

DRS-Database Rest Services



DRS – DATABASE REST SERVICES

Contents

1.	Introduction.....	3
2.	Administrator Module	5
3.	Services Module	11
4.	Support Module.....	17
5.	How to submit a request.....	23
6	Installation.....	33

DRS – DATABASE REST SERVICES

1. Introduction

DRS-DATABASE REST SERVICES is an application designed to generate **JSON** outputs from defined services that are associated with SQL commands. Several functions are available for defining the infrastructure required for creating, executing, and monitoring running services.

Access to DMSII databases can be done either through Unisys-JDBC or using Unisys-MCPSQL. The presence of one of these applications in the mainframe environment is a prerequisite for using DRS. DRS can be used to access relational databases.

DRS must be installed on a Linux or Windows server configured with Apache Tomcat or any other similar software and uses Oracle database for parameter storage, definitions, and execution tracking.

SQL commands for accessing information from DMSII databases follow the ANSI SQL92 standard and can have joins between datasets and subqueries, for example. In the case of using MCPSQL as the interface with DMSII databases, joins using datasets from different databases can be created.

Execution statistics can be stored in a parameterizable time interval, enabling viewing of activity history by service.

A service defined in DRS can be used as soon as it is created and the output content is formed by the list of fields defined in the SQL command.

You can define a list of servers that can send requests to a given service, allowing you to limit access to existing services to specific sources.

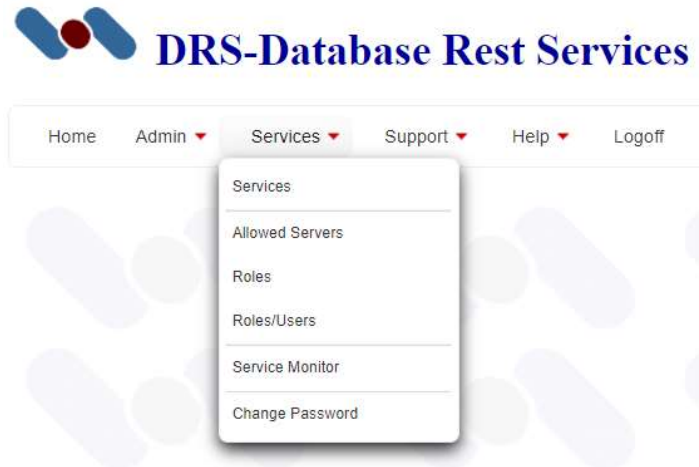
Three types of users can be defined to access DRS: administrator, service, and operator. Each of these users has a specific menu with the available functions. A user can have more than one type associated, with privileges of using functions present on more than one menu.

The administrator menu is presented below. This user type is responsible for configuring the infrastructure that will be used in defining services and running them.

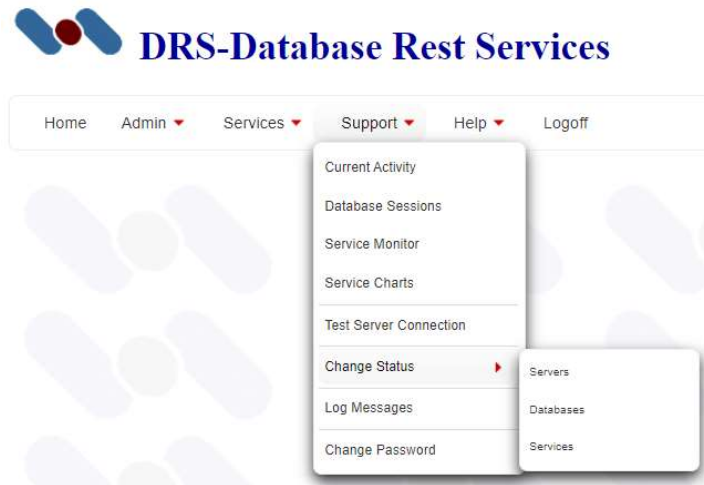


DRS – DATABASE REST SERVICES

The services menu contains the functions for maintaining services and defining security aspects involved with the defined services.



The support menu contains functions for monitoring and administering the DRS environment.



All these options will be described in the following topics.

DRS – DATABASE REST SERVICES

2. Administrator Module

This module is responsible for defining the global parameters, servers, databases, and users. These are the basic definitions to be used by the Service module to configure the services and security.

The functions available in this module are:

2.1 Parameters

This option is used to define the global parameters to be used during the DRS activity.

The previous image shows the attributes available to define or update an environment.

Admin Port Number: Define the port number used for administrative purposes. This is used when more than one instance of the application is running, on the same server or different servers. In this case, the drsDaemon must be running on the server defined in the **Admin Server Name** field.

All instances connect to the drsDaemon using the admin port and the daemon is responsible for synchronizing all instances.

Admin Server Name: Define the server name used for administrative purposes. The drsDaemon must be running on this server and the port defined in **Admin Port Name** will be used to synchronize all instances.

Max.Pool Size: Define the maximum number of database connections that can be opened and kept in the connection pool. If all connections in the pool are in use and the number of connections reaches the maximum allowed, any attempt to create a new connection will fail and the service returns an error message indicating this situation. As long as there is an idle connection in the pool, the service can process normally even if the number of connections in the pool reaches the maximum allowed.

DRS – DATABASE REST SERVICES

Max.Connection Timeout: Define the timeout for a connection in the pool to be closed. After no activity after the number of minutes defined in this field, the connection is automatically closed.

Max.Execution Timeout: Set the maximum time a connection can be in use to process one request. After this number of seconds, the connection is aborted and an error message is sent to the requestor.

Statistics Summary Interval: Set the interval at which the statistics of running services are summarized and stored in the database.

Stored Statistics: Define whether the statistics should be written in the statistics file.

Statistics Folder: Specify the folder name where the statistics file should be stored. This folder must be present on all servers if there is more than one server running instances of the DRS.

Debug Option: Set the debug flag to true or false. If true, additional messages are written in the log.

2.2 Servers

This function is used to query and update servers. There are four server types: MCP servers, SQL servers, external servers, and application servers.

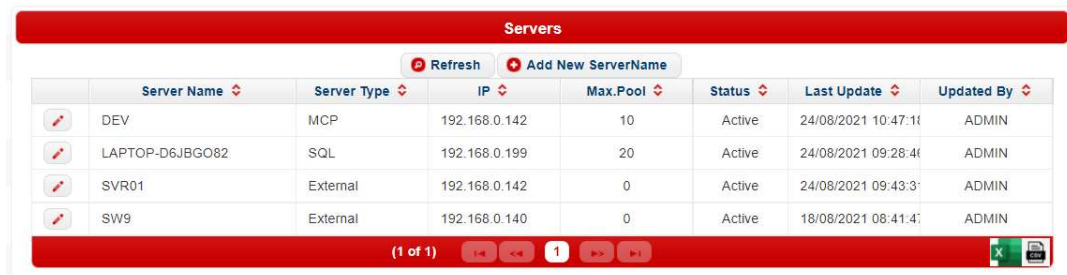
The first type, MCP, concerns all MCP servers that have DMSII databases that will be used by the services defined in the DRS.





The second type, SQL, concerns all SQL servers that have relational databases that will be used by the services defined in the DRS.



The third type, external, is related to servers that can request services to be performed. These servers can be used in the **Allowed Servers** function to create a list of servers that can access the services.

The last type, application, is related to servers that run instances of DRS. All servers must be defined.

The image below shows the **Servers** query page.

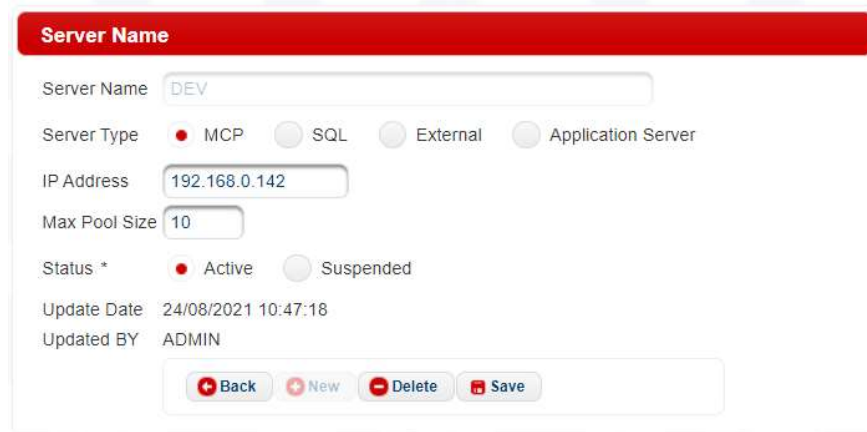


	Server Name	Server Type	IP	Max.Pool	Status	Last Update	Updated By
	DEV	MCP	192.168.0.142	10	Active	24/08/2021 10:47:18	ADMIN
	LAPTOP-D6JBG082	SQL	192.168.0.199	20	Active	24/08/2021 09:28:46	ADMIN
	SVR01	External	192.168.0.142	0	Active	24/08/2021 09:43:37	ADMIN
	SW9	External	192.168.0.140	0	Active	18/08/2021 08:41:47	ADMIN

From that page, it is possible to select one server to change, by pressing the  button. To create a new server, press the  button.

DRS – DATABASE REST SERVICES

The server detail page is displayed below.



The screenshot shows a web form titled "Server Name" with a red header. The form contains the following fields and controls:

- Server Name:** A text input field containing the value "DEV".
- Server Type:** A group of four radio buttons: "MCP" (selected), "SQL", "External", and "Application Server".
- IP Address:** A text input field containing the value "192.168.0.142".
- Max Pool Size:** A text input field containing the value "10".
- Status *:** A group of two radio buttons: "Active" (selected) and "Suspended".
- Update Date:** A text field showing the timestamp "24/08/2021 10:47:18".
- Updated BY:** A text field showing the username "ADMIN".
- Buttons:** A row of four buttons at the bottom: "Back" (with a plus icon), "New" (with a plus icon), "Delete" (with a minus icon), and "Save" (with a save icon).

The attributes available to a server are:

Server Name: The server name must be entered in this field, as the server is known on the network. If statistics is enabled for any service, the servers where the Drs is running must be defined.

Server Type: Three options are available. MCP must be used for all mainframes that have a DMSII database to be used by the services. External is a type to be used for servers that will request a service to be executed. On the **Allowed Servers** page, all external servers can be used to create a list of servers that can request services.

IP Address: The IP address of the server.

Max.Pool Size: Define the maximum number of database connections that can be opened and kept in the connection pool for this specific server.

Status: This field defines if the server is active or suspended. All transactions from a suspended server will be rejected.

2.3 Databases



This function is used to query and update DMSII databases. For each DMSII database, the mainframe where this database is present must be specified.

The Databases query page is shown below.



DRS – DATABASE REST SERVICES


Databases

ServerName: Database: Search Add New Database

	Database Name	Server Name	DB Type	Status	Last Update	Updated By
	ORACLE1	LAPTOP-D6JBGO82	SQL	Active	22/08/2021 19:48:45	ADMIN
	BTST	DEV	DMSII-MCPSQL	Active	24/08/2021 10:50:44	ADMIN

(1 of 1) < << 1 >> >



From that page, it is possible to select one database to change, by pressing the  button. To create a new database, press the button.

The database detail page is displayed below.

Database

Database Name *

DBPROD

Server Name *

DEV

Database Type *

☐ DMSII JDBC

☒ DMSII MCPSQL

☐ SQL

Status *

☒ Active

☐ Suspended

Update Date

Updated BY

The attributes available to a database are:

Database Name: The database name must be entered in this field.

If the database type is DMSII JDBC, the database name must be the same name used in the properties file. It can be different from the existing database name on the mainframe. The connect string present in the properties file will define the database to be accessed.

If the database type is DMSII MCPSQL, the database name must be the same defined in the MCPSQL/CONFIG file.

If the database type is SQL, the database name is related to a relational database must have the same name defined in the properties file present on SWDIR_SETUP directory (<database name>.properties).

Server Name: The server name where the database is present.

Database Type: This field defined how the database will be accessed. DMSII JDBC will use the Unisys JDBC software. DMSII MCPSQL will use the Unisys MCPSQL software. The software must be installed on the mainframes that have databases to be used by the services.

DRS – DATABASE REST SERVICES

Status: This field defines if the database is active or suspended. All transactions from a suspended database will be rejected.

2.4 Users

This function is used to query and update users. The user record is needed to connect to the application. The configuration process creates one user called **ADMIN** with password **admin**. This user is a Super user and must be used to configure all parameters and objects.

There are 4 user types: **Service**, **Admin**, **Support**, and **Super**. The first three user types have a specific menu option. It is possible to set more than one user type to a user.

The **Users** query page is shown below.

Users

Userid:


Search

Add New User

	User Id	Name	Service User	Oper User	Admin User	Super User	Last Access	Last IP	Status
	ADMIN	ADMINISTRATOR	Yes	Yes	Yes	Yes	26/07/2021 14:14:23	0.0.0.0:0.0.0.1	Active
	USER01	User number 01	Yes	No	No	No	26/07/2021 14:12:53	0.0.0.0:0.0.0.1	Active

(1 of 1)

1

From that page, it is possible to select one user to change, by pressing the  button. To create a new user, press the button.

The user detail page is displayed below.

User

User Id *

Name

Email

Password

User Type

☒ Service ☐ Support ☐ Admin ☐ Super

Status *

☒ Active ☐ Suspended

Last Access Date

Last Access Ip

Updated BY

The attributes available to a user are:

User Id: This is the user identification.

DRS – DATABASE REST SERVICES

Name: The user name should be entered in this field.

Email: The user's email.

Password: The password must be entered in this field.

User Type: The field defines the privileges the user will have. The Super user type includes all types. Each user type has its menu option.

Status: This field defines if the user is active or suspended. A suspended user cannot connect to the application.

DRS – DATABASE REST SERVICES

3. Services Module

This module is responsible for defining the services that will access DMSII databases and other related options.

The functions available in this module are:

3.1 Services

The creation of the services is done using the first option from the Services menu. The services query is shown below.

Services													
Service Filter:				Query		New Service							
	Service Name	Description	ServerName	Database Name	Service Type	Avg.Alert	Max. Alert	Rows Per Req.	Role Name	Sec. Enabled	Update Date	Status	Action
	service11	DMSII access	DEV	BTST	SQL	0.0000	0.0000	5		N	24/06/2021 11:46:58	Active	select * from clien whe
	service30	Test Oracle	LAPTOP-D6JBG082	ORACLE1	SQL	0.0000	0.0000	0		N	22/08/2021 19:58:05	Active	select * from clien
	service31	Test Oracle2	LAPTOP-D6JBG082	ORACLE1	SQL	0.0000	0.0000	0		N	22/08/2021 21:38:11	Active	select * from clien whe

From that page, it is possible to select one service to change, by pressing the button. To create a new service, press the **New Service** button.

The service page is displayed below.

Service

Service Name *

service20

Server Name *

MCP01.NET

Database Name

DBCNS

Description

DBCNS Rest Service

Service Type

☒ SQL ☐ URL

Avg. Response Time Alert

0.0000

Max. Response Time Alert

0.0000

Rows Per Request

0

Action *

SELECT O.ORDER_ID, O.ORDER_STATUS, O.ORDER_DATE, C.CLIENT_ID, C.CLIENT_NAME FROM ORDRS O, CLIEIN C where o.client_id=c.client_id and c.client_id=:clientid

Role Name

Select One

Return Data Only *

☐ Yes ☒ No

Parameter Type *

☒ Header ☐ Body

Security Enabled *

☐ Yes ☒ No

Stored Statistics *

☒ Yes ☐ No

Status *

☒ Active ☐ Suspended

Update Date

24/06/2021 12:13:19

Back

New

Delete

Save

Parameters

To define a service, the following attributes are available:

DRS – DATABASE REST SERVICES

Service Name: This is service identification. This name must be used as a parameter to the request. The URL to request to process the service will be discussed later in this document.

Server Name: The server name that contains the database to be used in the SQL statement. A combo box is available with a list of servers to be chosen.

Database Name: In this field the database name to be accessed must be informed. All databases from the selected server name will be present in a combo box.

Description: A text to describe the service.

Service Type: This option defines the type of service that is referred to by the destination field. Currently, only SQL is active. The URL type is related to a Cobol migrated to Java Online transaction that can become a Restful service.

Avg. Response Time Alert: The maximum average response time, in milliseconds, must be informed in this field. Every time the average time for the service exceeds the maximum specified, a message is written in the log and a record is created in the message table.

Max. Response Time Alert: The maximum response time, in milliseconds, must be informed in this field. Every time the response time for the service exceeds the maximum specified, a message is written in the log and a record is created in the message table.

Rows Per Request: This field defines the maximum number of records present in the service output.

Action: Define the service action to be executed. If the service type is SQL, this field must contain a valid SQL statement for the database defined in the Database Name field. If the service type is URL, the action field must contain a valid URL from a migrated application. The fields present on the Screen interface that will be part of this transaction must be defined as service parameters.

Role Name: If the service should be assigned to a role, one of the available roles must be selected from the combo box.

Returning Data Only: Specify if the output must contain only data or additional attributes like the number of records returned, response time, and transaction time.

Parameter Type: Define if the parameters set for the service must be placed in the header or the body of the request.


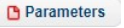
Security Enabled: If Yes is selected then the list of servers will be used to validate the origin of the request. No request will be accepted from a server that is not part of the allowed servers.

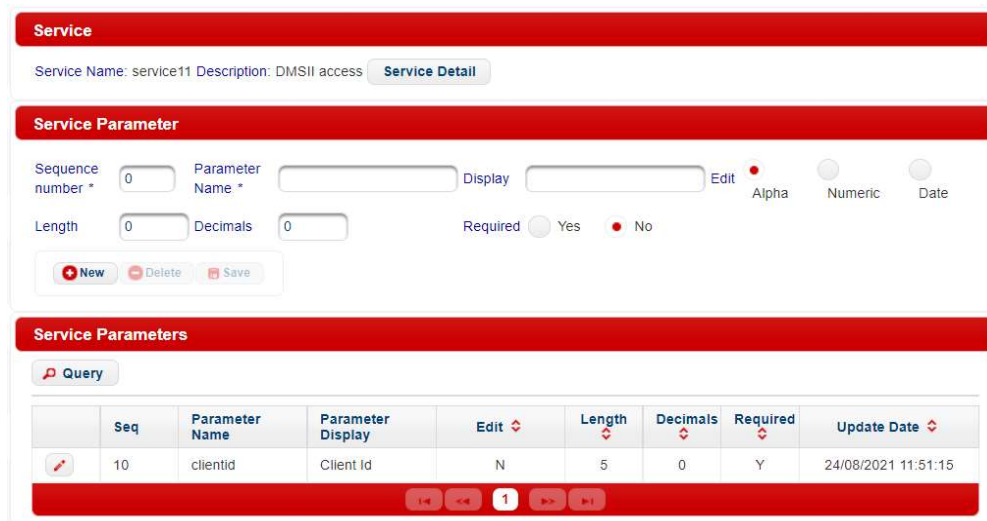
DRS – DATABASE REST SERVICES

Stored Statistics: Define if the statistics must be stored. This is related to the individual or summarized statistics. The summarized statistics are store in a table. The individual statistics are written in a file stored in the folder defined on the parameter page. The information is stored only if the servers where the DRS application is running are defined using the **Servers** page. The name must be the one obtained using the command **hostname**.

Status: This field defines if the service is active or suspended. A suspended service cannot be processed and the corresponding message will be sent back.

3.2 Service Parameters

It is possible to define the service parameters using the  button from the Services Query page. The parameters page can also be selected from the Service page, by pressing the  button.



Service

Service Name: service11 Description: DMSII access [Service Detail](#)

Service Parameter


Sequence number * Parameter Name * Display Edit ☒ Alpha ☐ Numeric ☐ Date

Length Decimals Required ☐ Yes ☒ No

[New](#) [Delete](#) [Save](#)

Service Parameters

[Query](#)

	Seq	Parameter Name	Parameter Display	Edit	Length	Decimals	Required	Update Date
	10	clientid	Client Id	N	5	0	Y	24/08/2021 11:51:15

[1](#) [2](#) [3](#) [4](#) [5](#)

The parameter page is divided into three parts. The first part contains the service name and description. There is a button to return to the service details. The second part is where the parameters are defined. The third part contains a grid with all the existing parameters. The button in the first column can be used to edit the parameter.

The existing attributes for a parameter are:

Sequence Number: A sequence number must be defined.

Parameter Name: The parameter name must be the same used in the SQL statement or the Screen filed name.

DRS – DATABASE REST SERVICES

Display: Define the display that will be used in the error message when an input error is found.

Edit: The edit of the field.

Length: The number of bytes of the parameter. If the field is numeric, this length is related to the integer part.

Decimals: The number of decimal places.

Required: Define whether the field is required or not. All required fields are checked when a request is received and an error message is sent when some of the fields are missing.

3.3 Allowed Servers

This page is used to define a list of servers that are allowed to send a request to a specific server. The validation will occur for every message received only if the field **Security Enabled** field in the service detail page is set to **Yes**.

The page can be shown below. The desired service should be select from the list and the available servers are listed on the left list box and the currently allowed servers are listed on the right list box.

After selecting the server, use the buttons in the middle of the page to move from one list box to another.

The screenshot shows a web interface titled "Allowed Servers" with a red header. Below the header, there is a "Services" dropdown menu set to "service30" and a "Query" button. The interface is divided into two main sections: "Servers Available" on the left and "Servers Selected" on the right, both with red headers. The "Servers Available" list contains two items: "SVR01" and "SW9". Between these two lists are two buttons: a right-pointing arrow and a left-pointing arrow, both circled in red. The "Servers Selected" list is currently empty.

3.4 Service Monitor

This page is used to see the activity of all services enabled. These services are listed in a grid and the number of transactions processed since the start of the application is displayed. The average response time for each service since the start is listed in a colored cell. If the background color is green, the time is less than 90% of the average response time alert defined in the service detail. If the background

DRS – DATABASE REST SERVICES

color is yellow, the average response time is between 90% and 100% of the average response time alert. Any value greater than the average response time alert has the background color in red.

Services Monitor								
Service Filter:			Refresh: 10		Query			
Service Name	ServerName	Database Name	Service Type	No.Transactions	Avg.Response	First Trans.	Last Trans.	Status
service11	DEV	BTST	SQL	0	0.00000			Active
service30	LAPTOP-D6JBG082	ORACLE1	SQL	53.148	0.00317	24/08/2021 12:11:25	24/08/2021 12:15:54	Active
service31	LAPTOP-D6JBG082	ORACLE1	SQL	0	0.00000			Active

Service Name: This field contains the service name.

ServerName: This field contains the server name of the database used to process the request.

Database Name: This field displays the database used to process the request.

Service Type: This field informs the service type: SQL if the destination is a database, URL if the destination is an application using SWScreen interface.

No.Transactions: This field lists the number of transactions processed since the application started.

Avg.Response: This cell shows the average response time since the application started.

First Trans.: This cell displays the date/time of the first transaction received by the service since the application started.

Last.Trans.: This cell displays the date/time of the last transaction processed by the service

3.5 Service Chart

Through the Service Chart page, it is possible to visualize the activity of a service in graphic mode.

The chart has the left axis for the number of transactions and the right axis for response time. This function can visualize the current activity or historical activity. If the **End Date** field is left blank, this indicates that the current activity will be displayed. When informing a **Start Date** and an **End Date**, the result page will display the historical information.

The statistics records are written in intervals defined on the Parameter page. The field **Statistics Summary Interval** defines the number of seconds when the statistic information is written.

DRS – DATABASE REST SERVICES



Service Name: One of the services in the list box must be selected.

Start Date/End Date: Define the interval that should be used to create the result page. If the End Date is blank, the current activity will be displayed.

Refresh: This field defines the interval in seconds at which the page is refreshed.

DRS – DATABASE REST SERVICES

4. Support Module

This module allows the user to check information about the status of several topics related to the application and take some actions.

4.1 Session In Use

This page shows all users connect to the application. The session id, ip address, and time when the session was started are present in the result grid.

Sessions In Use			
Refresh			
User Id	Session Id	IP Address	Session Started
ADMIN	0BD421BF0235DB7533CADE834AA95F88	192.168.0.199	24/08/2021 12:11:12
USER	928669DA46288B3DC90BB549C9D4A299	192.168.0.199	24/08/2021 12:23:45
1			

4.2 Database Connections

This page shows all connections present in the connection pool.

The server name and database name are displayed. The “Time Status” column contains information about when the last status change occurred.

If the Origin, User Id, and SQL Text columns are blank, the “Time Status” column indicates the idle time for that connection.

When a service process a request, the Origin, SQL statement, and User Id show the information about the SQL statement in progress and the requestor id. In this case, the “Time Status” column indicates when the query was started.

The first column has a button to kill the session in any state, idle or busy. If a busy connection id is killed, the query in progress is terminated and an error message will be sent to the requestor.

DRS – DATABASE REST SERVICES

Database Connections								
						Refresh: 10	Refresh	
<input type="checkbox"/>	Id	Server Name	Database Name	Pid	Time Status	Origin	User Id	Sql Text
<input checked="" type="checkbox"/> Kill Session	7	LAPTOP-D6JBG082	ORACLE1	0	24/08/2021 12:17:19			
<input checked="" type="checkbox"/> Kill Session	3	LAPTOP-D6JBG082	ORACLE1	0	24/08/2021 12:17:19			
<input checked="" type="checkbox"/> Kill Session	9	LAPTOP-D6JBG082	ORACLE1	0	24/08/2021 12:17:19			
<input checked="" type="checkbox"/> Kill Session	4	LAPTOP-D6JBG082	ORACLE1	0	24/08/2021 12:17:19			
<input checked="" type="checkbox"/> Kill Session	8	LAPTOP-D6JBG082	ORACLE1	0	24/08/2021 12:17:19			
<input checked="" type="checkbox"/> Kill Session	6	LAPTOP-D6JBG082	ORACLE1	0	24/08/2021 12:17:19			
<input checked="" type="checkbox"/> Kill Session	1	LAPTOP-D6JBG082	ORACLE1	0	24/08/2021 12:17:19			
<input checked="" type="checkbox"/> Kill Session	10	LAPTOP-D6JBG082	ORACLE1	0	24/08/2021 12:17:19			
<input checked="" type="checkbox"/> Kill Session	2	LAPTOP-D6JBG082	ORACLE1	0	24/08/2021 12:17:19			
<input checked="" type="checkbox"/> Kill Session	5	LAPTOP-D6JBG082	ORACLE1	0	24/08/2021 12:17:19			

4.3 Test Server Connection

This page is used to check if the connection between the application server and all mainframe servers are active.

To execute the process just select the checkbox related to the servers you want to test and press the **Test Connection** button. The corresponding message will be displayed indicating if any server cannot be reached.

Test Server Connection		
Refresh Test Connection		
<input type="checkbox"/>	Server Name	Status
<input checked="" type="checkbox"/>	DEV	Active
<input type="checkbox"/>	LAPTOP-D6JBG082	Active

4.4 Server Status

This function allows support users to check the current status of the servers and suspend the activity in a specific server, if necessary.

After selecting the **Servers** option in the **Change Status** menu option, a page is displayed with all existing servers in a grid.

The query can be restricted just to one status (Active or Suspended) or list all servers using the appropriate checkbox available above the grid.

DRS – DATABASE REST SERVICES

Server Status			
<input checked="" type="checkbox"/>	Active	<input checked="" type="checkbox"/>	Suspended
<input type="button" value="Refresh"/> <input type="button" value="Activate"/> <input type="button" value="Suspend"/>			
<input type="checkbox"/>	Host Name	Server Name	Status
<input checked="" type="checkbox"/>	DEV	DEV	Active
<input type="checkbox"/>	LAPTOP-D6JBGO82	LAPTOP-D6JBGO82	Active
<input type="checkbox"/>	SVR01	SVR01	Active
<input checked="" type="checkbox"/>	SW9	SW9	Active
<input type="button" value="1"/>			

Select the server to be changed and press the buttons or to enable or disable any activity in that server. An appropriate message will be sent.

Status changed to SUSPENDED for selected servers:2 servers updated			
Server Status			
<input checked="" type="checkbox"/>	Active	<input checked="" type="checkbox"/>	Suspended
<input type="button" value="Refresh"/> <input type="button" value="Activate"/> <input type="button" value="Suspend"/>			
<input type="checkbox"/>	Host Name	Server Name	Status
<input checked="" type="checkbox"/>	DEV	DEV	Suspended
<input type="checkbox"/>	LAPTOP-D6JBGO82	LAPTOP-D6JBGO82	Active
<input type="checkbox"/>	SVR01	SVR01	Active
<input checked="" type="checkbox"/>	SW9	SW9	Suspended
<input type="button" value="1"/>			

4.5 Database Status

This function allows support users to check the current status of the databases and suspend the activity in a specific database, if necessary.

After selecting the **Databases** option in the **Change Status** menu option, a page is displayed with all existing databases in a grid.

The query can be restricted just to one status (Active or Suspended) or list all databases using the appropriate checkbox available above the grid.

DRS – DATABASE REST SERVICES

Databases Status

Database: ☒ Active ☒ Suspended

<input type="checkbox"/>	Database Name	Host Name	Status
<input type="checkbox"/>	ORACLE1	LAPTOP-D6JBG082	Active
<input checked="" type="checkbox"/>	BTST	DEV	Active

Select the databases to be changed and press the buttons or to enable or disable any activity in that database. An appropriate message will be sent.

Databases Status

Database: ☒ Active ☒ Suspended

Status changed to SUSPENDED for selected databases:1 databases updated

<input type="checkbox"/>	Database Name	Host Name	Status
<input type="checkbox"/>	ORACLE1	LAPTOP-D6JBG082	Active
<input checked="" type="checkbox"/>	BTST	DEV	Suspended

4.6 Service Status

This function allows support users to check the current status of the services and suspend the activity in a specific service, if necessary.

After selecting the **Services** option in the **Change Status** menu option, a page is displayed with all existing services in a grid.



The query can be restricted just to one status (Active or Suspended) or list all services using the appropriate checkbox available above the grid.

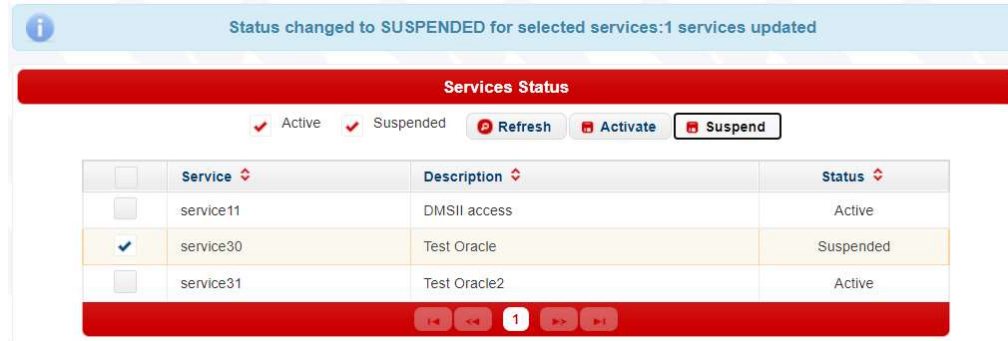
Services Status

☒ Active ☒ Suspended

<input type="checkbox"/>	Service	Description	Status
<input type="checkbox"/>	service11	DMSII access	Active
<input checked="" type="checkbox"/>	service30	Test Oracle	Active
<input type="checkbox"/>	service31	Test Oracle2	Active

DRS – DATABASE REST SERVICES

Select the services to be changed and press the buttons  or  to enable or disable any activity in that database. An appropriate message will be sent.



4.7 Log Messages

This page shows the messages generated while using the application. The message date/time, message type, module that generated the message, and the text are displayed when an interval date is specified.

The query uses the fields in the select area below to create the grid.

Start Date:  End Date:  Module Name:  Msg Type:  

The **Start Date** and **End Date** columns define the period of the log records to be displayed. If the columns **Module Name** and **Msg Type** have the default value **All**, all records from that period will be listed. If a specific module or type is defined, the query will be filtered.

There are three message types:

SUCCESS: This message type indicates that the text is related to a successful operation. Not all successful operations are logged but all meaningful are recorded.

WARNING: This message type is related to texts generated by the application after situations identified as temporary.

ERROR: this message type is related to operations that were not completed by the application due to security, database, or software error.

There are six modules:

ADMIN: this module is related to admin operations, such as definitions of hostnames, databases, and users.

DRS – DATABASE REST SERVICES

ALARM: this module is associated with response times that exceed the parameter defined to a service.

DAEMON: this module is related to functions performed by the daemon process.

SECURITY: this module is associated with functions that involve connections

SQL: this module is associated with functions that involve executing DMSII SQL statements.

SUPPORT: this module is associated with support actions, such as deleting a database session or aborting a background query.

Start Date: 25/08/2021 End Date: 25/08/2021 Module Name: All Msg Type: All Refresh				
Log Messages				
Id	Date	Msg Type	Module	Text
195	25/08/2021 11:08:13	SUCCESS	ADMIN	Database BTST stored by user ADMIN
194	25/08/2021 11:07:43	INFORMATION	SECURITY	User ADMIN connected(ip=192.168.0.199, SessionId=C3EACC8F24CB0FAA68F359DFCECB14B1)
193	25/08/2021 11:01:11	SUCCESS	OPERATOR	Status changed to SUSPENDED for selected services:1 updated by ADMIN
192	25/08/2021 10:57:38	SUCCESS	OPERATOR	Status changed to SUSPENDED for selected databases(BTST):1 updated by ADMIN
191	25/08/2021 10:57:29	SUCCESS	OPERATOR	Status changed to ACTIVE for selected servers(DEV SW9)2 updated by ADMIN
190	25/08/2021 10:56:16	SUCCESS	OPERATOR	Status changed to SUSPENDED for selected servers(DEV SW9)2 updated by ADMIN
189	25/08/2021 10:56:15	SUCCESS	OPERATOR	Status changed to ACTIVE for selected servers(DEV SW9)2 updated by ADMIN
188	25/08/2021 10:54:43	SUCCESS	OPERATOR	Status changed to SUSPENDED for selected servers(DEV SW9)2 updated by ADMIN
187	25/08/2021 10:51:24	INFORMATION	SECURITY	User ADMIN connected(ip=192.168.0.199, SessionId=E28B4E811383A7E87D30CDD00AA9BA30)
186	25/08/2021 09:39:30	SUCCESS	OPERATOR	Status changed to ACTIVE for selected databases(ORACLE1):1 updated by ADMIN
185	25/08/2021 09:39:20	INFORMATION	SECURITY	User ADMIN connected(ip=192.168.0.199, SessionId=FB7D842605989C66209FA02CCD82B4FC)
184	25/08/2021 09:33:35	SUCCESS	OPERATOR	Status changed to SUSPENDED for selected databases(ORACLE1):1 updated by ADMIN
183	25/08/2021 09:23:23	SUCCESS	OPERATOR	Status changed to ACTIVE for selected servers(LAPTOP-D6JBGO82)1 updated by ADMIN
182	25/08/2021 09:23:17	SUCCESS	OPERATOR	Status changed to SUSPENDED for selected servers(LAPTOP-D6JBGO82)1 updated by ADMIN

DRS – DATABASE REST SERVICES

5. How to submit a request

After creating or updating a particular service, you can submit a request to it. DRS has some endpoints that can be used to access the services available.

Depending on the type of SQL statement defined in a service, a specific endpoint must be used. The available endpoints are listed below:

5.1 /heartbeat

This endpoint can be used to check the availability of the DRS application. The return for this endpoint is a Json like that:

```
{
  "heartbeat": "DRS - service up and running",
  "time": "25-08-2021 09:39:07"
}
```

5.2 /services

This endpoint is to be used when a service contains a query to a database. One parameter, **service**, is needed to identify which services defined in DRS should be used. The request should be like that:

```
/services?service=<service name>
```

where <service name> is one of the services defined in DRS.

If the service has parameters, this endpoint uses the body to inform the parameters to be used by the service. The parameters are in the Json format and the parameter name must match the name defined in the **Service Parameter** page.

In the example below, the service **service31** has one parameter, **clientid**. The option defined for the Parameter type is Body, which means that the parameter should be informed in the message body.

The parameter page indicates that this is a numeric field and the length is 4 digits.

DRS – DATABASE REST SERVICES

Service

Service Name *

service31

Server Name *

LAPTOP-D6JBGO82

Database Name

ORACLE1

Description

Test Oracle2

Service Type

☒ SQL ☐ URL

Avg Response Time Alert

0.0000

Max Response Time Alert

0.0000

Rows Per Request

0

Action *

select * from clien where client_id=:clientid

Role Name

Select One

Return Data Only *

☐ Yes ☒ No

Parameter Type *

☐ Header ☒ Body

Security Enabled *

☐ Yes ☒ No

Stored Statistics *

☒ Yes ☐ No

Status *

☒ Active ☐ Suspended

Update Date

22/08/2021 21:38:11

Back

New

Delete

Save

Parameters

Service Parameters

Query

	Seq	Parameter Name	Parameter Display	Edit	Length	Decimals	Required	Update Date
	10	clientid	Client Id	N	4	0	Y	22/08/2021 21:38:43

1

When using the Postman to test the service, the configuration should be like:

GET http://192.168.0.199:8095/Drs/services?service=service31 Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   ... "clientid" : 2
3 }
```

The result is returned in the message body displayed below. The status field contains “ok”, indicating the request has been processed successfully.

DRS – DATABASE REST SERVICES

```
Body Cookies Headers (5) Test Results
Pretty Raw Preview Visualize JSON
1
2  "records": 1,
3  "responseTime": 0.0968103,
4  "time": "25-08-2021 12:16:46.208",
5  "items": [
6    {
7      "CREDIT_LIMIT": 35000,
8      "CITY": "LOS ANGELES",
9      "COUNTRY": "USA",
10     "CLIENT_ID": 2,
11     "CLIENT_NAME": "UNIVERSAL",
12     "ADDRESS": "UNIVERSAL BLVD, 872",
13     "STATE": "LA"
14   }
15 ],
16 "hasMoreRecords": false,
17 "status": "ok"
18
```

If you inform a value with a length greater than the defined in the **Service Parameter** page, an error message is sent. The status field has the value “ error”.

```
GET http://192.168.0.199:8095/Drs/services?service=service31 Send
Params Authorization Headers (8) Body Pre-request Script Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL JSON
1
2  ...."clientId" : 21111
3
Body Cookies Headers (4) Test Results
Pretty Raw Preview Visualize JSON
1
2  "time": "25-08-2021 12:19:25.745",
3  "message": "Parameter clientId maximum value exceeded; ",
4  "status": "error"
5
```

If a wrong service name is informed in the URL, in this case, service312, an error message is sent to the requestor.

DRS – DATABASE REST SERVICES

GET ▼ http://192.168.0.199:8095/Drs/services?service=service312 Send ▼

Params ● Authorization Headers (8) Body ● Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL JSON ▼

```
1 {
2   "clientId": 21111
3 }
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON ▼ ≡

```
1 {
2   "time": "25-08-2021 12:24:20.577",
3   "message": "Invalid service name",
4   "status": "error"
5 }
```

If a parameter was defined as Required and it is not informed, an error message is sent to the requestor. In the example below, the parameter was defined with the name `clientId1`. `clientId` is missing in the message body.

GET ▼ http://192.168.0.199:8095/Drs/services?service=service31 Send ▼

Params ● Authorization Headers (8) Body ● Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL JSON ▼

```
1 {
2   "clientId1": 21111
3 }
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON ▼ ≡

```
1 {
2   "time": "25-08-2021 12:28:49.237",
3   "message": "Parameter clientId is required; ",
4   "status": "error"
5 }
```

If a parameter was defined as numeric, an attempt to send an alphanumeric value causes an error and the corresponding message is sent back to the requestor.

DRS – DATABASE REST SERVICES

GET

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL

```
1 {
2   "clientid": "2"
3 }
```

Body Cookies Headers (4) Test Results

```
1 {
2   "time": "25-08-2021 12:34:03.087",
3   "message": "Parameter clientid must be numeric; ",
4   "status": "error"
5 }
```

5.3 /servicesh

This endpoint has the same functionality of /services, the difference is all parameters are sent in the message header instead of the message body.

5.4 /servicesPut

This endpoint is to be used when a service contains a statement to insert a record into the database. Only the message body is valid to inform the parameters to be used. The same validation as present in /service and /serviceh is performed for the parameters. In the following example, a service named service32 has been created. This service updates the credit limit for a specific client.

Service

Service Name *

Server Name *

Database Name

Description

Service Type ☒ SQL ☐ URL

Avg. Response Time Alert

Max. Response Time Alert

Rows Per Request

Action *

Role Name

Return Data Only * ☐ Yes ☒ No

Parameter Type * ☐ Header ☒ Body

Security Enabled * ☐ Yes ☒ No

Stored Statistics * ☒ Yes ☐ No

Status * ☒ Active ☐ Suspended

Update Date 25/08/2021 12:47:45

DRS – DATABASE REST SERVICES

The parameter page defined the two numeric fields and both fields are required when a request to this server is sent.

Service Parameters								
Query								
	Seq	Parameter Name	Parameter Display	Edit	Length	Decimals	Required	Update Date
	10	credit	Credit Limit	N	8	2	Y	25/08/2021 12:48:08
	20	clientid	clientid	N	4	0	Y	25/08/2021 12:48:21

When a request is submitted, the result is sent to the message body. The number of records updated is stored in the **records** field, and the **status** field has the value **ok**.

PUT

http://192.168.0.199:8095/Drs/servicesPut?service=service32

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

2

3

4

1

2

3

4

5

6

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

```
"records": 1,  
"responseTime": 0.0019321,  
"time": "25-08-2021 13:04:17.602",  
"status": "ok"
```

When informing an invalid client id, the **status** field has the value **notok** and the **records** field has the value **0**.

DRS – DATABASE REST SERVICES

The screenshot shows a REST client interface. The top bar has a dropdown menu set to 'PUT' and a text field with the URL 'http://192.168.0.199:8095/Drs/servicesPut?service=service32'. A blue 'Send' button is on the right. Below the URL bar, there are tabs for 'Params', 'Authorization', 'Headers (8)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Body' tab is selected, and the body type is set to 'JSON'. The request body is a JSON object:

```
{  "clientId": 112,  "credit": 123.55}
```

. Below the request body, there are tabs for 'Body', 'Cookies', 'Headers (5)', and 'Test Results'. The 'Body' tab is selected, and the response is shown in 'Pretty' format as a JSON object:

```
{  "records": 0,  "responseTime": 0.0015859,  "time": "25-08-2021 13:07:17.563",  "status": "notok"}
```

5.5 /servicesPost

This endpoint is to be used when a service contains a SQL statement to insert a new record into the database. The same validation is made for the parameters informed in the message body.

As in **/servicePut**, the status field has three values: **ok**, for successful operations, **notok** for errors when no records are inserted, and **error** if when the request cannot be completed due to validation or database constraints. The image below shows a service that inserts records in the **clien** table.

The screenshot shows a 'Service' configuration form. The 'Service Name' is 'service35'. The 'Server Name' is 'LAPTOP-D6JBG082'. The 'Database Name' is 'ORACLE1'. The 'Description' is 'Insert new Clie record'. The 'Service Type' is 'SQL'. The 'Avg. Response Time Alert' is '0.0000'. The 'Max. Response Time Alert' is '0.0000'. The 'Rows Per Request' is '0'. The 'Action' is a SQL statement:

```
Insert into CLIEN
(
  CLIENT_ID
  ,CLIENT_NAME
  ,ADDRESS
  ,CITY
  ,STATE
  ,COUNTRY
  ,CREDIT_LIMIT
) values (
  :clientId
  ,:name
  ,:address
  ,:city
  ,:state
  ,:country
  ,:credit
)
```

. The 'Role Name' is 'Select One'. The 'Return Data Only' is 'No'. The 'Parameter Type' is 'Body'. The 'Security Enabled' is 'No'. The 'Stored Statistics' is 'No'. The 'Status' is 'Active'.

DRS – DATABASE REST SERVICES

The parameters must be defined for this statement.

Service Parameters								
Query								
	Seq	Parameter Name	Parameter Display	Edit	Length	Decimals	Required	Update Date
	10	clientid	clientid	N	4	0	Y	26/08/2021 08:45:44
	20	name	name	A	40	0	Y	26/08/2021 08:46:08
	30	address	address	A	60	0	Y	26/08/2021 08:46:23
	40	city	city	A	30	0	Y	26/08/2021 08:46:36
	50	state	state	A	2	0	Y	26/08/2021 08:46:44
	60	country	country	A	30	0	Y	26/08/2021 08:47:00
	70	credit	credit	N	8	2	Y	26/08/2021 08:47:16

Using Postman, processing this service with the correct data returns a status field with the value **ok**. If one of the required fields is missing, an error will be returned.

POST

http://192.168.0.199:8095/Drs/servicesPost?service=service35

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

2

3

4

5

6

7

8

"credit": .444.02,

"city": "RIO DE JANEIRO",

"clientid": 12,

"name": "CLIENT NO. 11",

"address": "AV BRASIL",

"state": "RJ"

Body

Cookies

Headers (4)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

"time": "26-08-2021 09:04:18.288",

"message": "Parameter country is required; ",

"status": "error"

If there is an error when the statement is processed by the database server, the corresponding message will be sent to the requestor.

DRS – DATABASE REST SERVICES

The screenshot shows a REST client interface with a POST request to `http://192.168.0.199:8095/Drs/servicesPost?service=service35`. The request body is a JSON object with the following fields:

```
1 {
2   "credit": 900.02,
3   "city": "RIO DE JANEIRO",
4   "country": "BRAZIL",
5   "clientid": 12,
6   "name": "CLIENT NO. 12",
7   "address": "AV ATLANTICA",
8   "state": "RJ"
9 }
```

The response body is also in JSON format:

```
1 {
2   "time": "26-08-2021 09:52:35.974",
3   "message": "ORA-00001: unique constraint (DMS.CLIEN_U1) violated\n",
4   "status": "error"
5 }
```

5.6 /servicesDelete

This endpoint is to be used when a service contains SQL statement to delete a record from the database. The image below shows a service that deletes records in the **clien** table.

The screenshot shows the 'Service' configuration form. The fields are as follows:

- Service Name *: `service36`
- Server Name *: `LAPTOP-D6JBG082`
- Database Name: `ORACLE1`
- Description: `Delete records in Clie`
- Service Type: ☒ SQL ☐ URL
- Avg. Response Time Alert: `0.0000`
- Max. Response Time Alert: `0.0000`
- Rows Per Request: `0`
- Action *: `delete from clien where client_id=:clientid`
- Role Name: `Select One`
- Return Data Only *: ☐ Yes ☒ No
- Parameter Type *: ☐ Header ☒ Body
- Security Enabled *: ☐ Yes ☒ No
- Stored Statistics *: ☒ Yes ☐ No
- Status *: ☒ Active ☐ Suspended
- Update Date: `26/08/2021 09:56:07`

At the bottom, there are buttons for `Back`, `New`, `Delete`, `Save`, and `Parameters`.

DRS – DATABASE REST SERVICES

The parameter page for this service is displayed below.

Service Parameters								
Query								
	Seq	Parameter Name	Parameter Display	Edit	Length	Decimals	Required	Update Date
	10	clientid	clientid	N	4	0	Y	26/08/2021 09:56:19

Using Postman, processing this service with the correct data returns a status field with the value **ok** and the records field contains the number of records affected.

The screenshot shows a Postman interface for a DELETE request. The URL is `http://192.168.0.199:8095/Drs/servicesDelete?service=service36`. The request body is a JSON object: `{ "clientid": 12 }`. The response body is a JSON object: `{ "records": 1, "responseTime": 0.0028147, "time": "26-08-2021 10:02:23.808", "status": "ok" }`.

If the request tries to delete a non-existing record, the status field will contain the value **notok**.

The screenshot shows a Postman interface for a DELETE request. The URL is `http://192.168.0.199:8095/Drs/servicesDelete?service=service36`. The request body is a JSON object: `{ "clientid": 12 }`. The response body is a JSON object: `{ "records": 0, "responseTime": 0.0010376, "time": "26-08-2021 10:06:56.125", "status": "notok" }`.

DRS – DATABASE REST SERVICES

6 Installation

The current version of DRS uses a web application, a batch process(daemon), and an Oracle database. All files needed for the installation process can be downloaded from Github.

The Database Rest Services works using Unisys MCPSQL or Unisys JDBC. One of these software must be installed on the mainframe and the databases must be configured in their config files.

It is possible to use a relational database, like Oracle, to create services to be handled by DRS.

- **Creating the schema**

The first step to install the application is creating the database schema. DRS uses an Oracle database to store configuration settings and activity. The application can use any Oracle database version, including Oracle Express.

You need to create a specific username to install the database objects. This username and password will be used to update the drs.properties later.

When connected using the username created, you can execute the script DRS.sql. This script creates all database objects and inserts an initial record in the USERS table related to the user **ADMIN** and password **admin**. This user can be used to connect to the Web application with administrator privilege and configure the setup of the environment.

The script DRS_drop.sql can be used to drop all objects created.

- **Installing the Web application**

The war file contains an embedded Tomcat. The command line to start the application is:

```
java -jar Drs.war
```

Some additional information is needed to inform the application of some configurations.

Spring properties

Spring properties configuration can be stored in any folder. The drs.properties sample file is available on GitHub. Parameters must be changed according to the environment that is being configured, like the parameters to access the Oracle database.

To inform the application which configuration file to use, a java argument must be defined to when the application is started:

DRS – DATABASE REST SERVICES

```
java -Dspring.config.location=<folder name>\drs.properties Drs.war
```

SWDIR_SETUP

This variable must contain the folder where the configuration files will be stored.

If you are using the Unisys JDBC to connect to DMSII databases, you need to configure one property file per database. The property file name uses the same name you created using the Databases page.

The database name defined on the DRS application can be different from the existing DMSII database on the mainframe. The property file has a Connect String and that is the place you assign the existing DMSII database name. For example, you can define a database TEST using the Databases page and create a property file **TEST.properties** with a connect string assigned to the TESTDB:

```
connect_string=jdbc:unisys:mcpsql:Unisys.DMSII:resource=DBTST;host=192.168.16.5;port=2012
```

If you are using the Unisys MCPSQL to connect to DMSII databases, you need to configure the parameters in a file called **mcpsql.properties**.

The sample properties files are present in Github.

configDir

This property can be used when executing the java with -D option. The use of this variable is an alternative to SWDIR_SETUP.

```
java -DconfigDir=<folder name>  
-Dspring.config.location=<folder name>\drs.properties Drs.war
```