on the Programs download site, with which the Programs are intended to enable or enhance interoperation with Your application(s). “Separate Terms” refers to separate license terms that are specified in the Program Documentation, readmes or notice files and that apply to Separately Licensed Third Party Technology. “Separately Licensed Third Party Technology” refers to third party technology that is licensed under Separate Terms and not under the terms of this Agreement.

### License Rights and Restrictions

Oracle grants You a nonexclusive, nontransferable, limited license to, subject to the restrictions stated in this Agreement, (a) internally use the Programs solely for the purposes of developing, testing, prototyping and demonstrating Your applications, and running the Programs for Your own internal business operations; and (b) redistribute unmodified Programs and Programs Documentation pursuant to the Programs Redistribution section below. You may allow Your Contractor(s) to use the Programs, provided they are acting on Your behalf to exercise license rights granted in this Agreement and further provided that You are responsible for their compliance with this Agreement in such use. You will have a written agreement with Your Contractor(s) that strictly limits their right to use the Programs and that otherwise protects Oracle's intellectual property rights to the same extent as this Agreement. You may make copies of the Programs to the extent reasonably necessary to exercise the license rights granted in this Agreement. You may make one copy of the Programs for backup purposes.

Further, You may not:

- remove or modify any Program markings or any notice of Oracle's or a licensor's proprietary rights;
- use the Programs to provide third party training unless Oracle expressly authorizes such use on the Program's download page;
- assign this Agreement or distribute, give, or transfer the Programs or an interest in them to any third party, except as expressly permitted in this Agreement (the foregoing shall not be construed to limit the rights You may otherwise have with respect to Separately Licensed Third Party Technology);
- cause or permit reverse engineering (unless required by law for interoperability), disassembly or decompilation of the Programs; and
- disclose results of any Program benchmark tests without Oracle's prior consent.

The Programs may contain source code that, unless expressly licensed in this Agreement for other purposes (for example, licensed under an open source license), is provided solely for reference purposes pursuant to the terms of this Agreement and may not be modified.

All rights not expressly granted in this Agreement are reserved by Oracle. If You want to use the Programs or Your application for any purpose other than as expressly permitted under this Agreement, You must obtain from Oracle or an Oracle reseller a valid Programs license under a separate agreement permitting such use. However, You acknowledge that the Programs may not be intended for production use and/or Oracle may not make a version of the Programs available for production or other purposes; any development or other work You undertake with the Programs is at Your sole risk.

## Programs Redistribution

We grant You a nonexclusive, nontransferable right to copy and distribute unmodified Programs and Programs Documentation as part of and included in Your application that is intended to interoperate with the Associated Product, if any, provided that You do not charge Your end users any additional fees for the use of the Programs. Prior to distributing the Programs and Programs

Documentation, You shall require Your end users to execute an agreement binding them to terms, with respect to the Programs and Programs Documentation, materially consistent and no less restrictive than those contained in this section and the sections of this Agreement entitled "License Rights and Restrictions" (except that the redistribution right granted to You shall not be included; Your end users may not distribute Programs and Programs Documentation to any third parties), "Ownership," "Export Controls," "Disclaimer of Warranties; Limitation of Liability," "No Technical Support" (with respect to Oracle support; You may provide Your own support for Programs at Your discretion), "Audit; Termination (except that Oracle's audit right shall not be included)," "Relationship Between the Parties," and "U.S. Government End Users." You must also include a provision stating that Your end users shall have no right to distribute the Programs and Programs Documentation, and a provision specifying us as a third party beneficiary of the agreement. You are responsible for obtaining these agreements with Your end users.

You agree to: (a) defend and indemnify us against all claims and damages caused by Your distribution of the Programs and Programs Documentation in breach of this Agreement and/or failure to include the required contractual provisions in Your end user agreement as stated above; (b) keep executed end user agreements and records of end user information including name, address, date of distribution and identity of Programs distributed; (c) allow us to inspect Your end user agreements and records upon request; and, (d) enforce the terms of Your end user agreements so as to effect a timely cure of any end user breach, and to notify us of any breach of the terms.

## Ownership

Oracle or its licensors retain all ownership and intellectual property rights to the Programs.

## Third-Party Technology

The Programs may contain or require the use of third party technology that is provided with the Programs. Oracle may provide certain notices to You in Program Documentation, readmes or notice files in connection with such third party technology. Third party technology will be licensed to You either under the terms of this Agreement or, if specified in the Program Documentation, readmes or notice files, under Separate Terms. Your rights to use Separately Licensed Third Party Technology under Separate Terms are not restricted in any way by this Agreement. However, for clarity, notwithstanding the existence of a notice, third party technology that is not Separately Licensed Third Party Technology shall be deemed part of the Programs and is licensed to You under the terms of this Agreement.

## Source Code for Open Source Software

For software that You receive from Oracle in binary form that is licensed under an open source license that gives You the right to receive the source code for that binary, You can obtain a copy of the applicable source code from <https://oss.oracle.com/sources/> or <http://www.oracle.com/goto/opensourcecode>. If the source code for such software was not

provided to You with the binary, You can also receive a copy of the source code on physical media by submitting a written request pursuant to the instructions in the "Written Offer for Source Code" section of the latter website.

## Export Controls

Export laws and regulations of the United States and any other relevant local export laws and regulations apply to the Programs . You agree that such export control laws govern Your use of the Programs (including technical data) and any services deliverables provided under this agreement, and You agree to comply with all such export laws and regulations (including "deemed export" and "deemed re-export" regulations). You agree that no data, information, program and/or materials resulting from Programs or services (or direct products thereof) will be exported, directly or indirectly, in violation of these laws, or will be used for any purpose prohibited by these laws including, without limitation, nuclear, chemical, or biological weapons proliferation, or development of missile technology. Accordingly, You confirm:

- You will not download, provide, make available or otherwise export or re-export the Programs, directly or indirectly, to countries prohibited by applicable laws and regulations nor to citizens, nationals or residents of those countries.
- You are not listed on the United States Department of Treasury lists of Specially Designated Nationals and Blocked Persons, Specially Designated Terrorists, and Specially Designated Narcotic Traffickers, nor are You listed on the United States Department of Commerce Table of Denial Orders.
- You will not download or otherwise export or re-export the Programs, directly or indirectly, to persons on the above mentioned lists.
- You will not use the Programs for, and will not allow the Programs to be used for, any purposes prohibited by applicable law, including, without limitation, for the development, design, manufacture or production of nuclear, chemical or biological weapons of mass destruction.
- 

## Information Collection

The Programs' installation and/or auto-update processes, if any, may transmit a limited amount of data to Oracle or its service provider about those processes to help Oracle understand and optimize them. Oracle does not associate the data with personally identifiable information. Refer to Oracle's Privacy Policy at [www.oracle.com/privacy](http://www.oracle.com/privacy).

## Disclaimer of Warranties; Limitation of Liability

THE PROGRAMS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. ORACLE FURTHER DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT .

IN NO EVENT WILL ORACLE BE LIABLE FOR ANY INDIRECT, INCIDENTAL, SPECIAL, PUNITIVE OR CONSEQUENTIAL DAMAGES, OR DAMAGES FOR LOSS OF PROFITS, REVENUE, DATA OR DATA USE, INCURRED BY YOU OR ANY THIRD PARTY, WHETHER IN AN ACTION IN CONTRACT OR TORT, EVEN IF ORACLE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. ORACLE'S ENTIRE LIABILITY FOR DAMAGES UNDER THIS AGREEMENT SHALL IN NO EVENT EXCEED ONE THOUSAND DOLLARS (U.S. \$1,000) .

## **No Technical Support**

Unless Oracle support for the Programs, if any, is expressly included in a separate, current support agreement between You and Oracle, Oracle's technical support organization will not provide technical support, phone support, or updates to You for the Programs provided under this Agreement.

## **Audit; Termination**

Oracle may audit Your use of the Programs. You may terminate this Agreement by destroying all copies of the Programs. This Agreement shall automatically terminate without notice if You fail to comply with any of the terms of this Agreement, in which case You shall promptly destroy all copies of the Programs.

## **U.S. Government End Users**

Programs and/or Programs Documentation delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs and/or Programs Documentation shall be subject to the license terms and license restrictions set forth in this Agreement. No other rights are granted to the U.S. Government.

## **Relationship Between the Parties**

Oracle is an independent contractor and we agree that no partnership, joint venture, or agency relationship exists between us. We each will be responsible for paying our own employees, including employment related taxes and insurance.. Nothing in this agreement shall be construed to limit either party's right to independently develop or distribute software that is functionally similar to

the other party's products, so long as proprietary information of the other party is not included in such software.

## Entire Agreement; Governing Law

You agree that this Agreement is the complete agreement for the Programs and this Agreement supersedes all prior or contemporaneous agreements or representations, including any clickwrap, shrinkwrap or similar licenses, or license agreements for prior versions of the Programs. This Agreement may not be modified and the rights and restrictions may not be altered or waived except in a writing signed by authorized representatives of You and of Oracle. If any term of this Agreement is found to be invalid or unenforceable, the remaining provisions will remain effective.

This Agreement is governed by the substantive and procedural laws of the State of California, USA, and You and Oracle agree to submit to the exclusive jurisdiction of, and venue in, the courts of San Francisco or Santa Clara counties in California in any dispute arising out of or relating to this Agreement.

## Notices

Should you have any questions concerning this License Agreement, or if you desire to contact Oracle for any reason, please write:

- Oracle America, Inc.
- 500 Oracle Parkway
- Redwood City, CA 94065

**Oracle Employees:** Under no circumstances are Oracle Employees authorized to download software for the purpose of distributing it to customers. Oracle products are available to employees for internal use or demonstration purposes only. In keeping with Oracle's trade compliance obligations under U.S. and applicable multilateral law, failure to comply with this policy could result in disciplinary action up to and including termination.

Last updated: 30 November 2016

### 9.1.10 SAP Developer License Agreement

## SAP DEVELOPER LICENSE AGREEMENT

[https://tools.hana.ondemand.com/developer-license-3\\_1.txt](https://tools.hana.ondemand.com/developer-license-3_1.txt)

Version 3.1

Please scroll down and read the following Developer License Agreement carefully ("Developer Agreement"). By clicking "I Accept" or by attempting to download, or install, or use the SAP software and other materials that accompany this Developer Agreement ("SAP Materials"), You agree that this Developer Agreement forms a legally binding agreement between You ("You" or "Your") and SAP SE, for and on behalf of itself and its subsidiaries and affiliates (as defined in Section 15 of the German Stock Corporation Act) and You agree to be bound by all of the terms and conditions stated in this Developer Agreement. If You are trying to access or download the SAP Materials on behalf of Your employer or as a consultant or agent of a third party (either "Your Company"), You represent and warrant that You have the authority to act on behalf of and bind Your Company to the terms of this Developer Agreement and everywhere in this Developer Agreement that refers to 'You' or 'Your' shall also include Your Company. If You do not agree to these terms, do not click "I Accept", and do not attempt to access or use the SAP Materials.

1. LICENSE: SAP grants You a non-exclusive, non-transferable, non-sublicensable, revocable, limited use license to copy, reproduce and distribute the application programming interfaces ("API"), documentation, plug-ins, templates, scripts and sample code ("Tools") on a desktop, laptop, tablet, smart phone, or other appropriate computer device that You own or control (any, a "Computer") to create new applications ("Customer Applications"). You agree that the Customer Applications will not: (a) unreasonably impair, degrade or reduce the performance or security of any SAP software applications, services or related technology ("Software"); (b) enable the bypassing or circumventing of SAP's license restrictions and/or provide users with access to the Software to which such users are not licensed; (c) render or provide, without prior written consent from SAP, any information concerning SAP software license terms, Software, or any other information related to SAP products; or (d) permit mass data extraction from an SAP product to a non-SAP product, including use, modification, saving or other processing of such data in the non-SAP product. In exchange for the right to develop Customer Applications under this Agreement, You covenant not to assert any Intellectual Property Rights in Customer Applications created by You against any SAP product, service, or future SAP development.

2. INTELLECTUAL PROPERTY: (a) SAP or its licensors retain all ownership and intellectual property rights in the APIs, Tools and Software. You may not: a) remove or modify any marks or proprietary notices of SAP, b) provide or make the APIs, Tools or Software available to any third party, c) assign this Developer Agreement or give or transfer the APIs, Tools or Software or an interest in them to another individual or entity, d) decompile, disassemble or reverse engineer (except to the extent permitted by applicable law) the APIs Tools or Software, (e) create derivative works of or based on the APIs, Tools or Software, (f) use any SAP name, trademark or logo, or (g) use the APIs or Tools to modify existing Software or other SAP product functionality or to access the Software or other SAP products' source code or metadata.  
(b) Subject to SAP's underlying rights in any part of the APIs, Tools or Software, You retain all ownership and intellectual property rights in Your Customer Applications.

3. FREE AND OPEN SOURCE COMPONENTS: The SAP Materials may include certain third party free or open source components ("FOSS Components"). You may have additional rights in such FOSS Components that are provided by the third party licensors of those components.

4. **THIRD PARTY DEPENDENCIES:** The SAP Materials may require certain third party software dependencies ("Dependencies") for the use or operation of such SAP Materials. These dependencies may be identified by SAP in Maven POM files, product documentation or by other means. SAP does not grant You any rights in or to such Dependencies under this Developer Agreement. You are solely responsible for the acquisition, installation and use of Dependencies. SAP DOES NOT MAKE ANY REPRESENTATIONS OR WARRANTIES IN RESPECT OF DEPENDENCIES, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY AND OF FITNESS FOR A PARTICULAR PURPOSE. IN PARTICULAR, SAP DOES NOT WARRANT THAT DEPENDENCIES WILL BE AVAILABLE, ERROR FREE, INTEROPERABLE WITH THE SAP MATERIALS, SUITABLE FOR ANY PARTICULAR PURPOSE OR NON-INFRINGEMENT. YOU ASSUME ALL RISKS ASSOCIATED WITH THE USE OF DEPENDENCIES, INCLUDING WITHOUT LIMITATION RISKS RELATING TO QUALITY, AVAILABILITY, PERFORMANCE, DATA LOSS, UTILITY IN A PRODUCTION ENVIRONMENT, AND NON-INFRINGEMENT. IN NO EVENT WILL SAP BE LIABLE DIRECTLY OR INDIRECTLY IN RESPECT OF ANY USE OF DEPENDENCIES BY YOU.

5. **WARRANTY:**

- a) If You are located outside the US or Canada: AS THE API AND TOOLS ARE PROVIDED TO YOU FREE OF CHARGE, SAP DOES NOT GUARANTEE OR WARRANT ANY FEATURES OR QUALITIES OF THE TOOLS OR API OR GIVE ANY UNDERTAKING WITH REGARD TO ANY OTHER QUALITY. NO SUCH WARRANTY OR UNDERTAKING SHALL BE IMPLIED BY YOU FROM ANY DESCRIPTION IN THE API OR TOOLS OR ANY AVAILABLE DOCUMENTATION OR ANY OTHER COMMUNICATION OR ADVERTISEMENT. IN PARTICULAR, SAP DOES NOT WARRANT THAT THE SOFTWARE WILL BE AVAILABLE UNINTERRUPTED, ERROR FREE, OR PERMANENTLY AVAILABLE. FOR THE TOOLS AND API ALL WARRANTY CLAIMS ARE SUBJECT TO THE LIMITATION OF LIABILITY STIPULATED IN SECTION 4 BELOW.
- b) If You are located in the US or Canada: THE API AND TOOLS ARE LICENSED TO YOU "AS IS", WITHOUT ANY WARRANTY, ESCROW, TRAINING, MAINTENANCE, OR SERVICE OBLIGATIONS WHATSOEVER ON THE PART OF SAP. SAP MAKES NO EXPRESS OR IMPLIED WARRANTIES OR CONDITIONS OF SALE OF ANY TYPE WHATSOEVER, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY AND OF FITNESS FOR A PARTICULAR PURPOSE. IN PARTICULAR, SAP DOES NOT WARRANT THAT THE SOFTWARE WILL BE AVAILABLE UNINTERRUPTED, ERROR FREE, OR PERMANENTLY AVAILABLE. YOU ASSUME ALL RISKS ASSOCIATED WITH THE USE OF THE API AND TOOLS, INCLUDING WITHOUT LIMITATION RISKS RELATING TO QUALITY, AVAILABILITY, PERFORMANCE, DATA LOSS, AND UTILITY IN A PRODUCTION ENVIRONMENT.

6. **LIMITATION OF LIABILITY:**

- a) If You are located outside the US or Canada: IRRESPECTIVE OF THE LEGAL REASONS, SAP SHALL ONLY BE LIABLE FOR DAMAGES UNDER THIS AGREEMENT IF SUCH DAMAGE (I) CAN BE CLAIMED UNDER THE GERMAN PRODUCT LIABILITY ACT OR (II) IS CAUSED BY INTENTIONAL MISCONDUCT OF SAP OR (III) CONSISTS OF PERSONAL INJURY. IN ALL OTHER CASES, NEITHER SAP NOR ITS EMPLOYEES, AGENTS AND SUBCONTRACTORS SHALL BE LIABLE FOR ANY KIND OF DAMAGE OR CLAIMS HEREUNDER.
- b) If You are located in the US or Canada: IN NO EVENT SHALL SAP BE LIABLE TO YOU, YOUR COMPANY OR TO ANY THIRD PARTY FOR ANY DAMAGES IN AN AMOUNT IN EXCESS OF

\$100 ARISING IN CONNECTION WITH YOUR USE OF OR INABILITY TO USE THE TOOLS OR API OR IN CONNECTION WITH SAP'S PROVISION OF OR FAILURE TO PROVIDE SERVICES PERTAINING TO THE TOOLS OR API, OR AS A RESULT OF ANY DEFECT IN THE API OR TOOLS. THIS DISCLAIMER OF LIABILITY SHALL APPLY REGARDLESS OF THE FORM OF ACTION THAT MAY BE BROUGHT AGAINST SAP, WHETHER IN CONTRACT OR TORT, INCLUDING WITHOUT LIMITATION ANY ACTION FOR NEGLIGENCE. YOUR SOLE REMEDY IN THE EVENT OF BREACH OF THIS DEVELOPER AGREEMENT BY SAP OR FOR ANY OTHER CLAIM RELATED TO THE API OR TOOLS SHALL BE TERMINATION OF THIS AGREEMENT. NOTWITHSTANDING ANYTHING TO THE CONTRARY HEREIN, UNDER NO CIRCUMSTANCES SHALL SAP AND ITS LICENSORS BE LIABLE TO YOU OR ANY OTHER PERSON OR ENTITY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR INDIRECT DAMAGES, LOSS OF GOOD WILL OR BUSINESS PROFITS, WORK STOPPAGE, DATA LOSS, COMPUTER FAILURE OR MALFUNCTION, ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSS, OR EXEMPLARY OR PUNITIVE DAMAGES.

7. INDEMNITY: You will fully indemnify, hold harmless and defend SAP against law suits based on any claim: (a) that any Customer Application created by You infringes or misappropriates any patent, copyright, trademark, trade secrets, or other proprietary rights of a third party, or (b) related to Your alleged violation of the terms of this Developer Agreement.

8. EXPORT: The Tools and API are subject to German, EU and US export control regulations. You confirm that: a) You will not use the Tools or API for, and will not allow the Tools or API to be used for, any purposes prohibited by German, EU and US law, including, without limitation, for the development, design, manufacture or production of nuclear, chemical or biological weapons of mass destruction; b) You are not located in Cuba, Iran, Sudan, Iraq, North Korea, Syria, nor any other country to which the United States has prohibited export or that has been designated by the U.S. Government as a "terrorist supporting" country (any, an "US Embargoed Country"); c) You are not a citizen, national or resident of, and are not under the control of, a US Embargoed Country; d) You will not download or otherwise export or re-export the API or Tools, directly or indirectly, to a US Embargoed Country nor to citizens, nationals or residents of a US Embargoed Country; e) You are not listed on the United States Department of Treasury lists of Specially Designated Nationals, Specially Designated Terrorists, and Specially Designated Narcotic Traffickers, nor listed on the United States Department of Commerce Table of Denial Orders or any other U.S. government list of prohibited or restricted parties and f) You will not download or otherwise export or re-export the API or Tools , directly or indirectly, to persons on the above-mentioned lists.

9. SUPPORT: Other than what is made available on the SAP Community Website (SCN) by SAP at its sole discretion and by SCN members, SAP does not offer support for the API or Tools which are the subject of this Developer Agreement.

10. TERM AND TERMINATION: You may terminate this Developer Agreement by destroying all copies of the API and Tools on Your Computer(s). SAP may terminate Your license to use the API and Tools immediately if You fail to comply with any of the terms of this Developer Agreement, or, for SAP's convenience by providing you with ten (10) day's written notice of termination (including email). In case of termination or expiration of this Developer Agreement, You must destroy all

copies of the API and Tools immediately. In the event Your Company or any of the intellectual property you create using the API, Tools or Software are acquired (by merger, purchase of stock, assets or intellectual property or exclusive license), or You become employed, by a direct competitor of SAP, then this Development Agreement and all licenses granted in this Developer Agreement shall immediately terminate upon the date of such acquisition.

**11. LAW/VENUE:**

- a) If You are located outside the US or Canada: This Developer Agreement is governed by and construed in accordance with the laws of the Germany. You and SAP agree to submit to the exclusive jurisdiction of, and venue in, the courts of Karlsruhe in Germany in any dispute arising out of or relating to this Developer Agreement.
- b) If You are located in the US or Canada: This Developer Agreement shall be governed by and construed under the Commonwealth of Pennsylvania law without reference to its conflicts of law principles. In the event of any conflicts between foreign law, rules, and regulations, and United States of America law, rules, and regulations, United States of America law, rules, and regulations shall prevail and govern. The United Nations Convention on Contracts for the International Sale of Goods shall not apply to this Developer Agreement. The Uniform Computer Information Transactions Act as enacted shall not apply.

**12. MISCELLANEOUS:** This Developer Agreement is the complete agreement for the API and Tools licensed (including reference to information/documentation contained in a URL). This Developer Agreement supersedes all prior or contemporaneous agreements or representations with regards to the subject matter of this Developer Agreement. If any term of this Developer Agreement is found to be invalid or unenforceable, the surviving provisions shall remain effective. SAP's failure to enforce any right or provisions stipulated in this Developer Agreement will not constitute a waiver of such provision, or any other provision of this Developer Agreement.