Sailfish User Manual - Exactpro

Version Control

Version:	9.1
Release date:	12/08/2020
Pages:	62

Contents:

Contents: 2
1 Introduction 3
2 Prerequisites for Installation 4
  2.1 Operating System 4
  2.2 Software Requirements 4
  2.3 Hardware Requirements 4
  2.4 Browser Requirements 4
3 Installation 5
  3.1 Installation 5
    3.1.1 Linux 5
    3.1.2 Windows 6
  3.2 Sailfish update 6
  3.3 Several Sailfish WEB UIs under one Tomcat 6
    3.3.1 Linux 6
    3.3.2 Windows 8
4 Start Up and Shut Down 9
  4.1 Sailfish Backend Start 9
  4.2 Sailfish Backend Shutdown 9
  4.3 Sailfish WEB UI 9
5 Sailfish Web UI 10
  5.1 Environment Page 10
  5.2 Test Script Page 13
  5.3 Messages Page 15
  5.4 Download Page 19
  5.5 Dictionaries Page 20
    5.5.1 Dictionary Creation Using Sailfish GUI 21
    5.5.2 Dictionary validation 27
    5.5.3 Dictionary update 28
  5.6 Help Page 29
  5.7 Configuration Page 32
    5.7.1 Update Version 33
    5.7.2 Database 33
    5.7.3 Logging 34
    5.7.4 Workspace 37
    5.7.5 User Events 40
    5.7.6 Statistics DB 40
    5.7.7 Flight Recorder 42
    5.7.8 NetDumper 42
    5.7.9 Machine Learning 43
    5.7.10 Email Notification 44
    5.7.11 Cleanup 45
    5.7.12 Big Button 46
  5.8 Big Button Page 47
6 Sailfish Test Scripts basics 50
  6.1 Test Actions 50
    6.1.1 ‘Send’ action 50
    6.1.2 ‘Receive’ action 51
    6.1.3 ‘Retrieve’ action 51
    6.1.4 ‘Count’ action 51
    6.1.5 Other actions 51
  6.2 Test Case Structure 52
  6.3 Test Case Syntax 54
  6.4 Test Case Formulae 54
7 Common services 56
  7.1 Integration with DBMS 56
8 Test run results 59
9 Test failures investigation 60

1 Introduction

Sailfish is a test automation tool for testing bi-directional message flows in distributed trading platforms and market data delivery systems. The tool interacts with system under test via client-facing and internal protocols, including REST API. The purpose of Sailfish is to minimize manual intervention required to execute test suites. In its more sophisticated deployments, Sailfish makes it possible to achieve fully autonomous ordered test execution that does not require ongoing operator monitoring.

Sailfish has a modular structure, whereby a shared framework is used in conjunction with specialized plug-ins. Separate plug-ins are used for each project and include necessary protocol services, such as industry-standard protocols (e.g. FIX) and proprietary protocols.

2 Prerequisites for Installation

Sailfish is a cross-platform application. This section provides information regarding the recommended configuration of the system to install Sailfish.

2.1 Operating System

The prerequisites for installation are as follows:

1. Windows 32 bit (XP or higher) / Suse Linux (Enterprise Server 11 or higher);

2.2 Software Requirements

1. JDK 1.8u201 or higher
2. Tomcat 9.0.17 or higher (you can download tar.gz distributive from tomcat.apache.org)
3. Supported DBMS
   - MySQL server 5.5 or higher (excluding 8.0 or higher);
   - Postgresql 9.1 or higher;
   - Any DBMS with supporting JDBC driver and Hibernate Dialect.

2.3 Hardware Requirements

The hardware requirements for successful installation are as follows:

1. Dual core processor 2.5 GHz (Intel, AMD);
2. 4GB RAM (recommended);
3. 10 GB HDD.

2.4 Browser Requirements

Google Chrome 65.0.3325.181 or higher.

3 Installation

Before starting the installation of Sailfish on a computer, make sure that it meets the requirements outlined in section 2. Make sure that MySQL/PostgreSQL and Tomcat servers have been deployed.

3.1 Installation

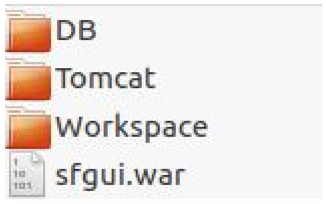
3.1.1 Linux

1. Create 2 folders on any disc with a sufficient amount of space: Sailfish/ - it will contain the application itself, and tmp/ – a temporary folder for archives.
2. Download the Tomcat server (apache-tomcat-8.x.xx or higher, e.g.: http://mirror.linux-ia64.org/apache/tomcat/tomcat-9/v9.0.12/bin/ pache-tomcat-9.0.12.tar.gz).
3. Unpack it to the Sailfish folder created in step 1 above.

NOTE!: Please note that Apache Tomcat is not a part of the Sailfish installation archive and should be installed separately (https://tomcat.apache.org/download-90.cgi).

4. Download a set of Sailfish installation files – the core distributive for Sailfish and the required plug-ins. (Alternatively, you can request them from GitHub https://github.com/Exactpro/sailfish-core).

Usually, the archive with the core set of files contains the following: DB, Tomcat, Workspace folders and a file named sfgui.war.



DB
Tomcat
Workspace
sfgui.war

5. Unzip the archive with the core set of files to the tmp/ folder created in step 1 above.
6. Create folder in <Deployed Tomcat>/webapps/demogui.
7. Extract the war archive from <tmp>/sfgui.war file to the <Deployed Tomcat>/webapps/demogui folder.
8. Extract the zipped archive with Sailfish plugins to the <Deployed Tomcat>/webapps/demogui folder.
9. The <tmp>/DB folder contains different scripts for creating databases. There are ready-made sets of scripts for MySQL and PostgreSQL.
10. For example, you can choose the create_mysql_db.sh file to define the correct parameters of the database. It may also be required to set up a “superuser” password. You can pass the required options as script arguments (create_mysql_db.sh -superpaassword <super user name>).
11. Save the changes to the create_mysql_db.sh file and launch it to create the DB for Sailfish.

NOTE!: This procedure should be performed after each version update of Sailfish before launching the new version of it.

12. Launch the startup.sh file.
13. Sailfish GUI will be available via the following link in your browser: http://localhost:8080/demogui.

3.1.2 Windows

If you have the Windows operating system - you will need to run .bat batches instead of .sh ones. All other operations should be the same as in section 3.1.1.

3.2 Sailfish update

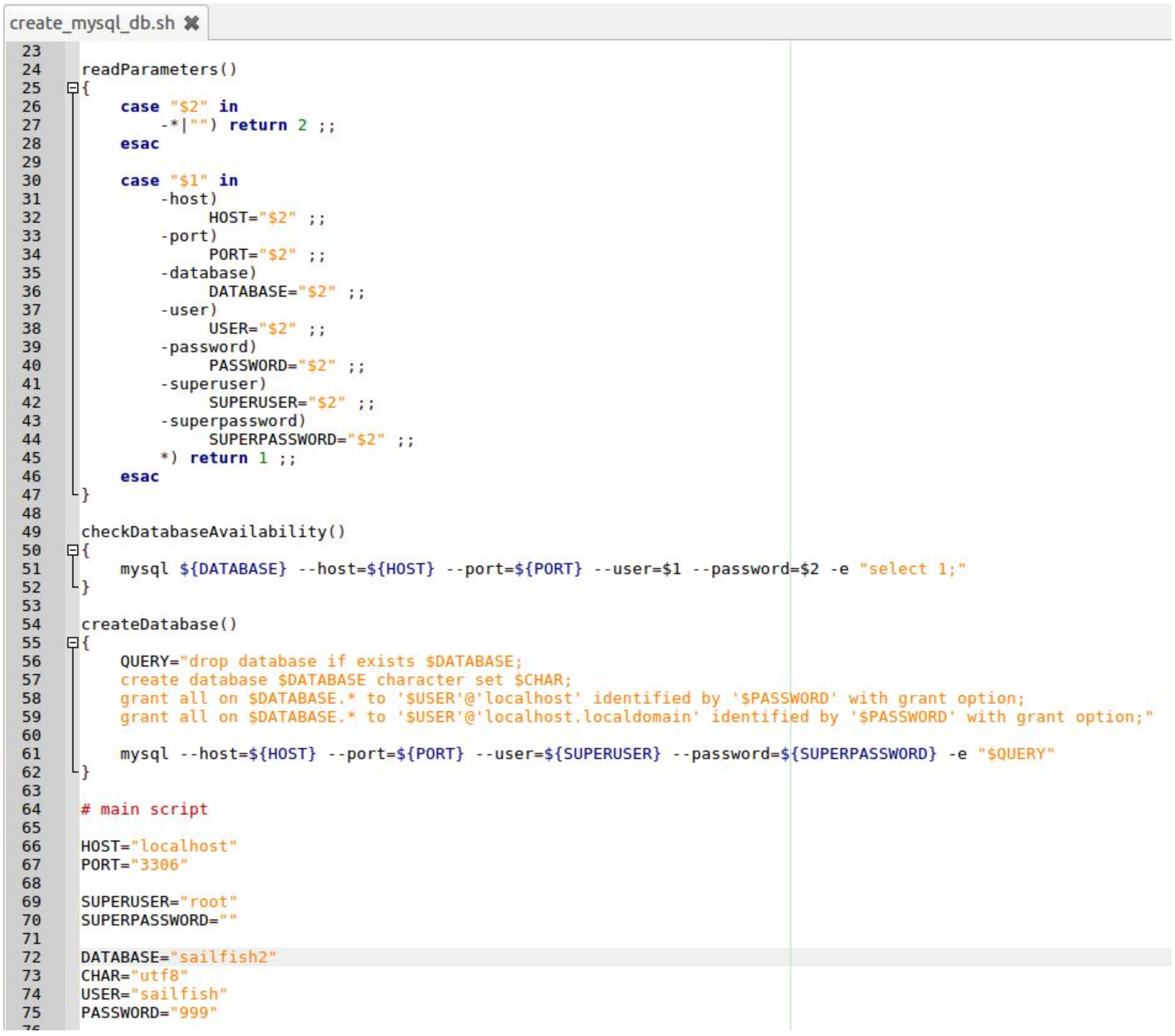
1. Shut down Sailfish by launching shutdown.bat/sh.
2. Create a back-up folder in Sailfish/.
3. Back up the files from <Deployed Tomcat>/webapps/demogui folder to the back-up folder (this allows using the current build if a new build has issues).
4. Download the sfgui.war archive installation file (Alternatively, you can request it from GitHub https://github.com/Exactpro/sailfish-core).
5. Extract the sfgui.war archive file to the <Deployed Tomcat>/webapps/demogui folder replacing the earlier files.
6. Start Sailfish by launching startup.bat/sh.
7. Sailfish GUI will be available via the following link in your browser: http://localhost:8080/demogui/.

3.3 Several Sailfish WEB UIs under one Tomcat

3.3.1 Linux

It’s required to configure a separate demogui folder in the Sailfish/apache-tomcat-8.5.33/webapps folder and configure separate DBs for each Sailfish.

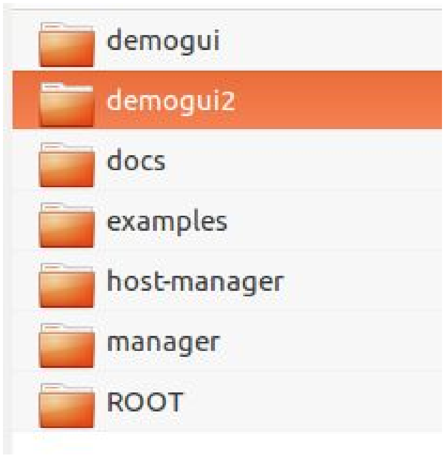
1. To create a new DB, the create_mysql_db.sh file can be used (see section 3.1. point 8). Update the .sh file with the new DB name (e.g. sailfish2) and execute the .sh file:


create_mysql_db.sh

```
readParameters()
{
    case "$2" in
        -*|"") return 2 ;;
    esac

    case "$1" in
        -host)
             HOST="$2" ;;
        -port)
             PORT="$2" ;;
        -database)
             DATABASE="$2" ;;
        -user)
             USER="$2" ;;
        -password)
             PASSWORD="$2" ;;
        -superuser)
             SUPERUSER="$2" ;;
        -superpassword)
             SUPERPASSWORD="$2" ;;
        *) return 1 ;;
    esac
}

checkDatabaseAvailability()
{
    mysql ${DATABASE} --host=${HOST} --port=${PORT} --user=$1 --password=$2 -e "select 1;"
}

createDatabase()
{
    QUERY="drop database if exists $DATABASE;
    create database $DATABASE character set $CHAR;
    grant all on $DATABASE.* to '$USER'@'localhost' identified by '$PASSWORD' with grant option;
    grant all on $DATABASE.* to '$USER'@'localhost.localdomain' identified by '$PASSWORD' with grant option;"

    mysql --host=${HOST} --port=${PORT} --user=${SUPERUSER} --password=${SUPERPASSWORD} -e "$QUERY"
}

# main script

HOST="localhost"
PORT="3306"

SUPERUSER="root"
SUPERPASSWORD=""

DATABASE="sailfish2"
CHAR="utf8"
USER="sailfish"
PASSWORD="999"
```

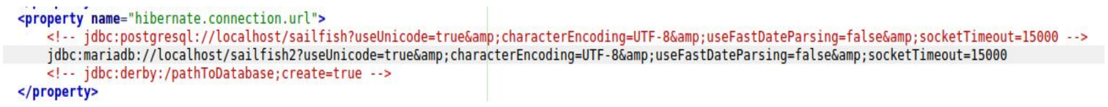
2. Navigate to the Sailfish/apache-tomcat-8.5.33/webapps folder.

3. Copy the /demogui folder and rename it, e.g. demogui2.



4. Navigate to the demoqui2/cfg folder and update the hibernate.cfg.xml file as follows:

update the ‘hibernate.connection.url’ section with the corresponding new DB name:



```
<property name="hibernate.connection.url">
    <!-- jdbc:postgresql://localhost/sailfish?useUnicode=true&amp;characterEncoding=UTF-8&amp;useFastDateParsing=false&amp;socketTimeout=15000 -->
    jdbc:mariadb://localhost/sailfish2?useUnicode=true&amp;characterEncoding=UTF-8&amp;useFastDateParsing=false&amp;socketTimeout=15000
    <!-- jdbc:derby:/pathToDatabase;create=true -->
</property>
```

5. Navigate to the Sailfish/apache-tomcat-x.x.xx/bin/ directory and launch the startup.sh file.
6. Both Sailfish UIs (demogui and demogui2) will be available in your browser:
demogui - on http://localhost:8080/demogui/
demogui2 - on http://localhost:8080/demogui2/

NOTE!: A restart of Tomcat triggers a restart of all Sailfish applications.

3.3.2 Windows

If you have the Windows operating system - you will need to run .bat batches instead of .sh ones. All other operations should be the same as in section 3.3.1.

4 Start Up and Shut Down

4.1 Sailfish Backend Start

1. Navigate to the <Deployed Tomcat>\bin folder;
2. Run the “startup.sh” file.

If you have the Windows operating system, - you will need to run .bat batches instead of .sh ones. All other operations should be the same.

4.2 Sailfish Backend Shutdown

1. Navigate to the <Deployed Tomcat>\bin folder;
2. Run the “shutdown.sh” file.

If you have the Windows operating system, - you will need to run .bat batches instead of .sh ones. All other operations should be the same.

4.3 Sailfish WEB UI

Open the URL in your browser (e.g. http://localhost:8080/sfgui/) where:

- host is the IP of the computer where Sailfish back-end was started (usually, it is localhost);
- port is the Apache port. The default value is 8080 and can be modified in the “server.xml” file of <Deployed Tomcat>\conf\ in the "Connector port="8080" protocol="HTTP/1.1" line;
- sfgui – is a directory with extracted “sfgui.war” file from the Sailfish core distributive.

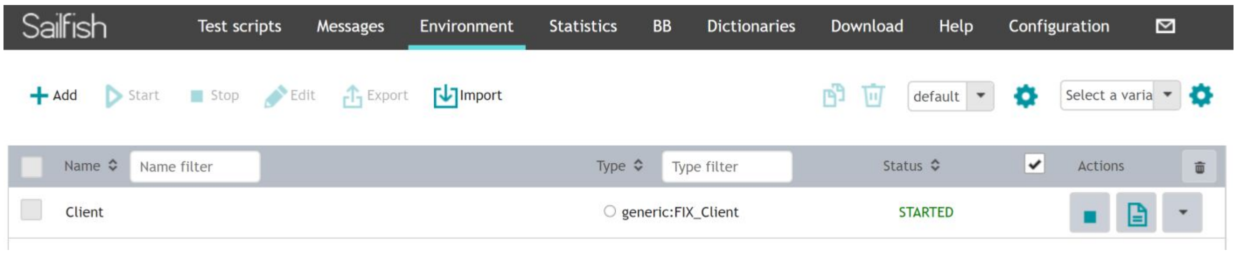
The Main Sailfish web page will be displayed.

5 Sailfish Web UI

5.1 Environment Page

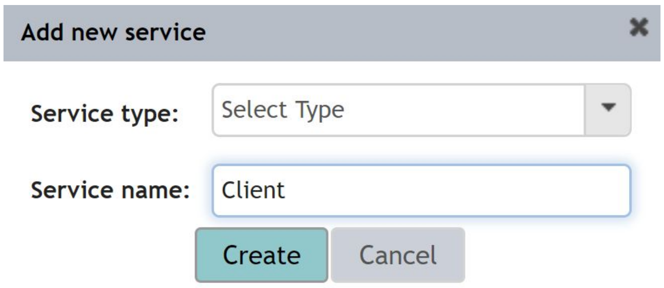
The "Environment" tab in Sailfish allows the user to manage the services and their parameters. "Services" in Sailfish are used for configuring settings for specific connections.

If there is a need to verify the same business scenario on another system – it is not necessary to re-build your test script – you can simply change the connection to the environment and run the existing script without any modifications.



Name	Type	Status	Actions
Client	generic:FIX_Client	STARTED	

A new service can be added by pressing the "Add" button:

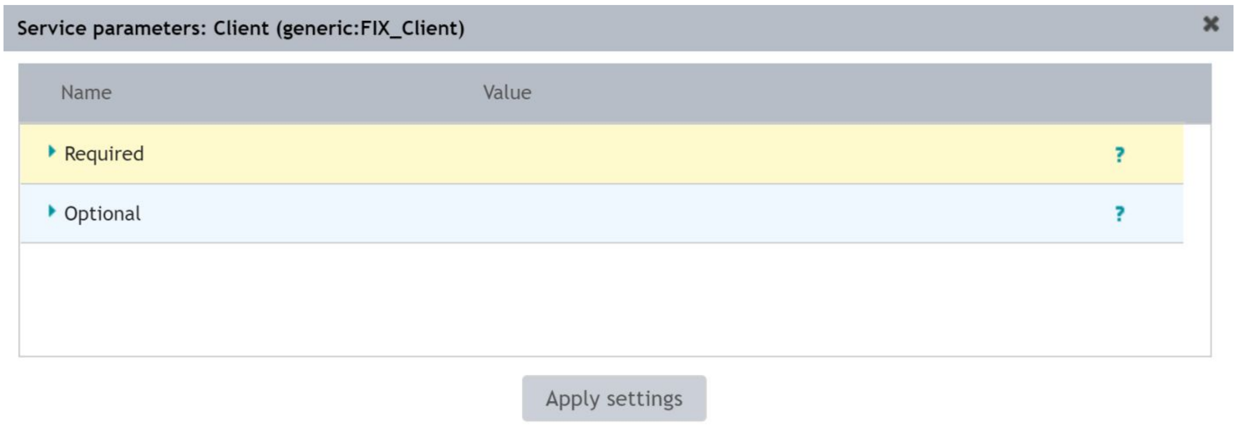


Add new service

Service type: Select Type

Service name: Client

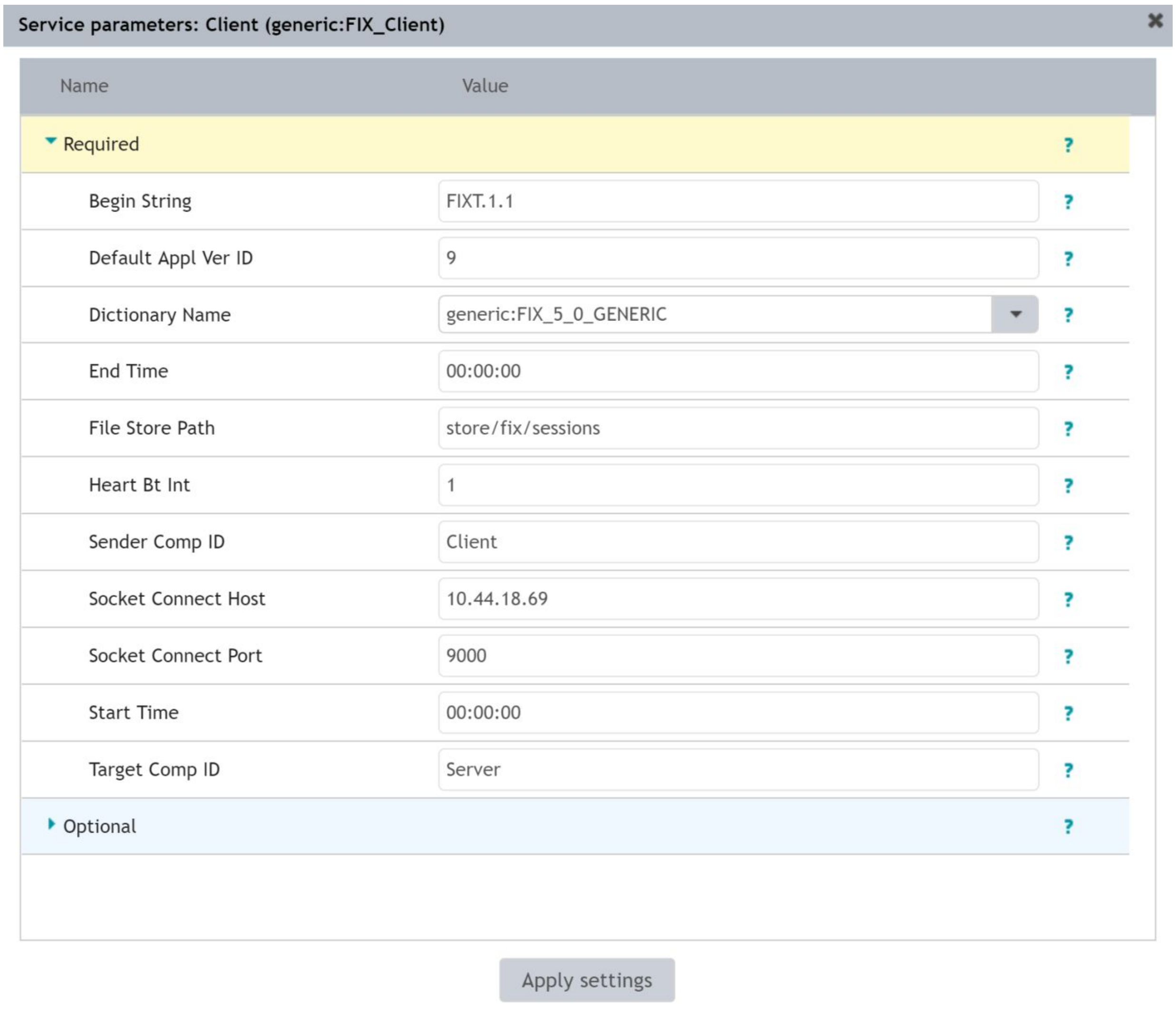
Service parameters also need to be specified. There are Required and Optional parameters. The Required parameters will be taken by Sailfish as key values.



Service parameters: Client (generic:FIX_Client)

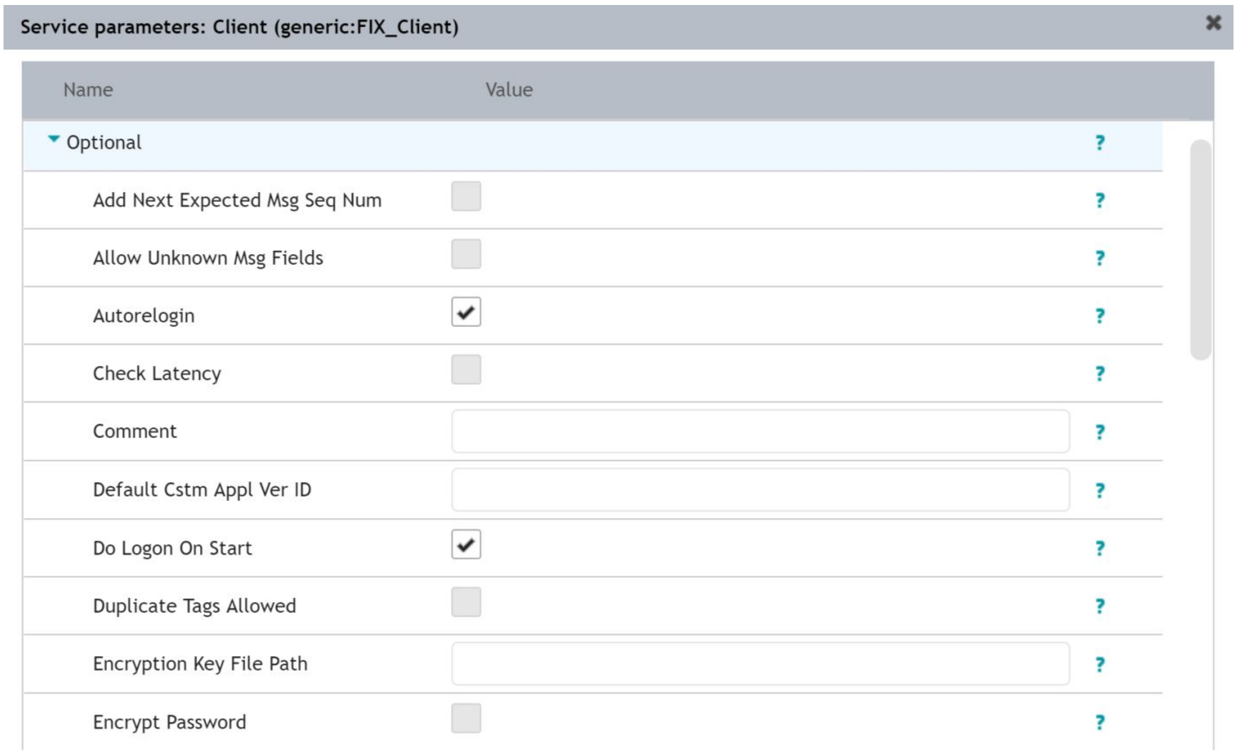
Name	Value
Required	
Optional	

Below is the example of a Test Service Required and Optional parameters setup for a FIX client.



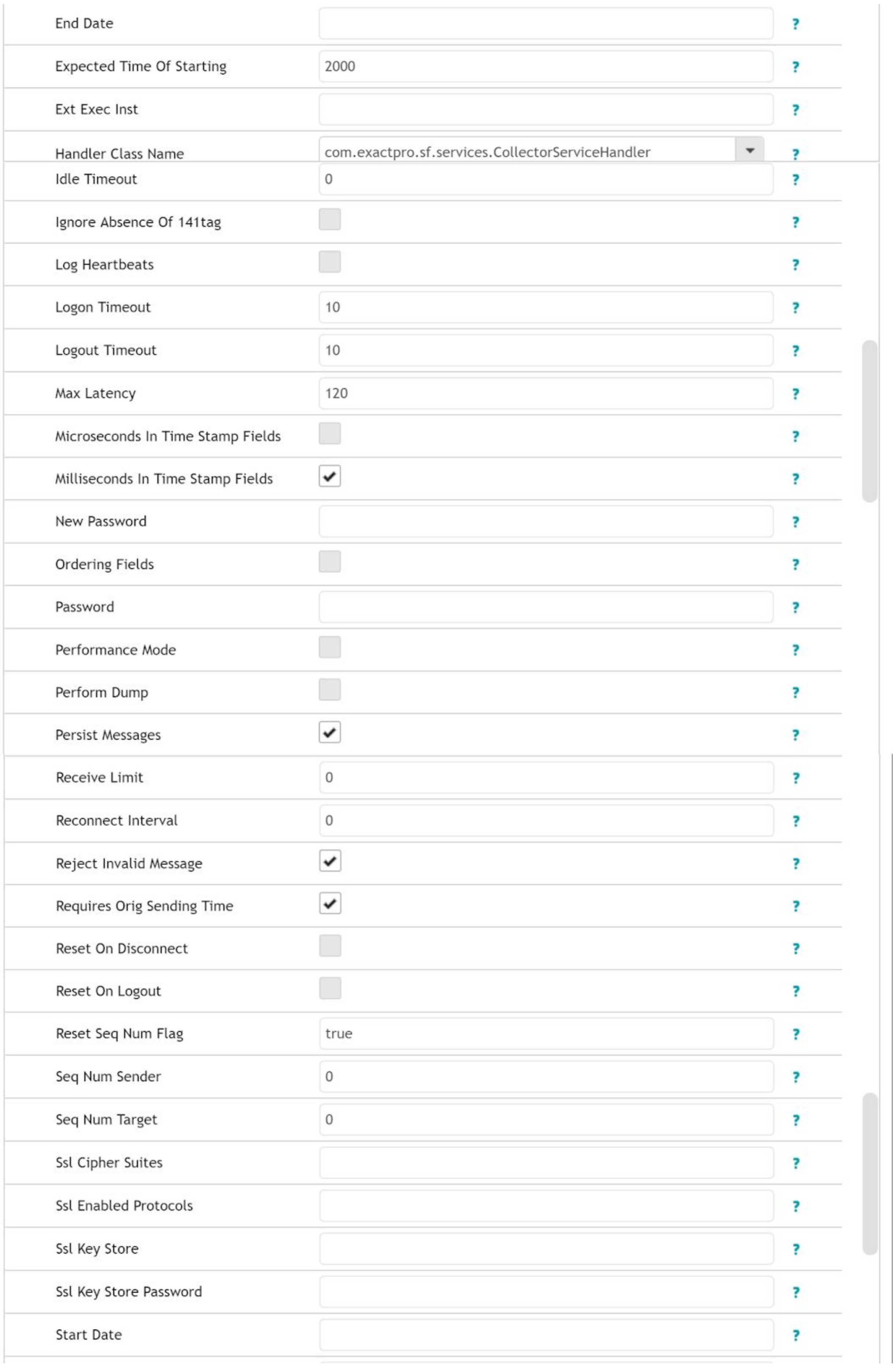
Service parameters: Client (generic:FIX_Client)

Name	Value
Required	
Begin String	FIXT.1.1
Default Appl Ver ID	9
Dictionary Name	generic:FIX_5_0_GENERIC
End Time	00:00:00
File Store Path	store/fix/sessions
Heart Bt Int	1
Sender Comp ID	Client
Socket Connect Host	10.44.18.69
Socket Connect Port	9000
Start Time	00:00:00
Target Comp ID	Server
Optional	

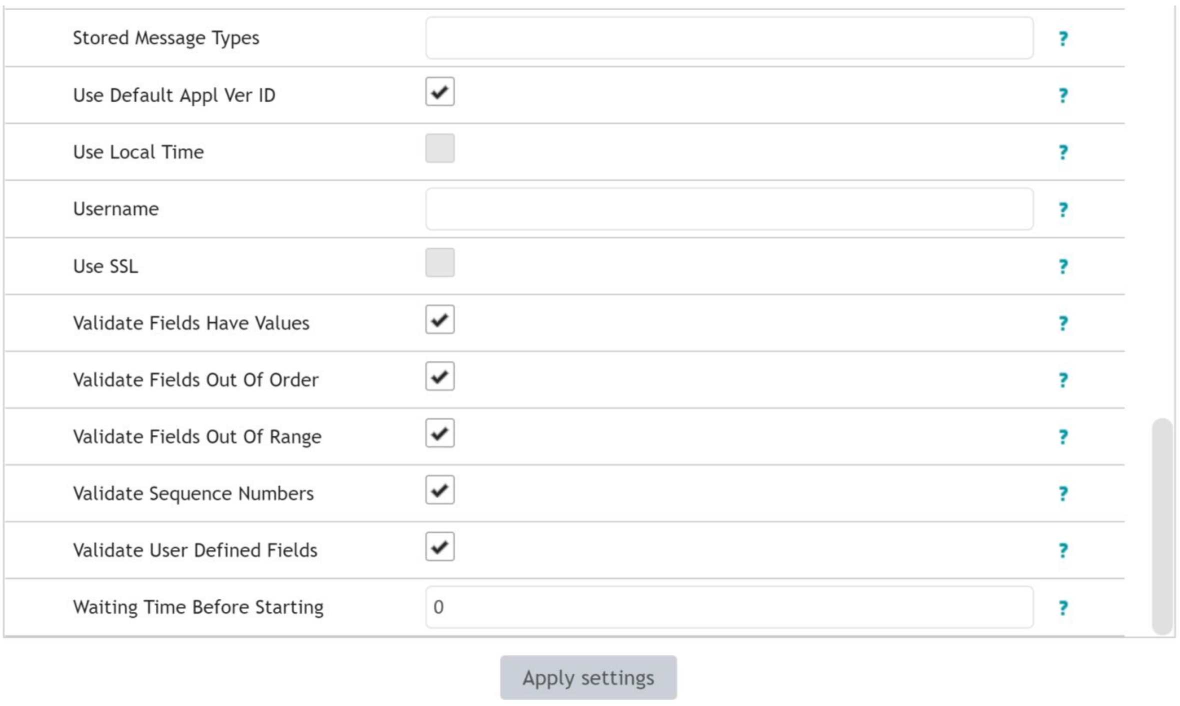


Service parameters: Client (generic:FIX_Client)

Name	Value
Optional	
Add Next Expected Msg Seq Num	☐
Allow Unknown Msg Fields	☐
Autorelogin	✓
Check Latency	☐
Comment	
Default Cstm Appl Ver ID	
Do Logon On Start	✓
Duplicate Tags Allowed	☐
Encryption Key File Path	
Encrypt Password	☐

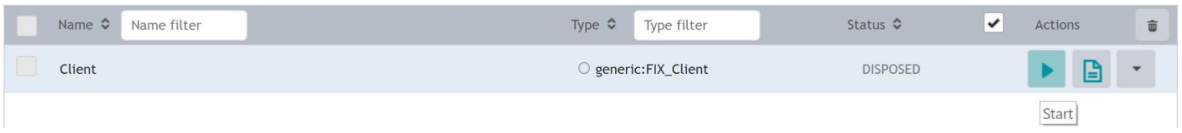


End Date	
Expected Time Of Starting	2000
Ext Exec Inst	
Handler Class Name	com.exactpro.sf.services.CollectorServiceHandler
Idle Timeout	0
Ignore Absence Of 141tag	☐
Log Heartbeats	☐
Logon Timeout	10
Logout Timeout	10
Max Latency	120
Microseconds In Time Stamp Fields	☐
Milliseconds In Time Stamp Fields	✓
New Password	
Ordering Fields	☐
Password	
Performance Mode	☐
Perform Dump	☐
Persist Messages	✓
Receive Limit	0
Reconnect Interval	0
Reject Invalid Message	✓
Requires Orig Sending Time	✓
Reset On Disconnect	☐
Reset On Logout	☐
Reset Seq Num Flag	true
Seq Num Sender	0
Seq Num Target	0
Ssl Cipher Suites	
Ssl Enabled Protocols	
Ssl Key Store	
Ssl Key Store Password	
Start Date	



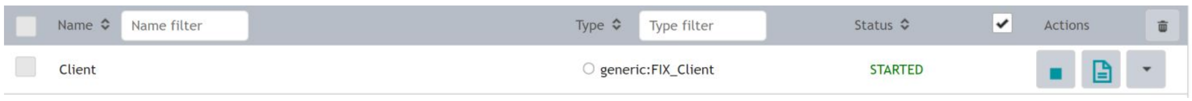
Stored Message Types		?
Use Default Appl Ver ID	✓	?
Use Local Time		?
Username		?
Use SSL		?
Validate Fields Have Values	✓	?
Validate Fields Out Of Order	✓	?
Validate Fields Out Of Range	✓	?
Validate Sequence Numbers	✓	?
Validate User Defined Fields	✓	?
Waiting Time Before Starting	0	?

Once the environment is configured, the connection can be started by pressing “Start”.



Name	Type	Status	Actions
Client	generic:FIX_Client	DISPOSED	

If it is started successfully, you will see the “Started” status.

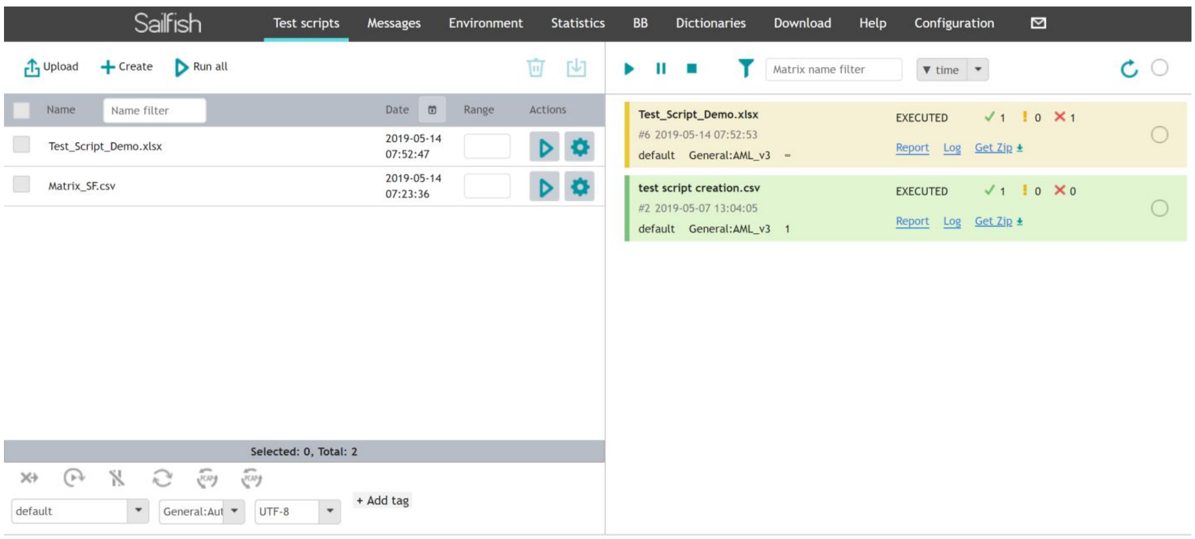


Name	Type	Status	Actions
Client	generic:FIX_Client	STARTED	

If the connection configuration is incorrect, the status will be ERROR.

If a connection can't be established at the current time, the status will be WARNING.

5.2 Test Script Page



Name	Date	Range	Actions
Test_Script_Demo.xlsx	2019-05-14 07:52:47		
Matrix_SF.csv	2019-05-14 07:23:36		

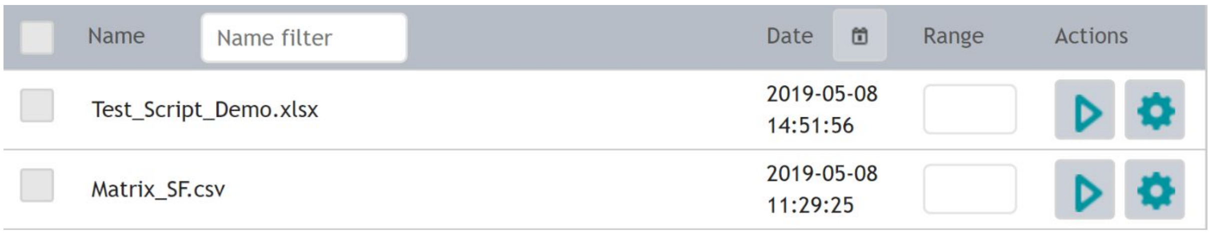
Selected: 0, Total: 2

Test_Script_Demo.xlsx EXECUTED ✓ 1 ! 0 ✕ 1
#6 2019-05-14 07:52:53 Report Log Get Zip
default General:AML_v3 -

test script creation.csv EXECUTED ✓ 1 ! 0 ✕ 0
#2 2019-05-07 13:04:05 Report Log Get Zip
default General:AML_v3 1

The Test Scripts tab allows the user to perform the following:

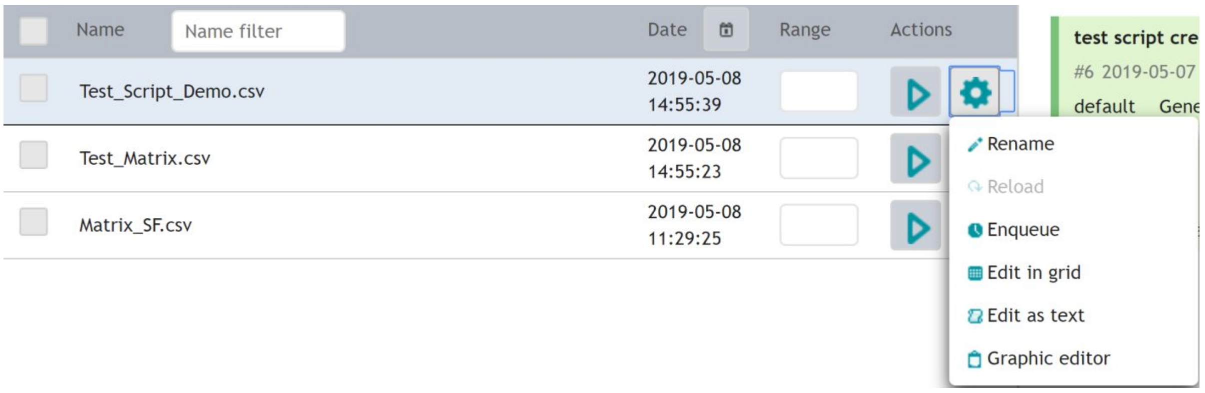
1. Upload an existing matrix (a CSV/XLS/XLSX file): the user can see all uploaded matrices as a list; the name, date and time of uploading are indicated;



Name	Name filter	Date	Range	Actions
Test_Script_Demo.xlsx		2019-05-08 14:51:56		
Matrix_SF.csv		2019-05-08 11:29:25		

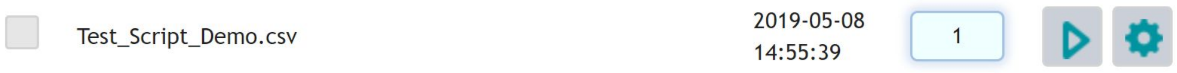
2. Create an empty form, so that the users' own test cases could be written within Sailfish editors;
3. Manage the list of uploaded or created matrices, i.e. to delete, edit, and launch a few selected matrices, a single matrix, or a group of matrices for execution. There are two options for launching the matrices for executions in the management panel at the bottom of the page:
   - Continue if Failed (if disabled, Sailfish will stop executing a test script after the first failed case and skip all the remaining test cases. Otherwise, the test script will execute all test cases in a script irrespective of failures);
   - Autostart (if enabled, the test services required for execution of a test script will be started (and stopped after script execution) automatically. Otherwise, the user will have to start the services manually from the 'Environment' tab);

- If the user selects a single matrix, the following options are available:
  - Edit the matrix in Sailfish editors: as plain text/grid editor;
  - Edit the matrix in an Excel table;
  - Launch the matrix for execution.



Name	Name filter	Date	Range	Actions
Test_Script_Demo.csv		2019-05-08 14:55:39		
Test_Matrix.csv		2019-05-08 14:55:23		
Matrix_SF.csv		2019-05-08 11:29:25		

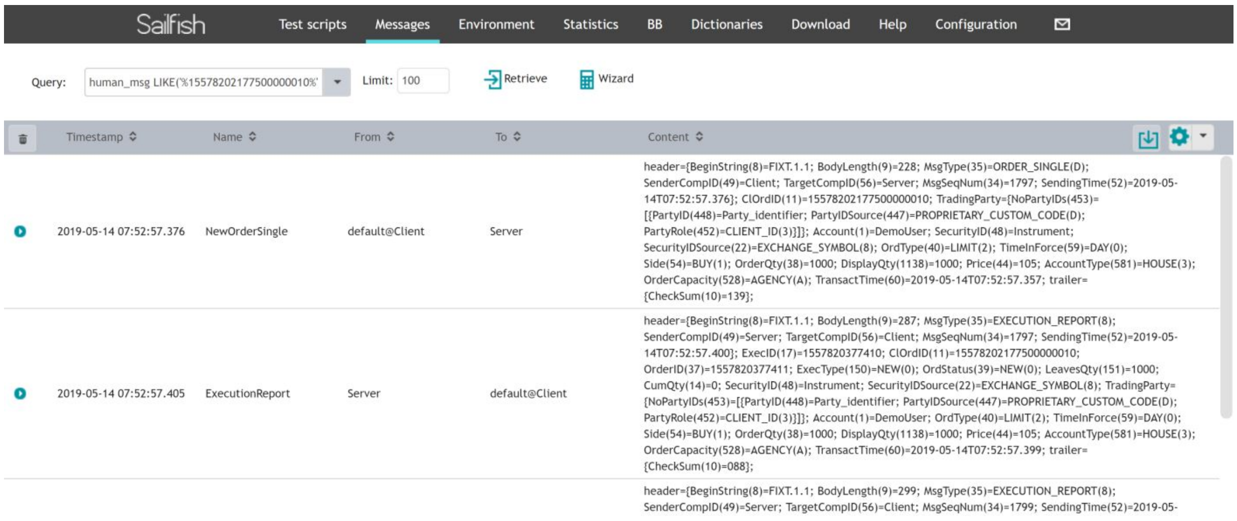
- Either the entire matrix can be launched, or certain test cases within a matrix can be selected and launched for execution (in the text panel opposite the script name);



Test_Script_Demo.csv	2019-05-08 14:55:39	1	

- View the list of matrices that have been run or enqueued for execution;
- For matrices that have been executed, you are able to view an executed test case report. The report contains step-by-step information about the outcome of each test case and each sent and received message.

5.3 Messages Page

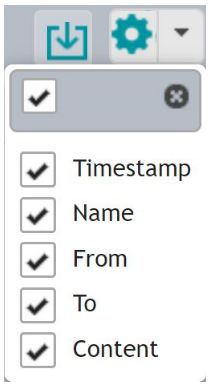


Query: human_msg LIKE('%15578202177500000010%' Limit: 100

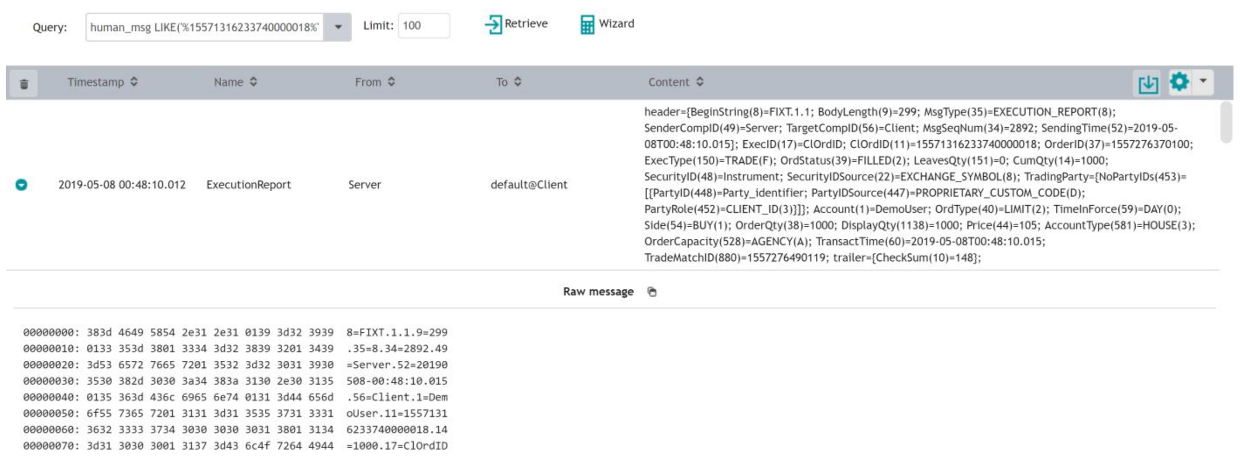
Timestamp	Name	From	To	Content
2019-05-14 07:52:57.376	NewOrderSingle	default@Client	Server	header={BeginString(8)=FIXT.1.1; BodyLength(9)=228; MsgType(35)=ORDER_SINGLE(D); SenderCompID(49)=Client; TargetCompID(56)=Server; MsgSeqNum(34)=1797; SendingTime(52)=2019-05-14T07:52:57.376}; ClOrdID(11)=15578202177500000010; TradingParty={NoPartyIDs(453)=[{PartyID(448)=Party_identifier; PartyIDSource(447)=PROPRIETARY_CUSTOM_CODE(D); PartyRole(452)=CLIENT_ID(3)}]}; Account(1)=DemoUser; SecurityID(48)=Instrument; SecurityIDSource(22)=EXCHANGE_SYMBOL(8); OrdType(40)=LIMIT(2); TimeInForce(59)=DAY(0); Side(54)=BUY(1); OrderQty(38)=1000; DisplayQty(1138)=1000; Price(44)=105; AccountType(581)=HOUSE(3); OrderCapacity(528)=AGENCY(A); TransactTime(60)=2019-05-14T07:52:57.357; trailer={CheckSum(10)=139};
2019-05-14 07:52:57.405	ExecutionReport	Server	default@Client	header={BeginString(8)=FIXT.1.1; BodyLength(9)=287; MsgType(35)=EXECUTION_REPORT(8); SenderCompID(49)=Server; TargetCompID(56)=Client; MsgSeqNum(34)=1797; SendingTime(52)=2019-05-14T07:52:57.400}; ExecID(17)=1557820377410; ClOrdID(11)=15578202177500000010; OrderID(37)=1557820377411; ExecType(150)=NEW(0); OrdStatus(39)=NEW(0); LeavesQty(151)=1000; CumQty(14)=0; SecurityID(48)=Instrument; SecurityIDSource(22)=EXCHANGE_SYMBOL(8); TradingParty={NoPartyIDs(453)=[{PartyID(448)=Party_identifier; PartyIDSource(447)=PROPRIETARY_CUSTOM_CODE(D); PartyRole(452)=CLIENT_ID(3)}]}; Account(1)=DemoUser; OrdType(40)=LIMIT(2); TimeInForce(59)=DAY(0); Side(54)=BUY(1); OrderQty(38)=1000; DisplayQty(1138)=1000; Price(44)=105; AccountType(581)=HOUSE(3); OrderCapacity(528)=AGENCY(A); TransactTime(60)=2019-05-14T07:52:57.399; trailer={CheckSum(10)=088};
				header={BeginString(8)=FIXT.1.1; BodyLength(9)=299; MsgType(35)=EXECUTION_REPORT(8); SenderCompID(49)=Server; TargetCompID(56)=Client; MsgSeqNum(34)=1799; SendingTime(52)=2019-05-

The Messages tab allows the user to interact with the “Messages” table in the Sailfish database. The table is used for storing messages from all services. The working area of the page is divided into the following sections:

- The “Wizard” and “Query” fields are used to create HQL (Hibernate Query Language) queries (more information regarding HQL is available here: http://docs.jboss.org/hibernate/orm/3.3/reference/en/html/queryhql.html);
- The “Limit” field is used to set the user’s query limit;
- The “Columns” field (when clicked) provides a drop-down list with fields available for showing in UI;



- The ‘Show/Hide raw messages’ button to the left of each message opens a raw message in Hex and a human-readable format.


Query: human_msg LIKE('%155713162337400000018%' Limit: 100

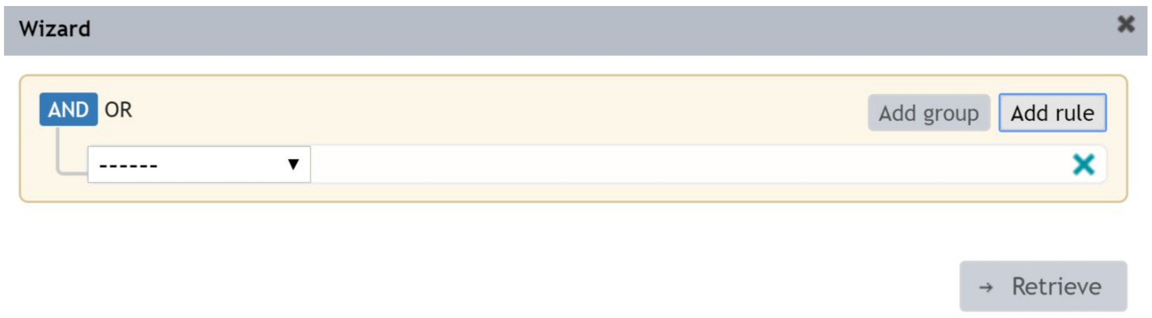
	Timestamp	Name	From	To	Content
	2019-05-08 00:48:10.012	ExecutionReport	Server	default@Client	header={BeginString(8)=FIXT.1.1; BodyLength(9)=299; MsgType(35)=EXECUTION_REPORT(8); SenderCompID(49)=Server; TargetCompID(56)=Client; MsgSeqNum(34)=2892; SendingTime(52)=2019-05-08T00:48:10.015}; ExecID(17)=ClOrdID; ClOrdID(11)=155713162337400000018; OrderID(37)=1557276370100; ExecType(150)=TRADE(F); OrdStatus(39)=FILLED(2); LeavesQty(151)=0; CumQty(14)=1000; SecurityID(48)=Instrument; SecurityIDSource(22)=EXCHANGE_SYMBOL(8); TradingParty={NoPartyIDs(453)=[{PartyID(448)=Party_identifier; PartyIDSource(447)=PROPRIETARY_CUSTOM_CODE(D); PartyRole(452)=CLIENT_ID(3)}]}; Account(1)=DemoUser; OrdType(40)=LIMIT(2); TimeInForce(59)=DAY(0); Side(54)=BUY(1); OrderQty(38)=1000; DisplayQty(1138)=1000; Price(44)=105; AccountType(581)=HOUSE(3); OrderCapacity(528)=AGENCY(A); TransactTime(60)=2019-05-08T00:48:10.015; TradeMatchID(880)=1557276490119; trailer={CheckSum(10)=148};

Raw message

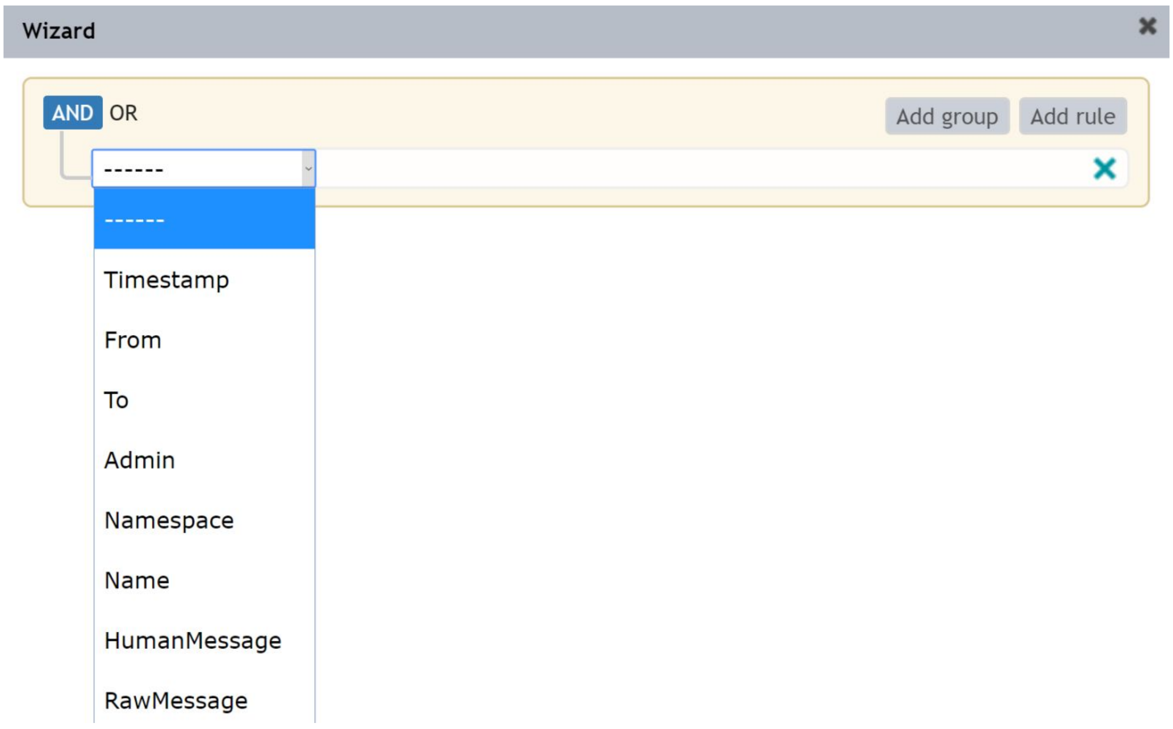
```
00000000: 383d 4649 5854 2e31 2e31 0139 3d32 3939  8=FIXT.1.1.9=299
00000010: 0133 353d 3801 3334 3d32 3839 3201 3439  .35=8.34=2892.49
00000020: 3d53 6572 7665 7201 3532 3d32 3031 3930  =Server.52=20190
00000030: 3530 382d 3030 3a34 383a 3130 2e30 3135  508-00:48:10.015
00000040: 0135 363d 436c 6965 6e74 0131 3d44 656d  .56=Client.1=Dem
00000050: 6f55 7365 7201 3131 3d31 3535 3731 3331  oUser.11=1557131
00000060: 3632 3333 3734 3030 3030 3031 3801 3134  6233740000018.14
00000070: 3d31 3030 3001 3137 3d43 6c4f 7264 4944  =1000.17=ClOrdID
```

In order to create HQL queries using “Wizard”, the user should follow the steps below.

1. Click on the “Wizard” icon. The “Wizard” window opens:



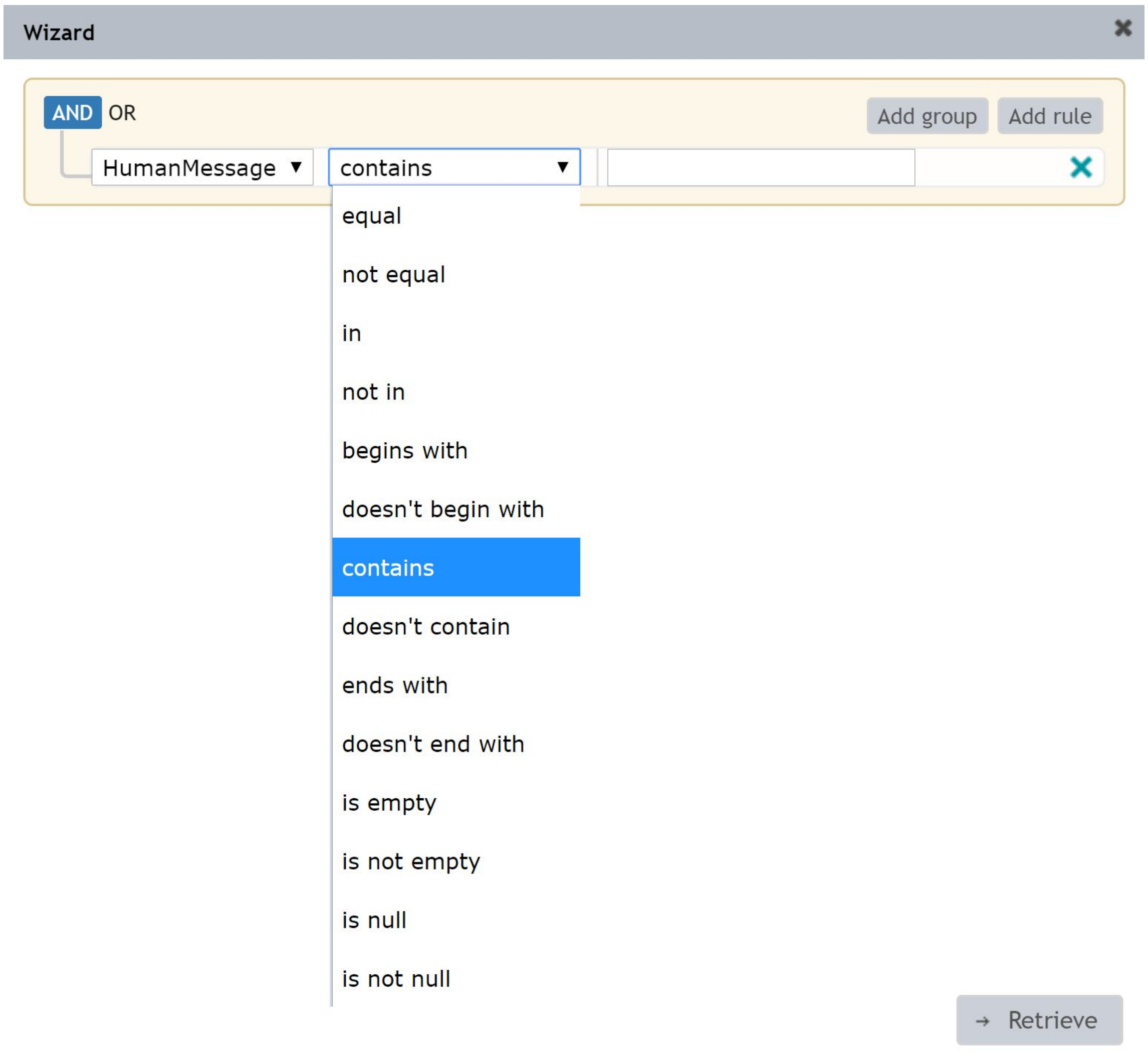
2. Pick one of the two options available: “Add rule” and “Add Group”. “Add rule” allows adding expressions which are executed side by side. “Add Group” allows adding expressions which are executed in a pre-assigned order. Also, two operations can be used: “OR” and “AND”. To create a query, select the required options Add rule or/and Add Group in the upper right corner of the window and operations OR/AND in the upper left corner of the window (a single operation for all rules and a particular operation for every group).
3. Select a field in the table from the first drop-down list for every rule and group:



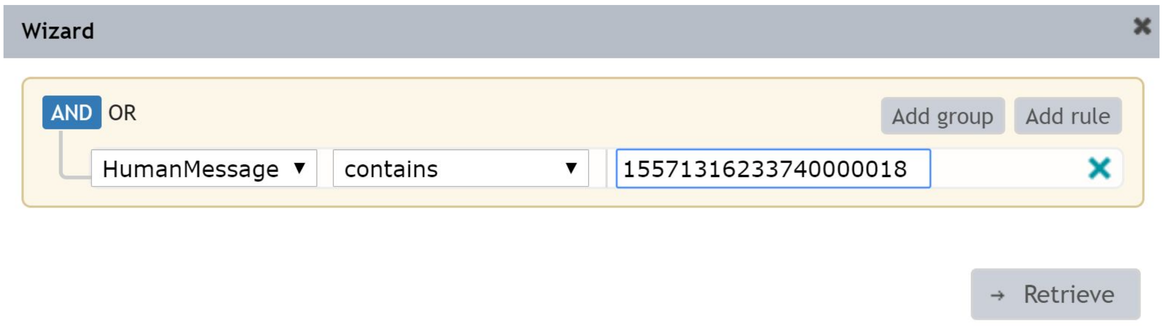
Where:

- Timestamp – the time when a message arrived;
- From/To – the service/gateway the message was sent to or received from;
- Admin – an administrative message;
- Namespace - used for technical needs;
- Name – the name of the message;
- HumanMessage – the contents of the message;
- RawMessage – a raw message in hex;

4. Select operators from the second drop-down list for every rule and group:



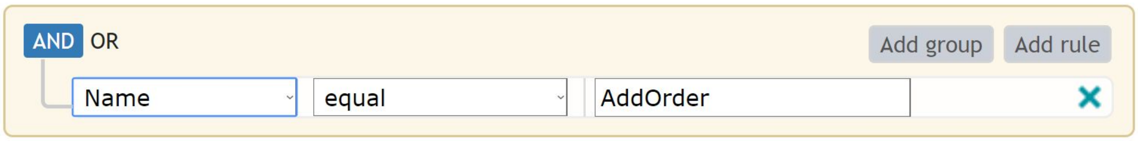
5. Enter a value (if required) into the last field for every rule and group;



HumanMessage ▼ | contains ▼ | 15571316233740000018

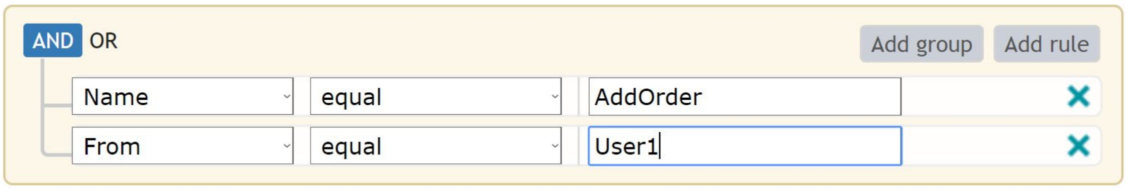
6. Press the “Retrieve” button. The created query gets included into the “Query” field, and the messages in the table are sorted accordingly.

Example: It’s required to filter messages by Add Order (**NOTE!:** the message name should be written without spaces as a dictionary or script). The following rule can be used:



Name | equal | AddOrder

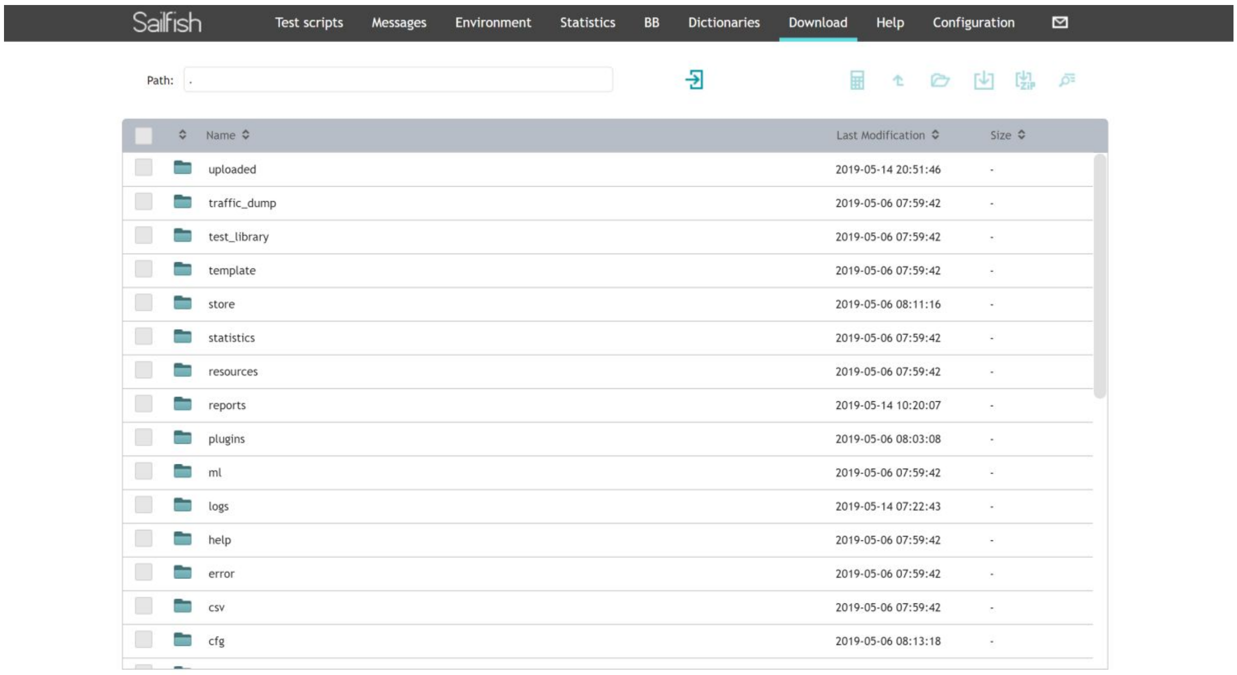
Then, it's required to filter AddOrder messages from the User1 service. To achieve that, add a new rule with the following rule using operation AND:



Name	equal	AddOrder
From	equal	User1

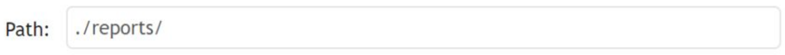
After "Retrieve" is pressed, the messages are filtered by the following query: msg_name = 'AddOrder' AND from_service = 'User1'.

5.4 Download Page

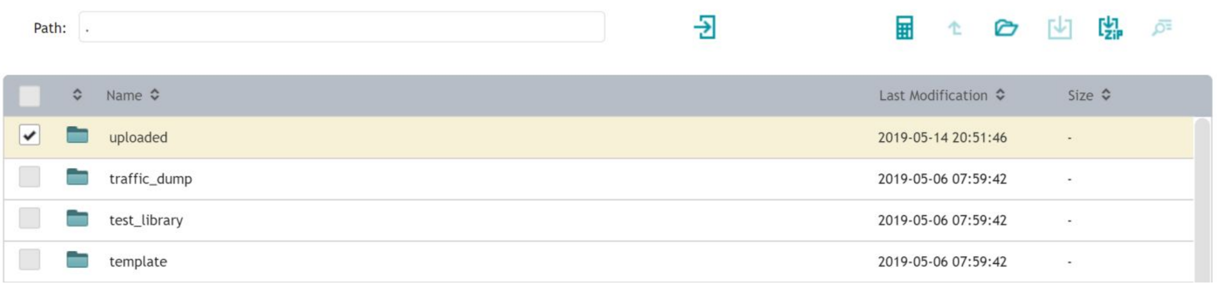


Name	Last Modification	Size
uploaded	2019-05-14 20:51:46	-
traffic_dump	2019-05-06 07:59:42	-
test_library	2019-05-06 07:59:42	-
template	2019-05-06 07:59:42	-
store	2019-05-06 08:11:16	-
statistics	2019-05-06 07:59:42	-
resources	2019-05-06 07:59:42	-
reports	2019-05-14 10:20:07	-
plugins	2019-05-06 08:03:08	-
ml	2019-05-06 07:59:42	-
logs	2019-05-14 07:22:43	-
help	2019-05-06 07:59:42	-
error	2019-05-06 07:59:42	-
csv	2019-05-06 07:59:42	-
cfg	2019-05-06 08:13:18	-

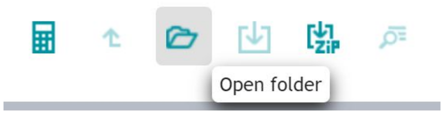
All folders that exist on the local disk under <Deployed Tomcat>/webapps/sfgui/ are listed here and can be accessed from the Sailfish web GUI. The path to the folder can be specified in the "Path" field.

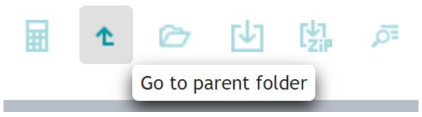

Path: ./reports/

Once the folder is selected, the top panel gets activated.



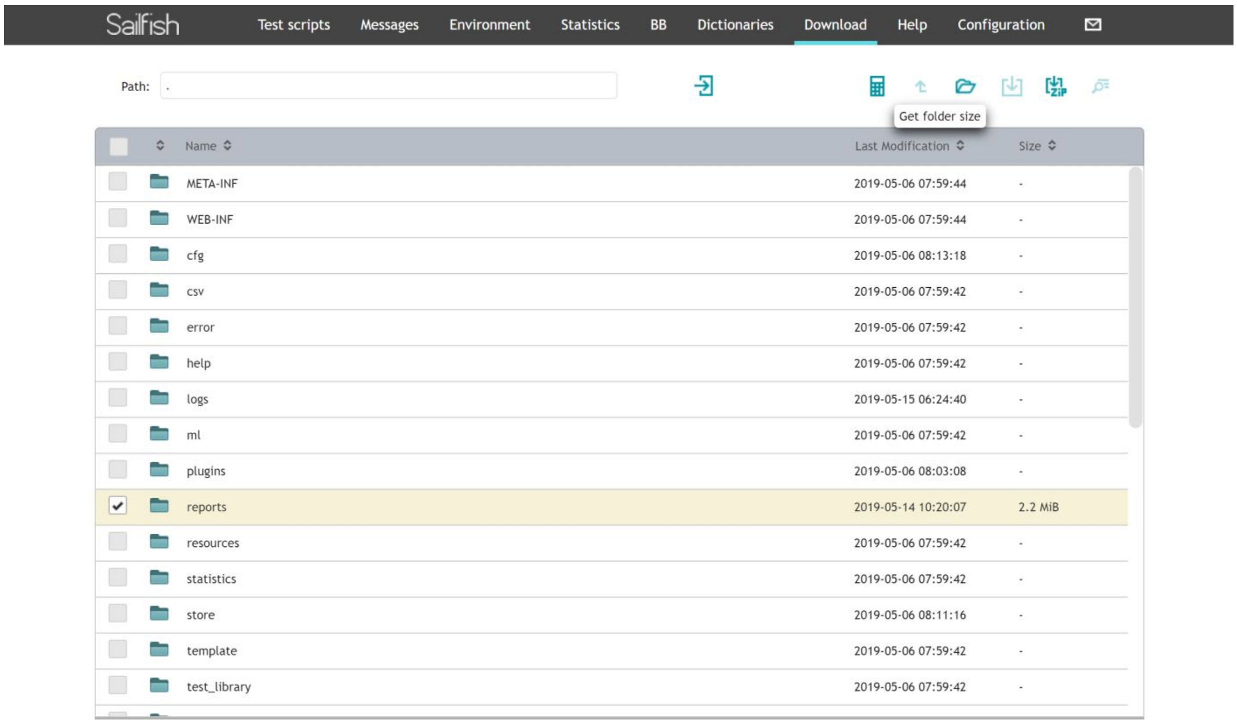
	Name	Last Modification	Size
✓	uploaded	2019-05-14 20:51:46	-
	traffic_dump	2019-05-06 07:59:42	-
	test_library	2019-05-06 07:59:42	-
	template	2019-05-06 07:59:42	-

You can navigate through folders by pressing 

to open the folder and pressing 

to go to the parent folder.

It is also possible to get the folder size by selecting the folder and pressing "Get folder size". The size will appear on the right side of the selected folder in the "Size" column.



	Name	Last Modification	Size
	META-INF	2019-05-06 07:59:44	-
	WEB-INF	2019-05-06 07:59:44	-
	cfg	2019-05-06 08:13:18	-
	csv	2019-05-06 07:59:42	-
	error	2019-05-06 07:59:42	-
	help	2019-05-06 07:59:42	-
	logs	2019-05-15 06:24:40	-
	ml	2019-05-06 07:59:42	-
	plugins	2019-05-06 08:03:08	-
✓	reports	2019-05-14 10:20:07	2.2 MiB
	resources	2019-05-06 07:59:42	-
	statistics	2019-05-06 07:59:42	-
	store	2019-05-06 08:11:16	-
	template	2019-05-06 07:59:42	-
	test_library	2019-05-06 07:59:42	-

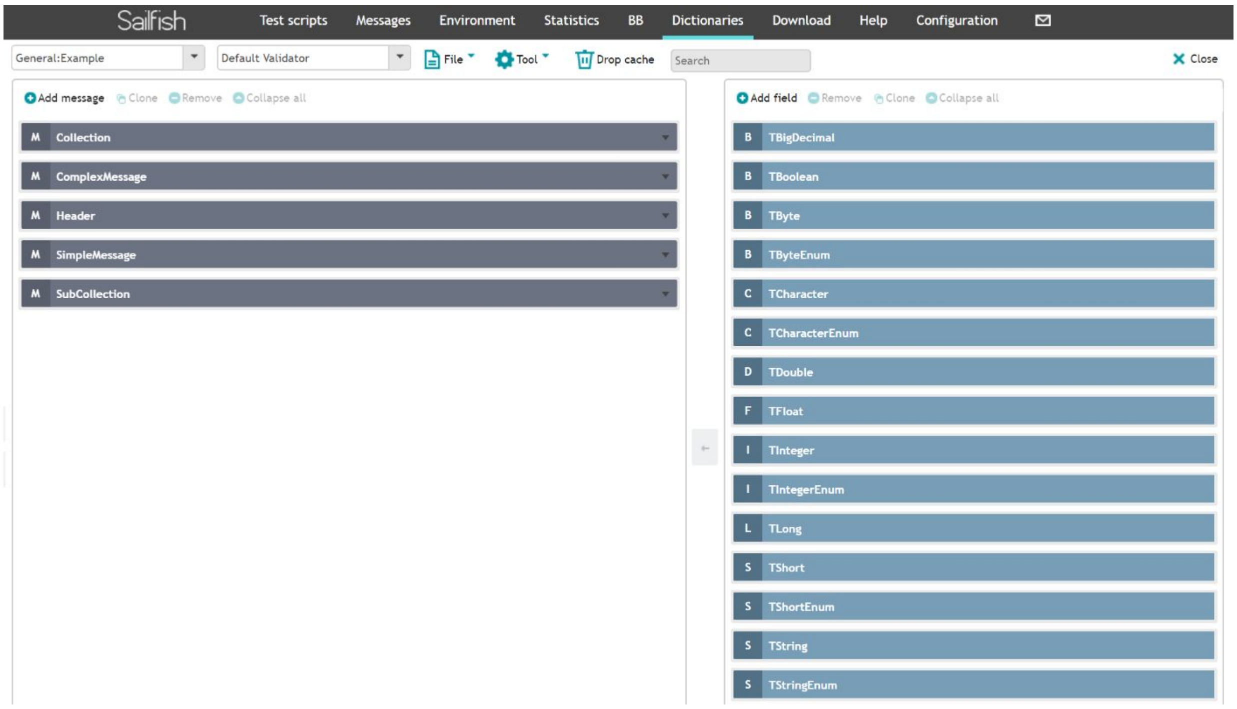
You can download the selected folder as .ZIP by pressing or download selected file by pressing . To show file content, press .

Note: Logs of Sailfish services can be found in '\logs\services\<environment name> path'.

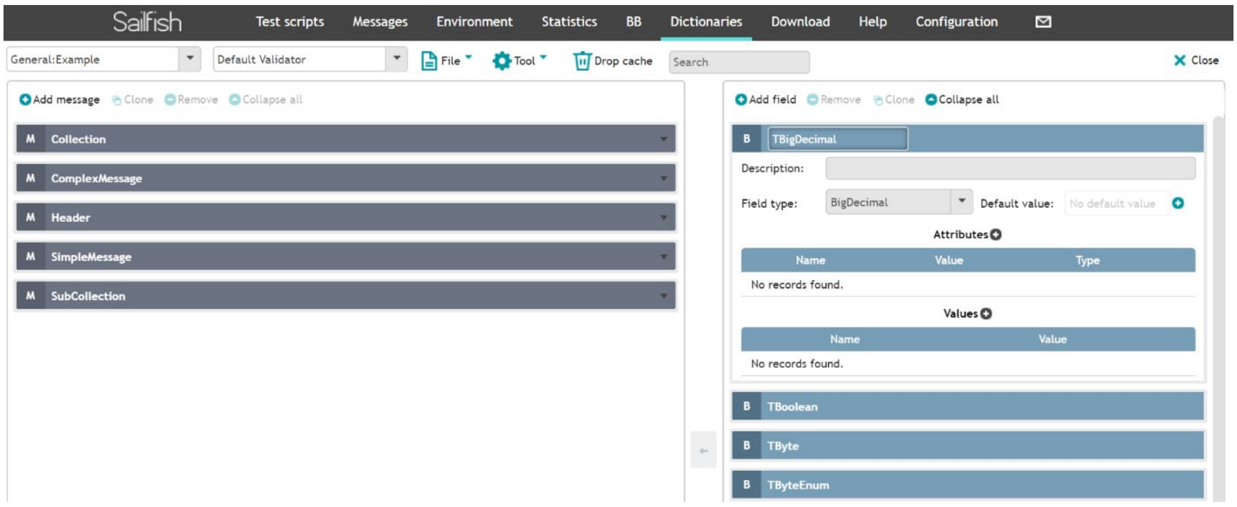
5.5 Dictionaries Page

The Sailfish 'Dictionaries' web page allows updating and changing the interface dictionaries for Sailfish services via the web GUI if there are changes in the interfaces (due to a new release or a bug fix).

To load an existing dictionary, click on the dropdown menu in the top left corner and select the required dictionary. The right part of the page is dedicated to the fields, while the left part is dedicated to the messages.



The user can search for a certain value, update it and add it to any message, e.g., select any field, on the right part of the page. Expand the attributes and values of the field by clicking on the left button.

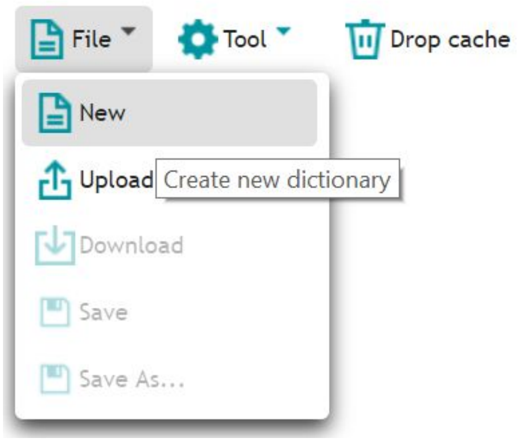


Then, the user can update, add or delete them. After that, you can select the message on the left part of the page, where the updated or the added field should be included.

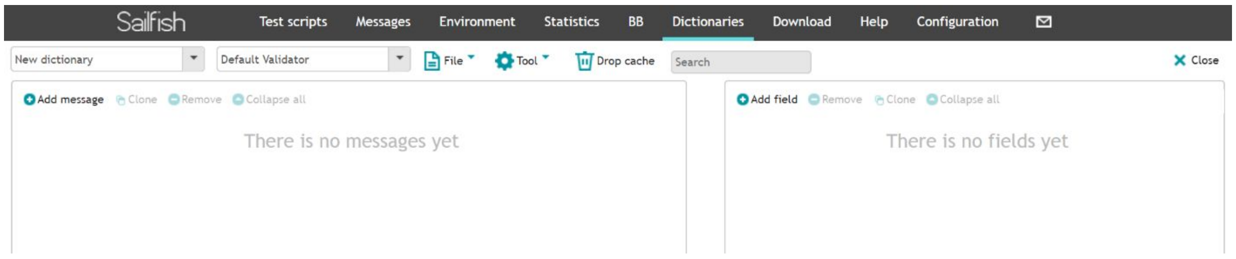
The user can update messages as well. A new dictionary can be created using one of the two ways described below.

5.5.1 Dictionary Creation Using Sailfish GUI

1. Close the dictionary if any existing one is already loaded by clicking on ✕ Close in the top right corner.
2. Click on the “New” button.



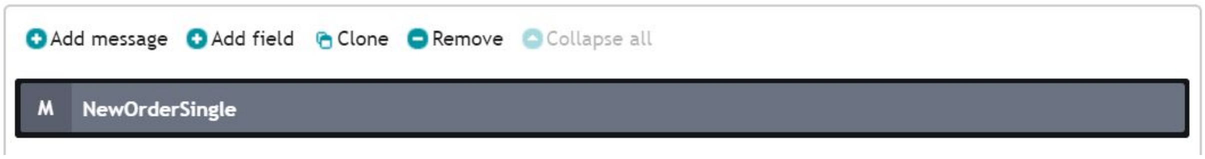
Empty fields and messages areas are added as follows:



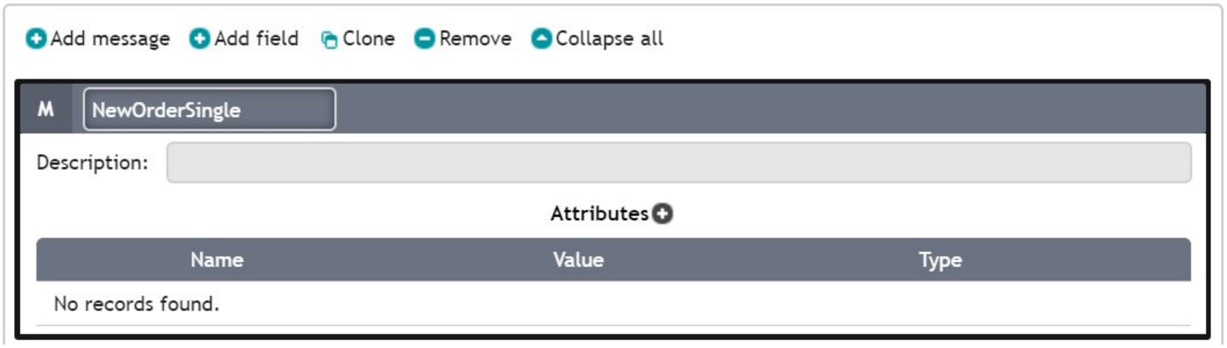
3. Add messages in the left part of the page as per the interface specification documents.

   a) To add a message, click on “Add message”. In the window that pops up, specify the name of the message (**NOTE!:** the name should be written without spaces and it should be unique, e.g. NewOrderSingle) and its description and press “OK”. The message is added to the messages and can be seen in the left part of the page.

Example:

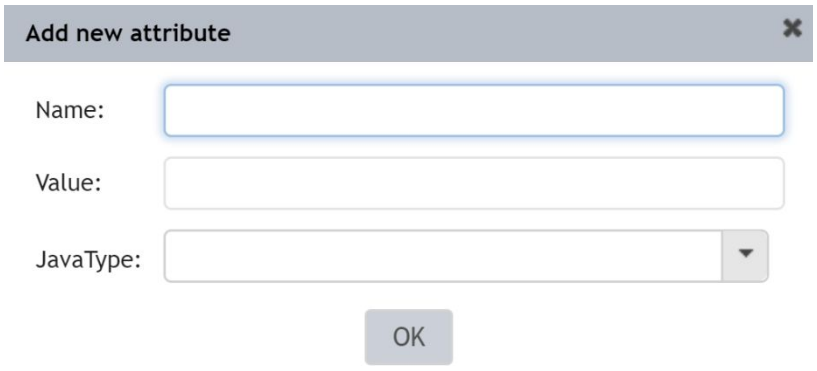


   b) Expand the message by clicking on the left button and add attributes to the message (if required) by clicking on ⊕ near the Attributes:



Name	Value	Type
No records found.		

   c) In the window that pops up, all fields are mandatory:



Where

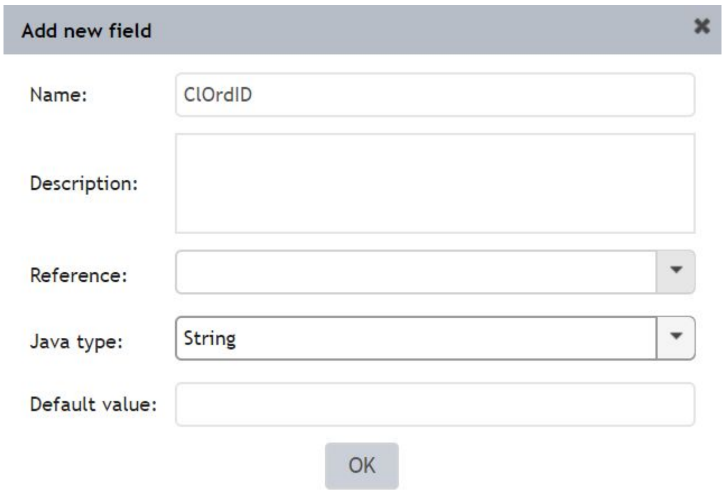
- Name (**NOTE!:** the name should be written without spaces and it must be unique among the names in the field);
- Value should be applicable to the specified JavaType of the attribute;
- JavaType - data type which can be kept in the field;

4. Add the fields to the message based on the interface specification documents.

There are 2 ways to add a field to a message:

1) If a field has a limited quantity of values, it must be displayed on the right part of the page and it's better to include it there if the same field should be added to several messages. Then, add them to all the required messages. To achieve that:

   a) click on "Add field" in the right part of the page to add a new field.



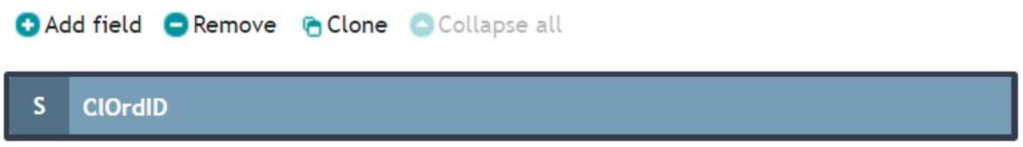
Name: ClOrdID
Java type: String
Default value:

The following fields should be populated in the window that pops up:

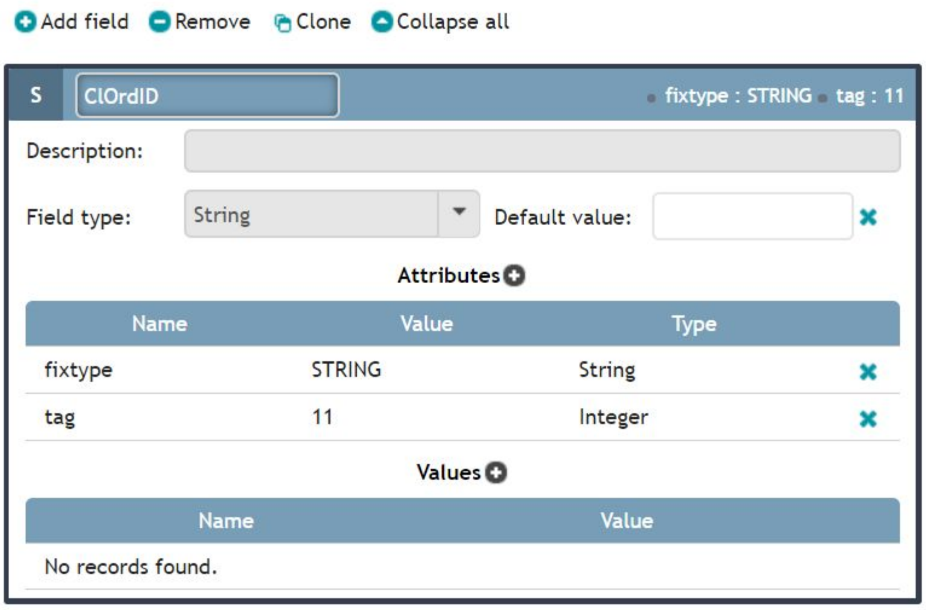
- Name – this is a mandatory field (**NOTE!:** the name should be written without spaces and it must be unique among the fields in the right part of the page);
- Description – this is an optional field;
- JavaType – this is a mandatory field; this is a data type which can be kept in the field;
- Default value – this is an optional field; it will be used if a value in the message being sent is absent. If a field does not have a default value, any value applicable for the field type can be specified. Otherwise, only one value from the ones specified for the field can be used.

b) After populating the fields, press “OK”. The field is added to the fields in the right part of the page.

Example:



c) Expand the field by clicking on the left button and add attributes and values to the field (if required) by clicking on near the Attributes and near the Values.



Description:

Field type: String   Default value:

Attributes

Name	Value	Type
fixtype	STRING	String
tag	11	Integer

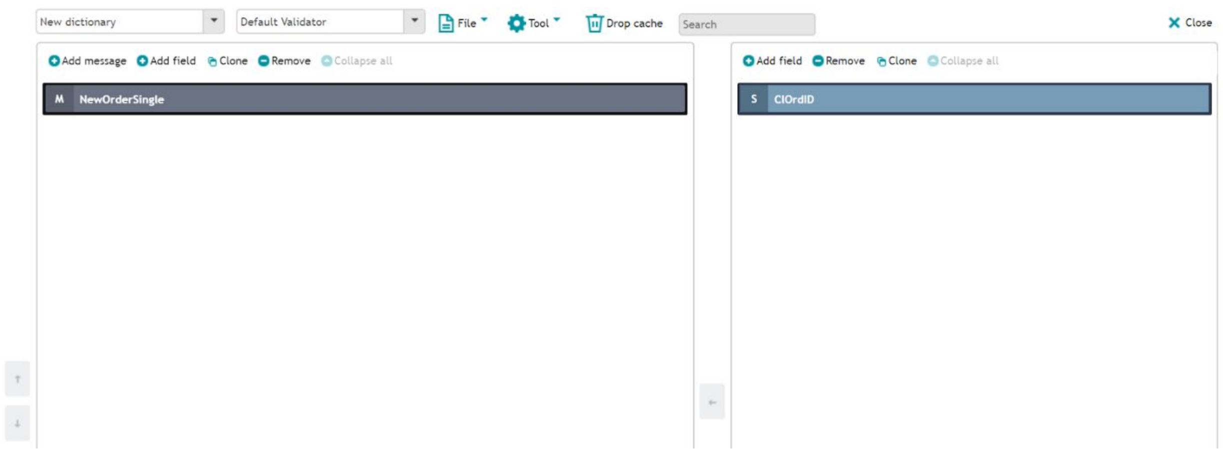
Values

Name	Value
No records found.	

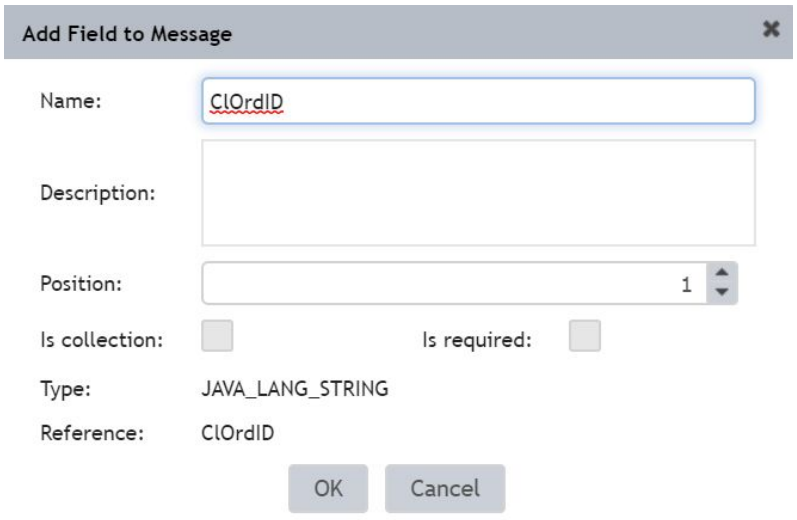
In the window that pops up, all the fields are mandatory and should be specified as described above.

d) Include all the required fields listed in the left part of the page into the messages.

e) Select a field from the right part of the page and a message from the left part of the page, where it’s required to add the field, and press :



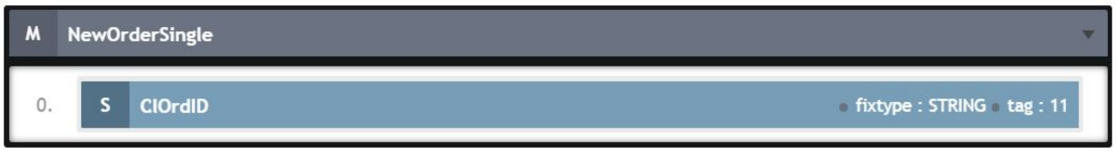
f) Populate the fields in the window that pops up (if required) and press “OK”.



Add Field to Message

Name:	ClOrdID
Description:	
Position:	1
Is collection:	☐
Is required:	☐
Type:	JAVA_LANG_STRING
Reference:	ClOrdID

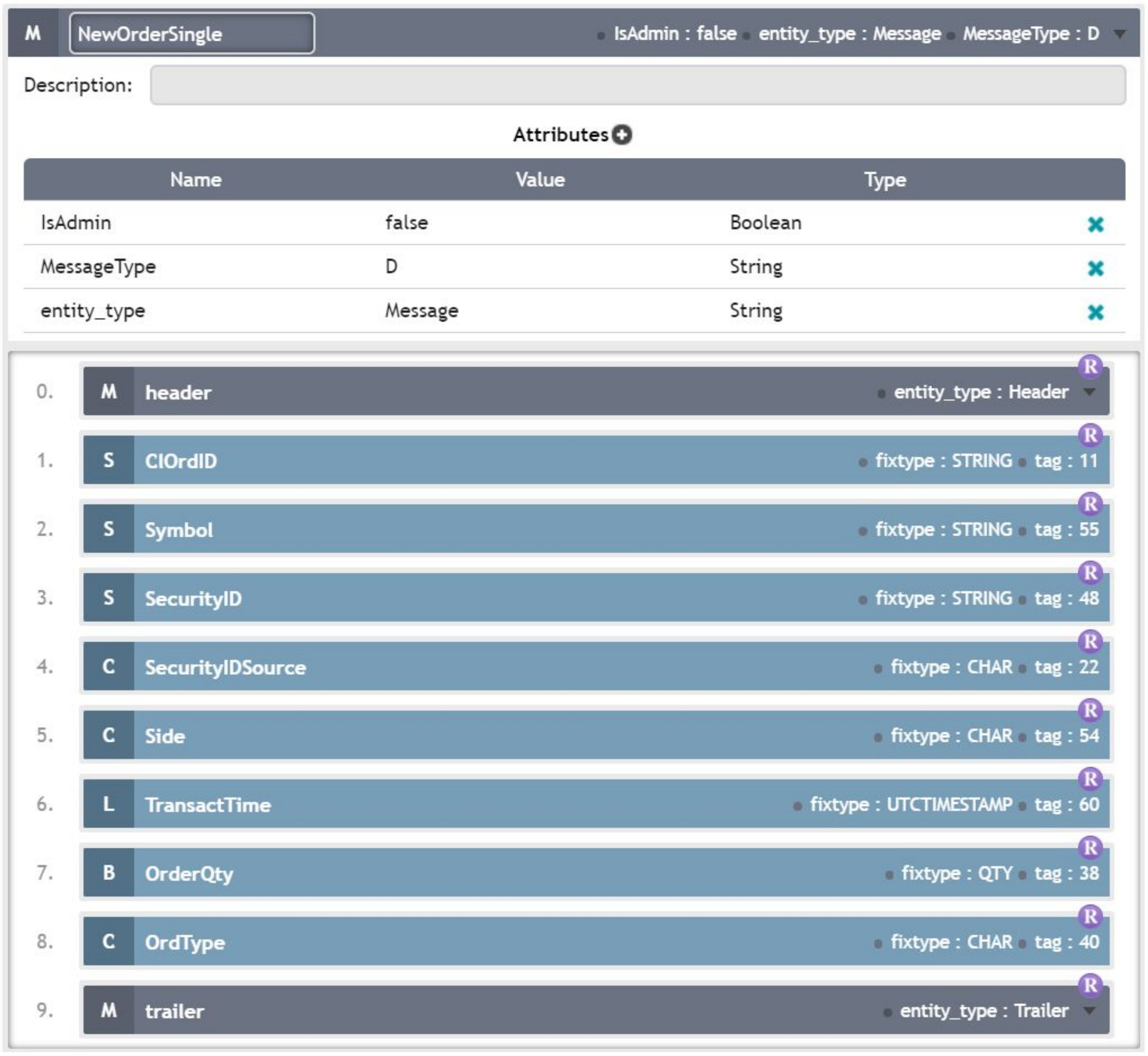
The field gets added to the message as a reference field:



M NewOrderSingle

0. S ClOrdID • fixtype : STRING • tag : 11

g) Expand the field by clicking on the left button and add attributes (if required) by clicking on ⊕ near the Attributes (**NOTE!:** all fields are mandatory in the pop-up window, as described above).



M NewOrderSingle · IsAdmin : false · entity_type : Message · MessageType : D

Description:

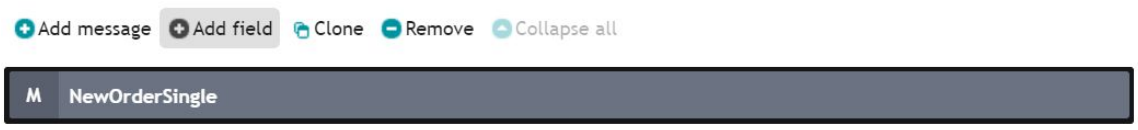
Attributes

Name	Value	Type
IsAdmin	false	Boolean
MessageType	D	String
entity_type	Message	String

0. M header · entity_type : Header
1. S ClOrdID · fixtype : STRING · tag : 11
2. S Symbol · fixtype : STRING · tag : 55
3. S SecurityID · fixtype : STRING · tag : 48
4. C SecurityIDSource · fixtype : CHAR · tag : 22
5. C Side · fixtype : CHAR · tag : 54
6. L TransactTime · fixtype : UTCTIMESTAMP · tag : 60
7. B OrderQty · fixtype : QTY · tag : 38
8. C OrdType · fixtype : CHAR · tag : 40
9. M trailer · entity_type : Trailer

2) The second way is adding a field straight into a message. To achieve that, perform the following actions:

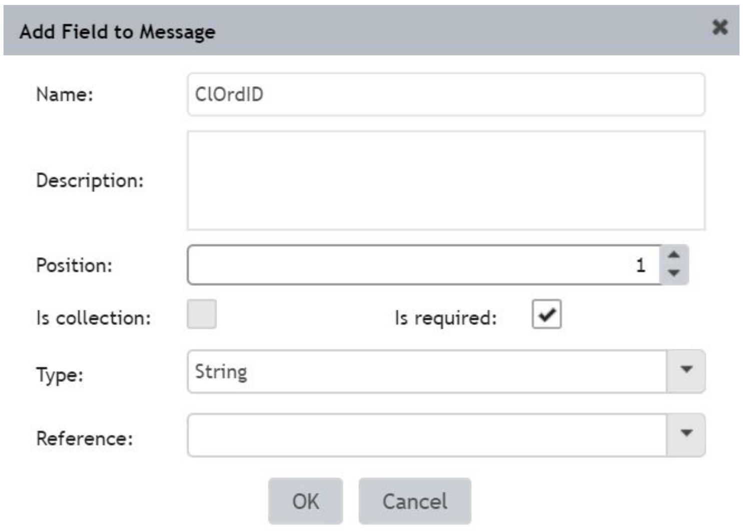
a) Select the message and click on “Add field”.



b) The following fields should be populated in the window that appears:

- Name – this is a mandatory field (**NOTE!:** the name should be written without spaces and it must be unique among the fields in the right part of the page);
- Description – this is an optional field;
- “Is required” is ticked when it's a mandatory field of the message;
- “Is collection” is ticked if a field can contain several values;
- Type – this is a mandatory field if Reference field is not present; this is a data type which can be kept in the field;

- Reference – this is an optional field, a prototype of the field from the right part of the page, from which JavaType, Attributes and Values are inherited.



Name:	ClOrdID
Description:	
Position:	1
Is collection:	☐
Is required:	✓
Type:	String
Reference:	

c) Then, add all required attributes to the field as described above.

NOTE!:

- If an existing message should be included as a field of a message, select Type = Message and Reference = Message name which should be added;
- Fields should be added according to the specification. To change a field position, select the field and use left-hand arrows of the Messages area. The numbering starts from 0.

5. After creating all required messages and filling them up with required fields, save the new dictionary by clicking on “Save as”, fill in all the fields in the dialog window and click on “Save”.

NOTE!: Don’t close the dictionary while adding messages and fields, since changes won’t be saved in this case. Alternatively, save your changes regularly.

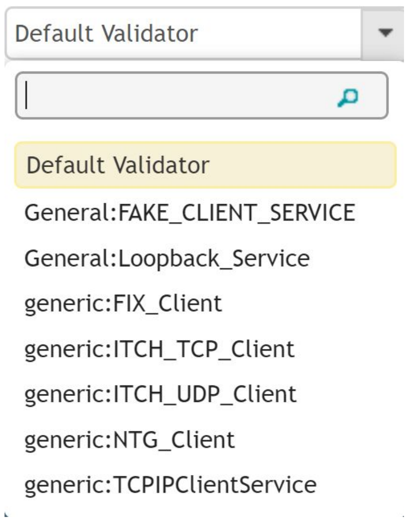
NOTE!: The name of the dictionary should be unique.

The new dictionary will be available right after it has been created. If a user is updating the existing one, it is enough to clean the cache by pressing Drop cache in the same browser tab where the dictionary editor is opened to ensure the dictionary works properly.

5.5.2 Dictionary validation

To validate the dictionary to make sure it matches the service that the dictionary is used for, do the following:

1. Choose a validator which matches the service the dictionary is used for:



2. After selecting a validator, a dictionary will be checked with the validator right away. A pop-up message appears, displaying errors, if there are any.

3. All elements with errors will be highlighted in red. A message with the explanation will be shown if you hover over an element.

5.5.3 Dictionary update

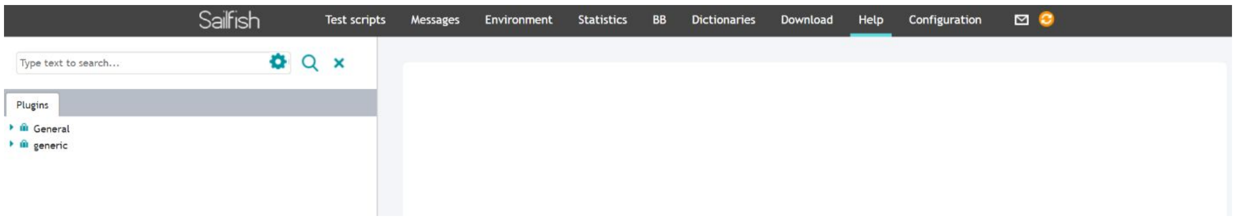
To update a dictionary, do the following:

1. Move to the “Dictionary” tab and select the required dictionary under the “Select dictionary” dropdown menu.
2. Change any messages and their fields in the right and/or left parts of the page.
    a) To change attributes/values and other parameters of the message/field, do the following:
    - Click on the left button;
    - Click on any field and overwrite the value;
    - Click on “Enter”;
    - To delete attributes and other parameters of the message/field, click on [x].

    b) To change message fields, do the following:
    - Expand the message by clicking on the right button;
    - Change attributes and other parameters as described in the previous point;
    - To delete attributes and other parameters, click on [x];

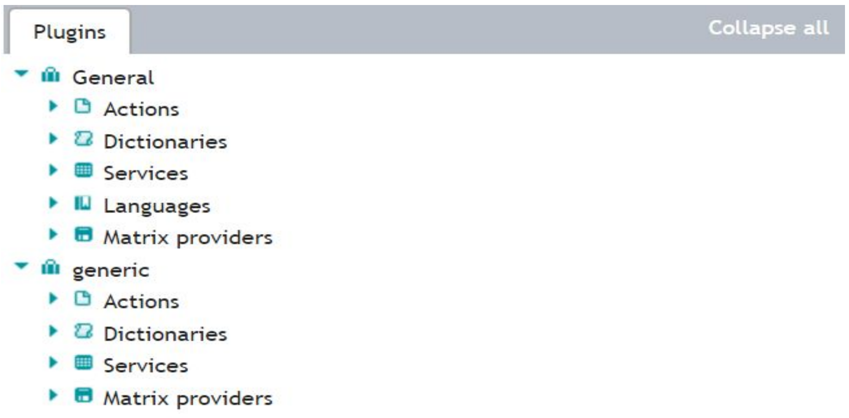
    c) To remove a message/field, select it and click on “Remove”.
    d) To add a new message/field, click on “Add message/field” and refer to the relevant actions described in the dictionary creation section (steps 3-4 of Method 1).
3. When the above is done, click on the “Save” button.
4. Clean the cache on the browser tab where the dictionary editor is opened to ensure the dictionary works properly.

5.6 Help Page

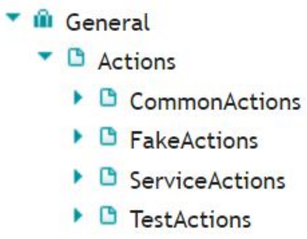


The search form with searching options and plugins installed is displayed in the left part of the window. Each plugin contains the following units:

- Actions – the available actions with their description;
- Dictionaries – the available dictionaries with their structure description;
- Services – a list of services available in the plugins, organized by name;
- Languages – the supported matrix languages;
- Matrix providers – the implemented methods of loading matrices to Sailfish.

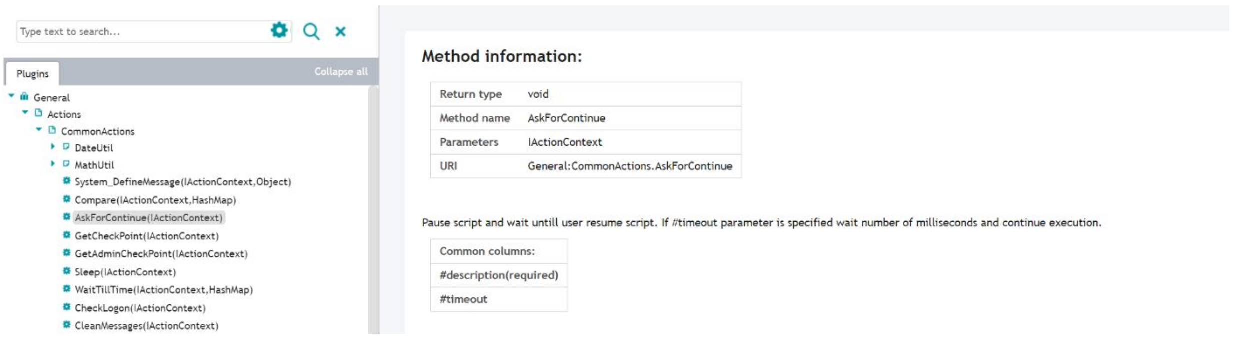


You can find help on the Actions, the group types of which are listed below:



- Common Actions – a set of basic actions for performing simple tasks;
- Fake Actions – actions that emulate receiving fake messages; these actions are used in automated testing;
- Service Actions – actions dedicated to operations with the service;
- Test Actions – actions dedicated to interaction of the Sailfish service with a remote system.

By pressing each of the Actions, you will see the method information and other details in the right part of the window.


Type text to search...

Plugins — Collapse all

- General
  - Actions
    - CommonActions
      - DateUtil
      - MathUtil
      - System_DefineMessage(IActionContext,Object)
      - Compare(IActionContext,HashMap)
      - AskForContinue(IActionContext)
      - GetCheckPoint(IActionContext)
      - GetAdminCheckPoint(IActionContext)
      - Sleep(IActionContext)
      - WaitTillTime(IActionContext,HashMap)
      - CheckLogon(IActionContext)
      - CleanMessages(IActionContext)

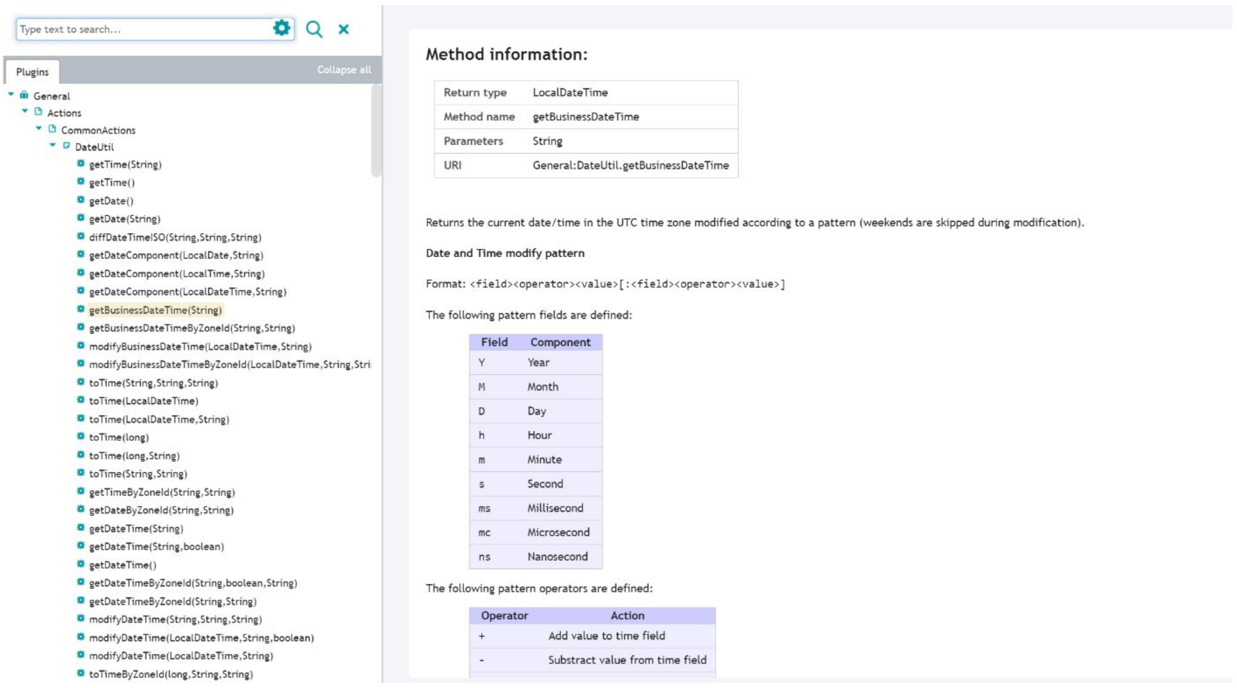
Method information:

Return type	void
Method name	AskForContinue
Parameters	IActionContext
URI	General:CommonActions.AskForContinue

Pause script and wait untill user resume script. If #timeout parameter is specified wait number of milliseconds and continue execution.

Common columns:
#description(required)
#timeout

On the Sailfish Help page you can also find tips on how to use Date, Math and other Utils.


Type text to search...

Plugins — Collapse all

- General
  - Actions
    - CommonActions
      - DateUtil
        - getTime(String)
        - getTime()
        - getDate()
        - getDate(String)
        - diffDateTimeISO(String,String,String)
        - getDateComponent(LocalDate,String)
        - getDateComponent(LocalTime,String)
        - getDateComponent(LocalDateTime,String)
        - getBusinessDateTime(String)
        - getBusinessDateTimeByZoneId(String,String)
        - modifyBusinessDateTime(LocalDateTime,String)
        - modifyBusinessDateTimeByZoneId(LocalDateTime,String,Stri
        - toTime(String,String,String)
        - toTime(LocalDateTime)
        - toTime(LocalDateTime,String)
        - toTime(long)
        - toTime(long,String)
        - toTime(String,String)
        - getTimeByZoneId(String,String)
        - getDateByZoneId(String,String)
        - getDateTime(String)
        - getDateTime(String,boolean)
        - getDateTime()
        - getDateTimeByZoneId(String,boolean,String)
        - getDateTimeByZoneId(String,String)
        - modifyDateTime(String,String,String)
        - modifyDateTime(LocalDateTime,String,boolean)
        - modifyDateTime(LocalDateTime,String)
        - toTimeByZoneId(long,String,String)

Method information:

Return type	LocalDateTime
Method name	getBusinessDateTime
Parameters	String
URI	General:DateUtil.getBusinessDateTime

Returns the current date/time in the UTC time zone modified according to a pattern (weekends are skipped during modification).

Date and Time modify pattern

Format: `<field><operator><value>[:<field><operator><value>]`

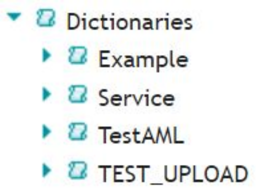
The following pattern fields are defined:

Field	Component
Y	Year
M	Month
D	Day
h	Hour
m	Minute
s	Second
ms	Millisecond
mc	Microsecond
ns	Nanosecond

The following pattern operators are defined:

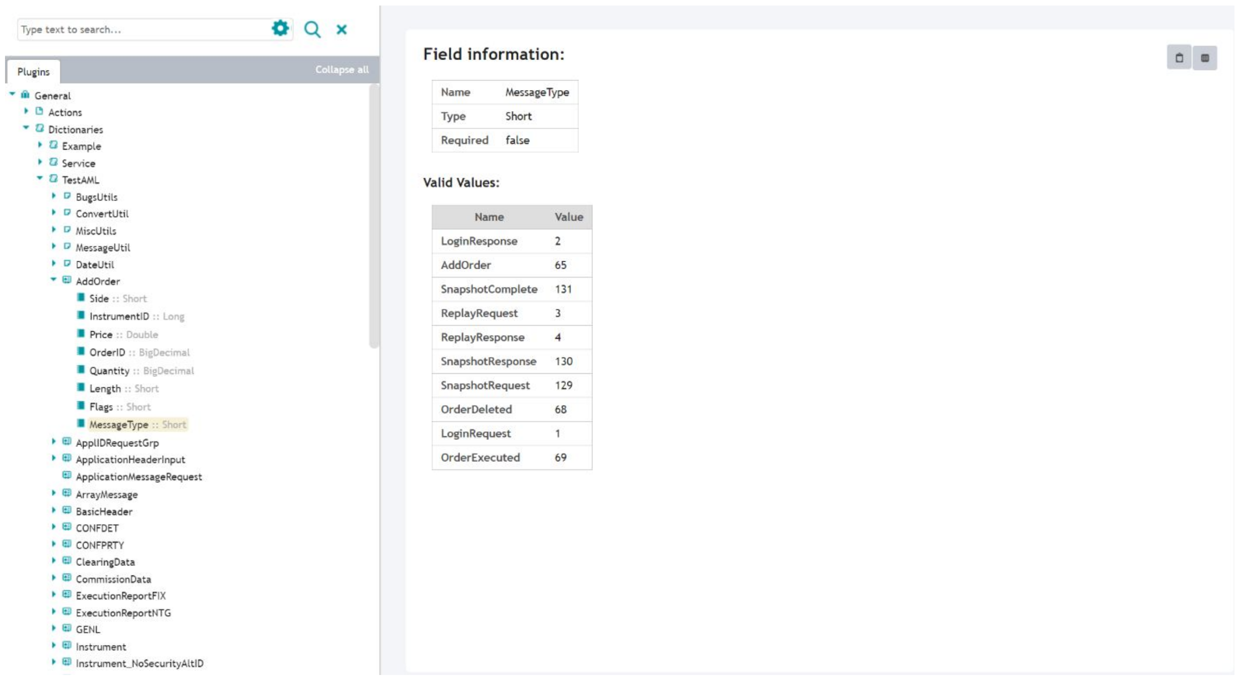
Operator	Action
+	Add value to time field
-	Substract value from time field

The implemented dictionaries are listed as follows:


- Dictionaries
  - Example
  - Service
  - TestAML
  - TEST_UPLOAD

By selecting the dictionary and the message, it is possible to see Help on each of the fields.

Example of Help on a MessageType Field:



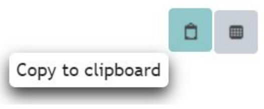
- General
  - Actions
  - Dictionaries
    - Example
    - Service
    - TestAML
      - BugsUtils
      - ConvertUtil
      - MiscUtils
      - MessageUtil
      - DateUtil
      - AddOrder
        - Side :: Short
        - InstrumentID :: Long
        - Price :: Double
        - OrderID :: BigDecimal
        - Quantity :: BigDecimal
        - Length :: Short
        - Flags :: Short
        - MessageType :: Short
      - ApplIDRequestGrp
      - ApplicationHeaderInput
      - ApplicationMessageRequest
      - ArrayMessage
      - BasicHeader
      - CONFDET
      - CONFPRTY
      - ClearingData
      - CommissionData
      - ExecutionReportFIX
      - ExecutionReportNTG
      - GENL
      - Instrument
      - Instrument_NoSecurityAltID

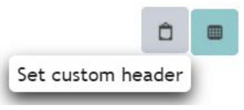
Field information:

Name	MessageType
Type	Short
Required	false

Valid Values:

Name	Value
LoginResponse	2
AddOrder	65
SnapshotComplete	131
ReplayRequest	3
ReplayResponse	4
SnapshotResponse	130
SnapshotRequest	129
OrderDeleted	68
LoginRequest	1
OrderExecuted	69

Clicking on a particular field gives the option to Copy to clipboard by clicking on 

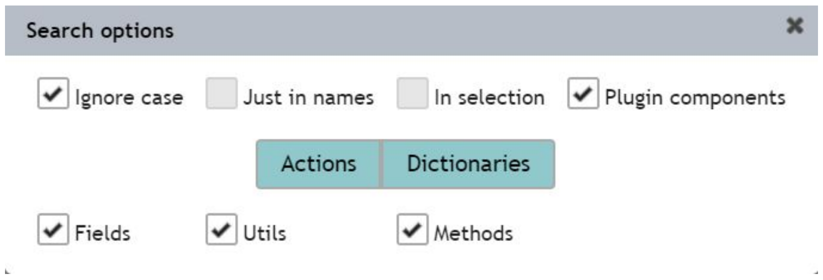
and the option to Set custom header by clicking on 

in the top right part of the window.

Alternatively, you can type the name of the action/message in the

field in the top left part of the window to get a help window on it.

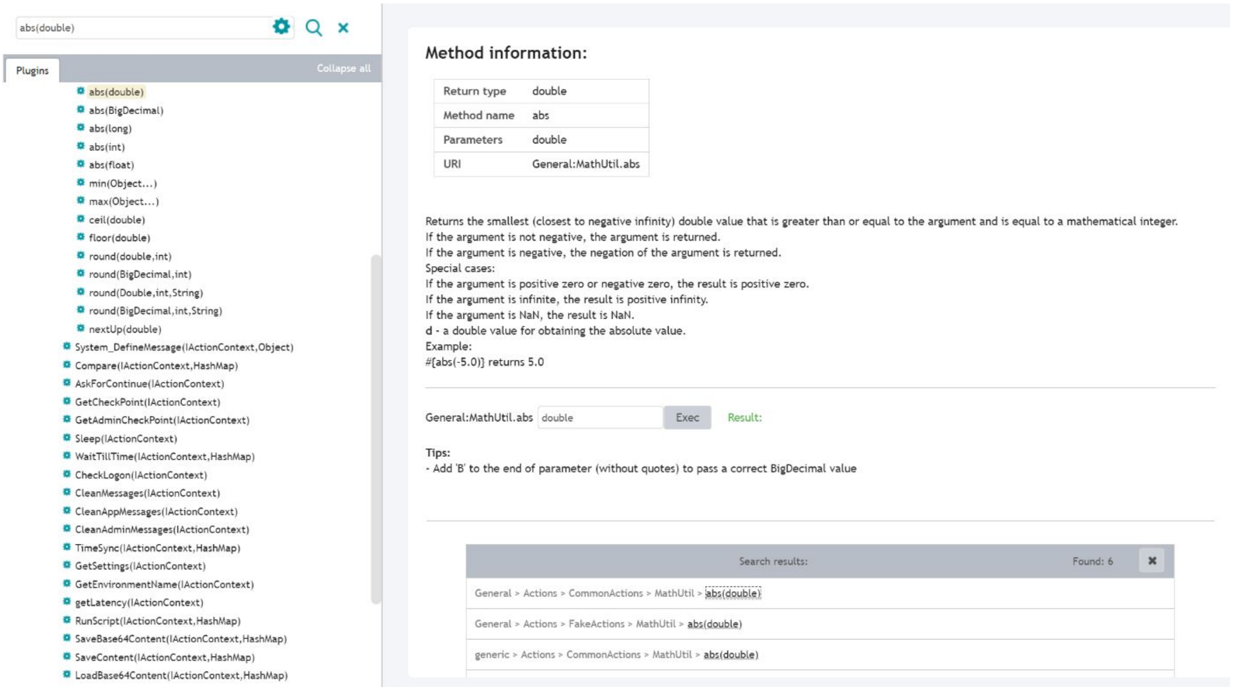
By clicking on you can choose the search parameters.



[✓ Ignore case] Just in names In selection [✓ Plugin components]

[✓ Fields] [✓ Utils] [✓ Methods]

Example:


abs(double)

```
abs(double)
abs(BigDecimal)
abs(long)
abs(int)
abs(float)
min(Object...)
max(Object...)
ceil(double)
floor(double)
round(double,int)
round(BigDecimal,int)
round(Double,int,String)
round(BigDecimal,int,String)
nextUp(double)
System_DefineMessage(IActionContext,Object)
Compare(IActionContext,HashMap)
AskForContinue(IActionContext)
GetCheckPoint(IActionContext)
GetAdminCheckPoint(IActionContext)
Sleep(IActionContext)
WaitTillTime(IActionContext,HashMap)
CheckLogon(IActionContext)
CleanMessages(IActionContext)
CleanAppMessages(IActionContext)
CleanAdminMessages(IActionContext)
TimeSync(IActionContext,HashMap)
GetSettings(IActionContext)
GetEnvironmentName(IActionContext)
getLatency(IActionContext)
RunScript(IActionContext,HashMap)
SaveBase64Content(IActionContext,HashMap)
SaveContent(IActionContext,HashMap)
LoadBase64Content(IActionContext,HashMap)
```

Method information:

Return type	double
Method name	abs
Parameters	double
URI	General:MathUtil.abs

Returns the smallest (closest to negative infinity) double value that is greater than or equal to the argument and is equal to a mathematical integer.
If the argument is not negative, the argument is returned.
If the argument is negative, the negation of the argument is returned.
Special cases:
If the argument is positive zero or negative zero, the result is positive zero.
If the argument is infinite, the result is positive infinity.
If the argument is NaN, the result is NaN.
d - a double value for obtaining the absolute value.
Example:
#{abs(-5.0)} returns 5.0

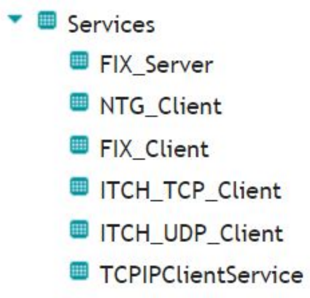
General:MathUtil.abs double Exec Result:

Tips:
- Add 'B' to the end of parameter (without quotes) to pass a correct BigDecimal value

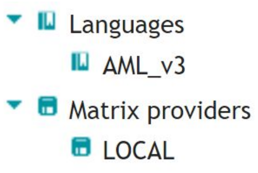
Search results: Found: 6

General > Actions > CommonActions > MathUtil > abs(double)
General > Actions > FakeActions > MathUtil > abs(double)
generic > Actions > CommonActions > MathUtil > abs(double)

The list of implemented Services can be found in the left part of the Plugins window as well. By clicking on each of them, you can see the Service information.



There is also Help on Languages and Matrix providers.



5.7 Configuration Page

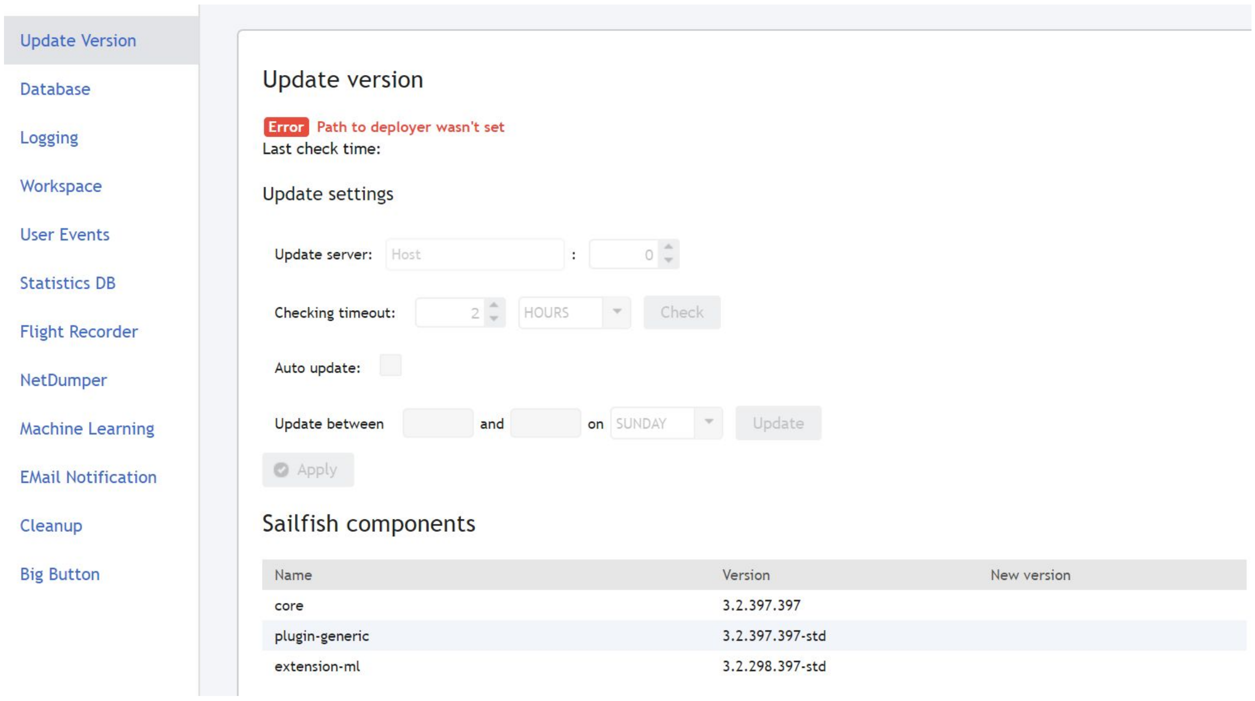
The “Configuration” tab contains the following sections:

- Update Version – contains the update settings and information about Sailfish components;
- Database – contains Hibernate settings or connection to the Sailfish database;
- Logging – to configure logging. The “Clean” button will erase all messages and service events from the database;
- Workspace – to configure workspaces;

- User Events – contains information about user events/messages;
- Statistics DB – to configure the Sailfish statistics database;
- Flight Recorder – to make a profile recording;
- NetDumper – to record pickups for network traffic recording;
- Machine Learning – to get machine learning data and analyze execution reports;
- EMail Notification – to send notifications with execution results to the user's email;
- Cleanup – to perform cleanup of reports, matrices, etc.;
- Big Button – helps to run several matrices on different instances of Sailfish, which are connected, and send execution reports to the main one.

5.7.1 Update Version

Navigating to the Configuration Page, first the Update Version tab with the relevant information will appear. Here, the list of Sailfish components can be observed.



Update version

Error Path to deployer wasn't set

Last check time:

Update settings

Update server: Host : 0

Checking timeout: 2 HOURS

Auto update:

Update between and on SUNDAY

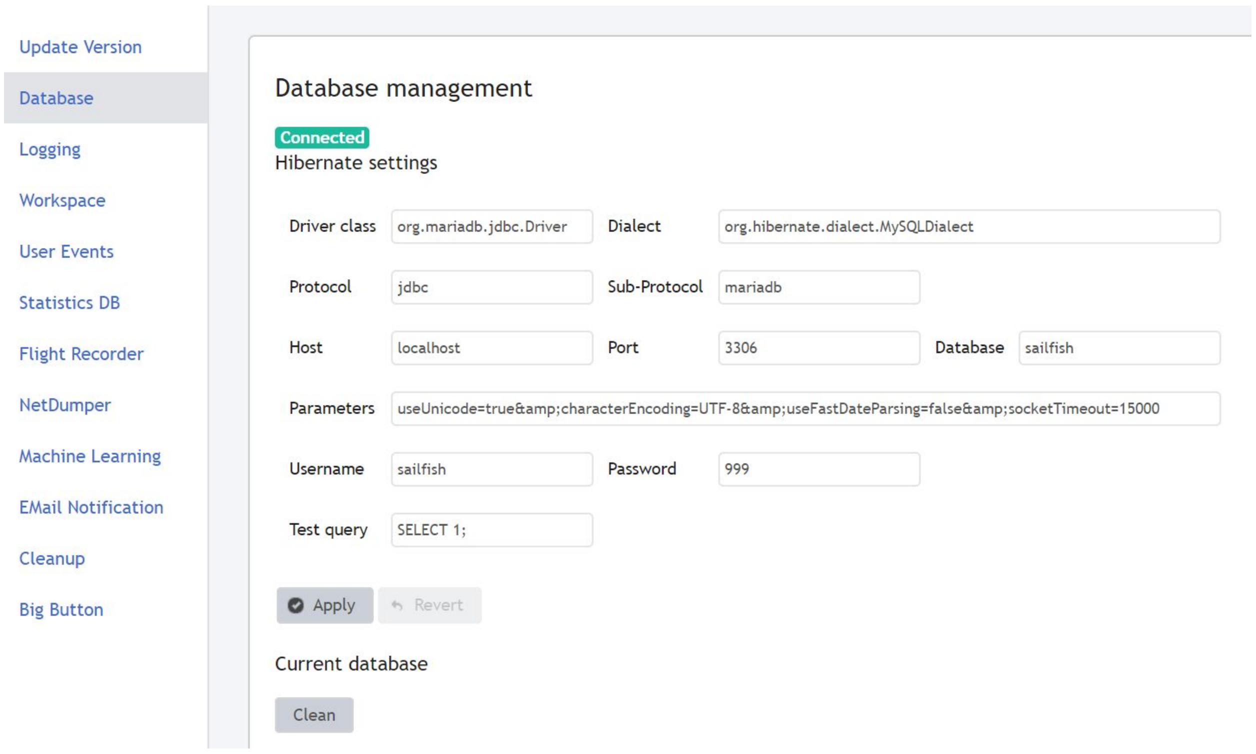
Sailfish components

Name	Version	New version
core	3.2.397.397	
plugin-generic	3.2.397.397-std	
extension-ml	3.2.298.397-std	

5.7.2 Database

In the Database tab you can find the current status of the Database to which Sailfish is connected, manage the hibernate settings and perform the cleanup of the Database removing all messages and service events.

These settings are used for HDD storage of the db type which is configured on Workspace -> Storage Type.



Database management

Connected

Hibernate settings

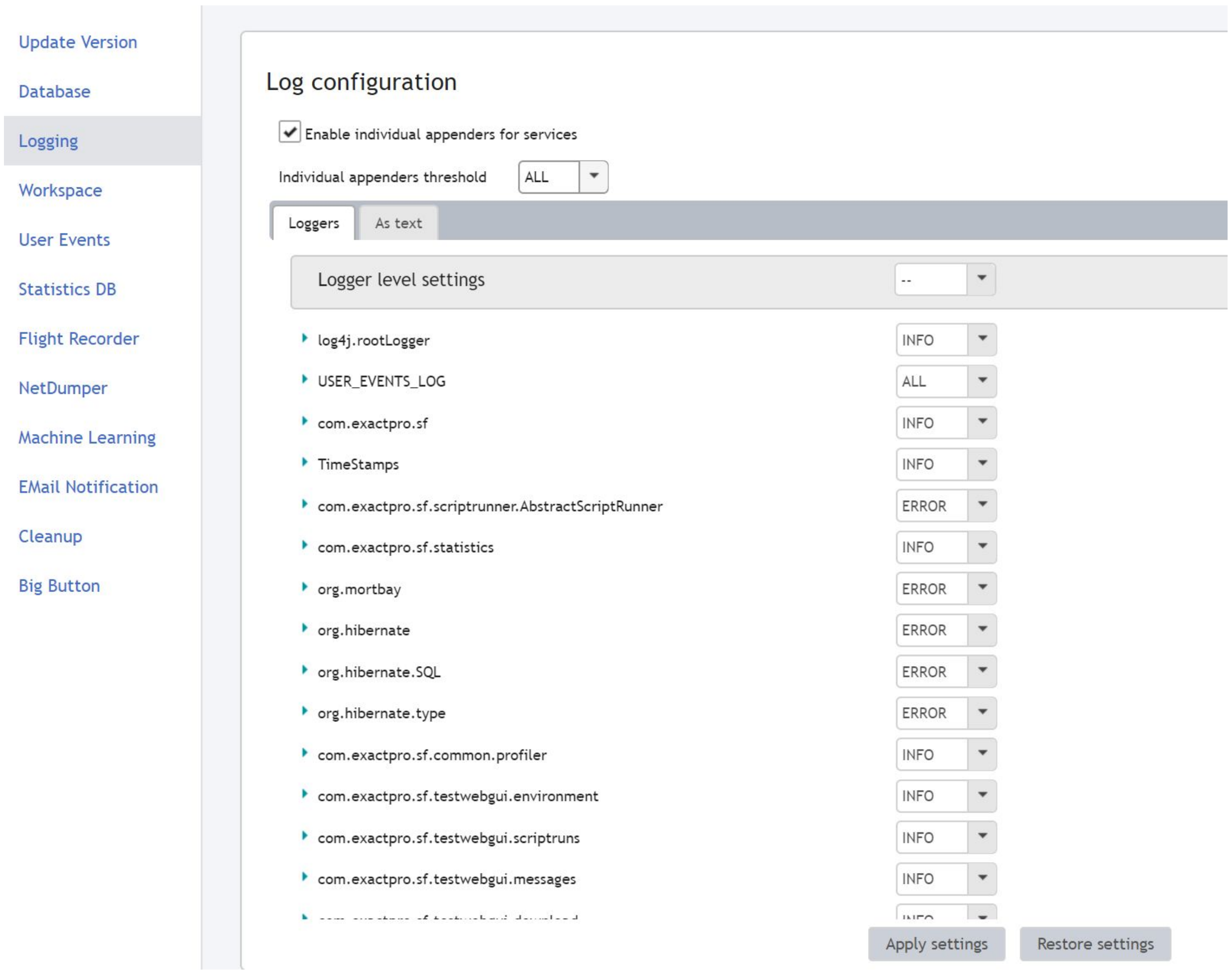
Driver class	org.mariadb.jdbc.Driver	Dialect	org.hibernate.dialect.MySQLDialect		
Protocol	jdbc	Sub-Protocol	mariadb		
Host	localhost	Port	3306	Database	sailfish
Parameters	useUnicode=true&amp;characterEncoding=UTF-8&amp;useFastDateParsing=false&amp;socketTimeout=15000				
Username	sailfish	Password	999		
Test query	SELECT 1;				

Current database

Note: Only one Database can be connected to Sailfish.

5.7.3 Logging

In the Logging tab, the Log configuration can be changed.



Log configuration

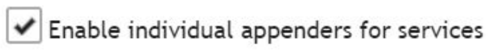
✓ Enable individual appenders for services

Individual appenders threshold: ALL

Logger level settings: --

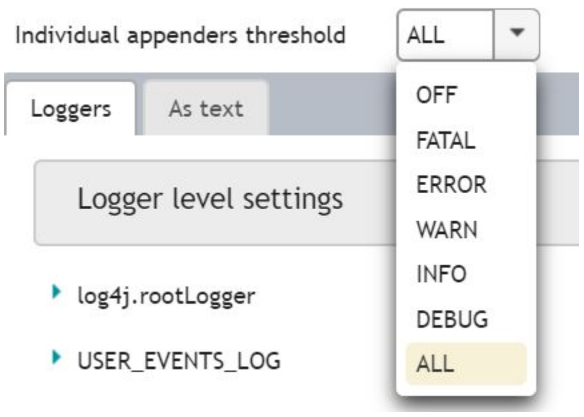
log4j.rootLogger	INFO
USER_EVENTS_LOG	ALL
com.exactpro.sf	INFO
TimeStamps	INFO
com.exactpro.sf.scriptrunner.AbstractScriptRunner	ERROR
com.exactpro.sf.statistics	INFO
org.mortbay	ERROR
org.hibernate	ERROR
org.hibernate.SQL	ERROR
org.hibernate.type	ERROR
com.exactpro.sf.common.profiler	INFO
com.exactpro.sf.testwebgui.environment	INFO
com.exactpro.sf.testwebgui.scriptruns	INFO
com.exactpro.sf.testwebgui.messages	INFO

You can enable individual appenders for services by ticking the corresponding checkbox

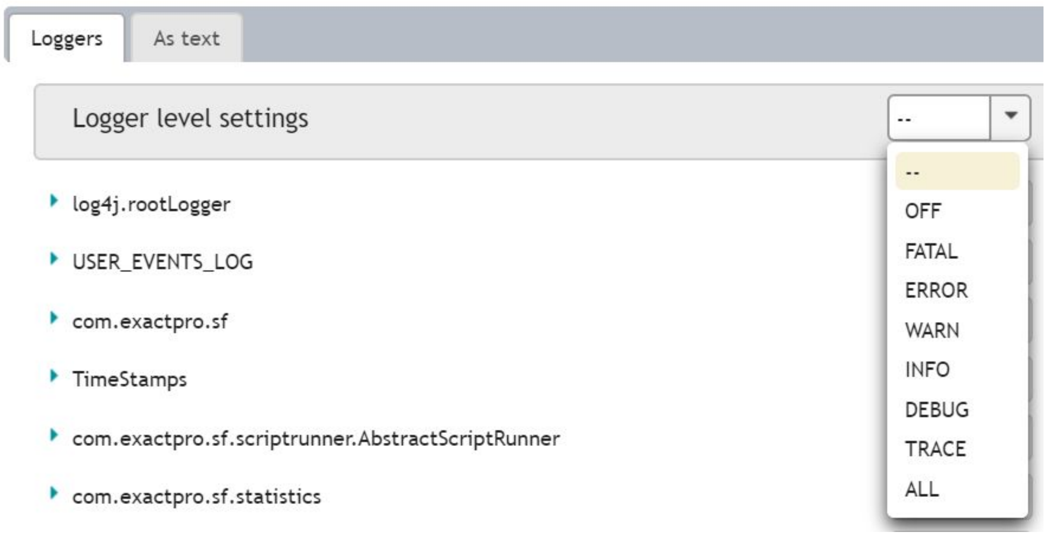


.

In the drop-down menu, it is possible to select the Individual appenders threshold.

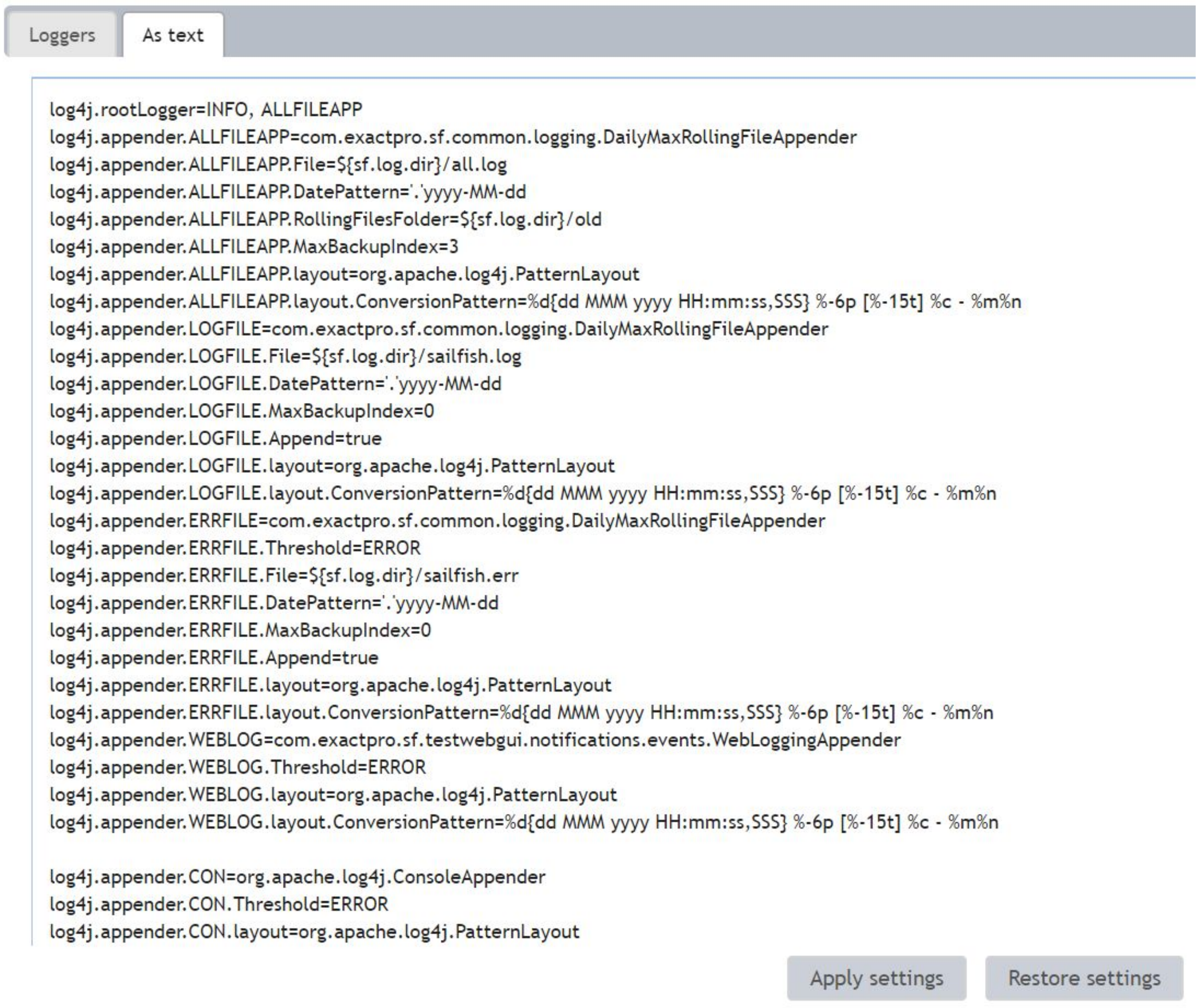


In the Loggers tab, you can either configure the logging level for all loggers at once in the corresponding drop-down menu or change this setting for each of the Logger.



- log4j.rootLogger
- USER_EVENTS_LOG
- com.exactpro.sf
- TimeStamps
- com.exactpro.sf.scriptrunner.AbstractScriptRunner
- com.exactpro.sf.statistics

By navigating to the “As text” tab, you can see the same settings displayed in the text manner.

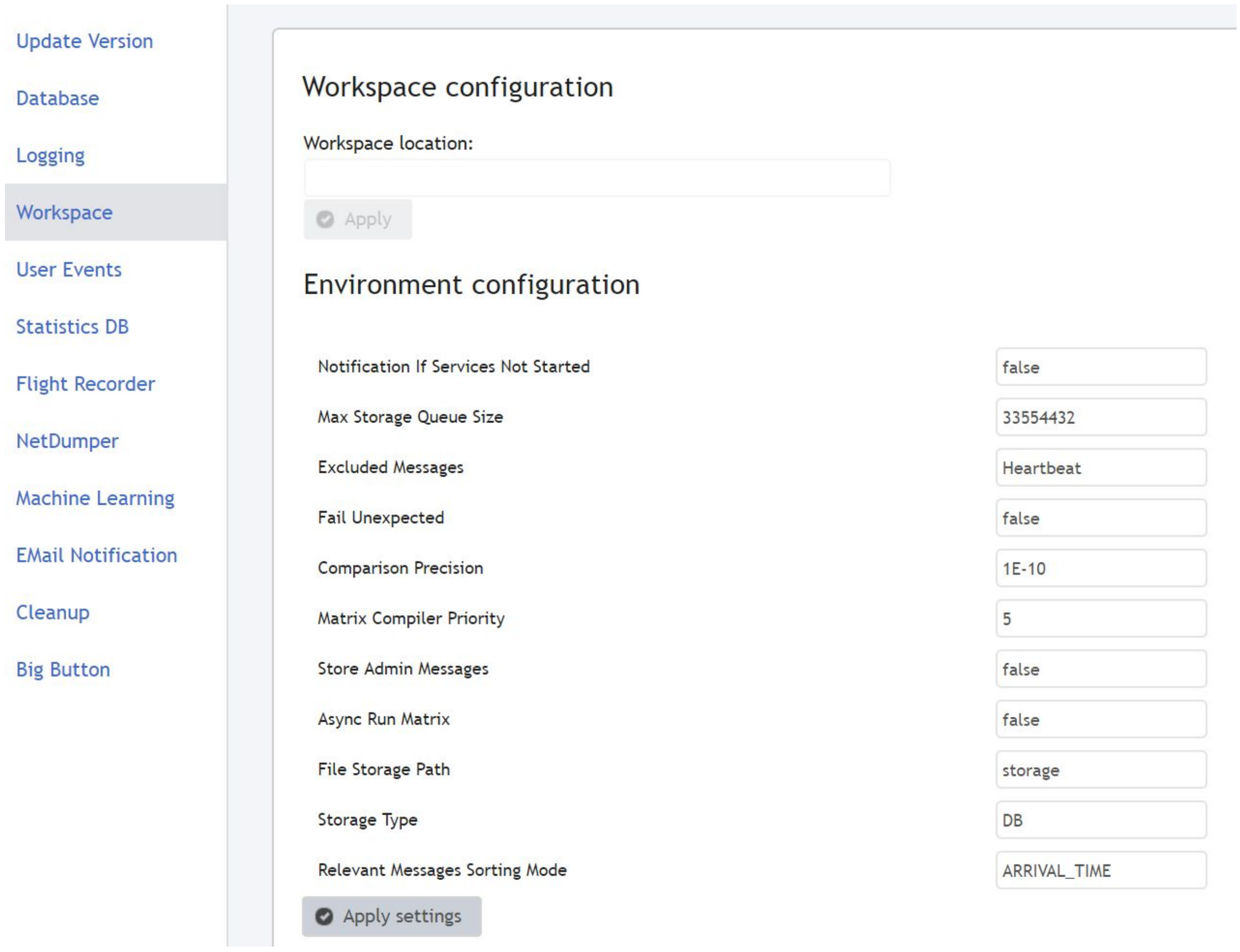


```
log4j.rootLogger=INFO, ALLFILEAPP
log4j.appender.ALLFILEAPP=com.exactpro.sf.common.logging.DailyMaxRollingFileAppender
log4j.appender.ALLFILEAPP.File=${sf.log.dir}/all.log
log4j.appender.ALLFILEAPP.DatePattern='.'yyyy-MM-dd
log4j.appender.ALLFILEAPP.RollingFilesFolder=${sf.log.dir}/old
log4j.appender.ALLFILEAPP.MaxBackupIndex=3
log4j.appender.ALLFILEAPP.layout=org.apache.log4j.PatternLayout
log4j.appender.ALLFILEAPP.layout.ConversionPattern=%d{dd MMM yyyy HH:mm:ss,SSS} %-6p [%-15t] %c - %m%n
log4j.appender.LOGFILE=com.exactpro.sf.common.logging.DailyMaxRollingFileAppender
log4j.appender.LOGFILE.File=${sf.log.dir}/sailfish.log
log4j.appender.LOGFILE.DatePattern='.'yyyy-MM-dd
log4j.appender.LOGFILE.MaxBackupIndex=0
log4j.appender.LOGFILE.Append=true
log4j.appender.LOGFILE.layout=org.apache.log4j.PatternLayout
log4j.appender.LOGFILE.layout.ConversionPattern=%d{dd MMM yyyy HH:mm:ss,SSS} %-6p [%-15t] %c - %m%n
log4j.appender.ERRFILE=com.exactpro.sf.common.logging.DailyMaxRollingFileAppender
log4j.appender.ERRFILE.Threshold=ERROR
log4j.appender.ERRFILE.File=${sf.log.dir}/sailfish.err
log4j.appender.ERRFILE.DatePattern='.'yyyy-MM-dd
log4j.appender.ERRFILE.MaxBackupIndex=0
log4j.appender.ERRFILE.Append=true
log4j.appender.ERRFILE.layout=org.apache.log4j.PatternLayout
log4j.appender.ERRFILE.layout.ConversionPattern=%d{dd MMM yyyy HH:mm:ss,SSS} %-6p [%-15t] %c - %m%n
log4j.appender.WEBLOG=com.exactpro.sf.testwebgui.notifications.events.WebLoggingAppender
log4j.appender.WEBLOG.Threshold=ERROR
log4j.appender.WEBLOG.layout=org.apache.log4j.PatternLayout
log4j.appender.WEBLOG.layout.ConversionPattern=%d{dd MMM yyyy HH:mm:ss,SSS} %-6p [%-15t] %c - %m%n

log4j.appender.CON=org.apache.log4j.ConsoleAppender
log4j.appender.CON.Threshold=ERROR
log4j.appender.CON.layout=org.apache.log4j.PatternLayout
```

You can either Apply or restore the settings after making the changes.

5.7.4 Workspace

Sailfish workspaces allow you to keep the main Sailfish distribution, user configurations and dictionaries separated. This makes it possible to upgrade Sailfish or dictionaries leaving the user files as reports, matrices and database connection settings unchanged. Among single project needs, such customization can vary, and it was decided to introduce several workspaces.
The environment settings which can be changed are displayed below:



Workspace configuration

Workspace location:

Environment configuration

Notification If Services Not Started	false
Max Storage Queue Size	33554432
Excluded Messages	Heartbeat
Fail Unexpected	false
Comparison Precision	1E-10
Matrix Compiler Priority	5
Store Admin Messages	false
Async Run Matrix	false
File Storage Path	storage
Storage Type	DB
Relevant Messages Sorting Mode	ARRIVAL_TIME

- **Notification If Services Not Started** – is used to notify the user, before matrix execution is started, if the services specified in the test script are not started.

This feature can be useful when running the test scripts manually, because it allows to take an action before the execution is started.

If Notification If Services Not Started=true, and at least one of the services specified in the matrix is not started, the execution of the test script will be on hold, which would be observed on the Test scripts page in the Reports, and on the Matrix execution panel the user will see: 'The following services have not been started' (not started services will be listed). The user will be able to move to another step, continue or stop the execution.

- **Max Storage Queue Size** – maximum MQ volume capacity in RAM for saving messages in HDD. If the memory is full, the arrived messages will not be stored. These messages will not be displayed on the Messages page, neither in the 'All Messages' table in the report and will not be found using the retrieve action. MQ overflow can be provoked by the arrival of a large number of messages. The 'Max Storage Queue Size' functionality allows preventing JVM crash with OutOfMemory error.

- **Excluded Messages** – messages which will not be stored in the DB, and the user will not see them in the report or in the Messages page.

The excluded messages must be listed separated by commas and must have the same names as the ones set in the dictionary.

This feature might be useful for messages like Heartbeat – this will allow to decrease the size of the report, thus, saving memory, excluding undescriptive messages.

- **Fail Unexpected** – the setting that allows the user to set up the default value in the #faild_unexpected field, which sets the way of comparing the messages, for example, in the receive action.

Possible values in the #faild_unexpected field:

N or n – only filtered fields are checked;

Y or n – all fields in the message are checked;

A or a – all fields in the message as well as the message structure are checked.

- **Comparison Precision** - allows to set up the default sensibility of numeric values with a floating point. The values are considered equal when their absolute difference does not exceed the set value.

Comparison Precision is considered when Receive actions are executed and can be re-set up using the Precision application function.

- **Matrix Compiler Priority** – is used for regulating the priority between the Matrices compilation and the rest of the actions: start of the services, matrices execution and general Sailfish processes.

Integer values have to be specified: 1 – minimal priority, 10 – maximum priority. By default, 5 is used.

- **Store Admin Messages** – the setting that allows the user to store or not to store admin messages in HDD storage (e.g. messages for creating, maintaining and closing the connections between the services and the target server).

If admin messages are being stored, they will not be displayed in the report or on the Messages page, and they will not be obtained via the retrieve action.

Not storing admin messages will save the space in HDD storage.

- **Async Run Matrix** – this parameter is used for changing the matrix execution mode.

false – sequential execution (only one matrix will be executed at once);

true – execution in parallel (this mode allows to run up to 3 matrices at once, unless the same services are used in these matrices).

- **File Storage Path** – the directory where the files which contain all the sent/received messages are stored.

- **Storage Type** – this setting allows to set the type of HDD storage used for the long-term storage of the received/sent messages.

Supported storage types:

file – for each message, a separate file is created in the directory specified in the 'File Storage Path' field. This type of storage is recommended to be used when opening Sailfish for a short period of time: to check the connection to the test system, to run a smoke test, for demo demonstration.

When using this storage type, the user cannot query the messages on the Messages page.

db – messages are saved in a Database. The DB connection should be set up in the Database tab. This storage type is recommended for long-term work with Sailfish.

- **Relevant Messages Sorting Mode** – this setting allows to change the way of displaying similar messages in the failed actions of the report.

Supported parameters:

ARRIVAL_TIME – similar messages will be displayed in the same order as they have been received by the service (by default);

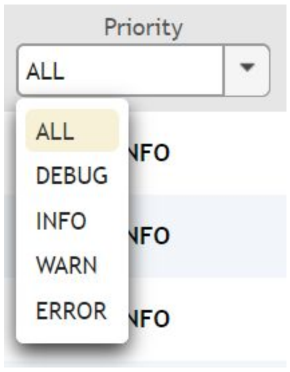
FAILED_FIELDS – similar messages will be displayed in the report sorted by the number of failed fields, from low to high. This function is only supported in an HTML report.

NOTE!: Changes of the following settings will be applied only after Sailfish restart:

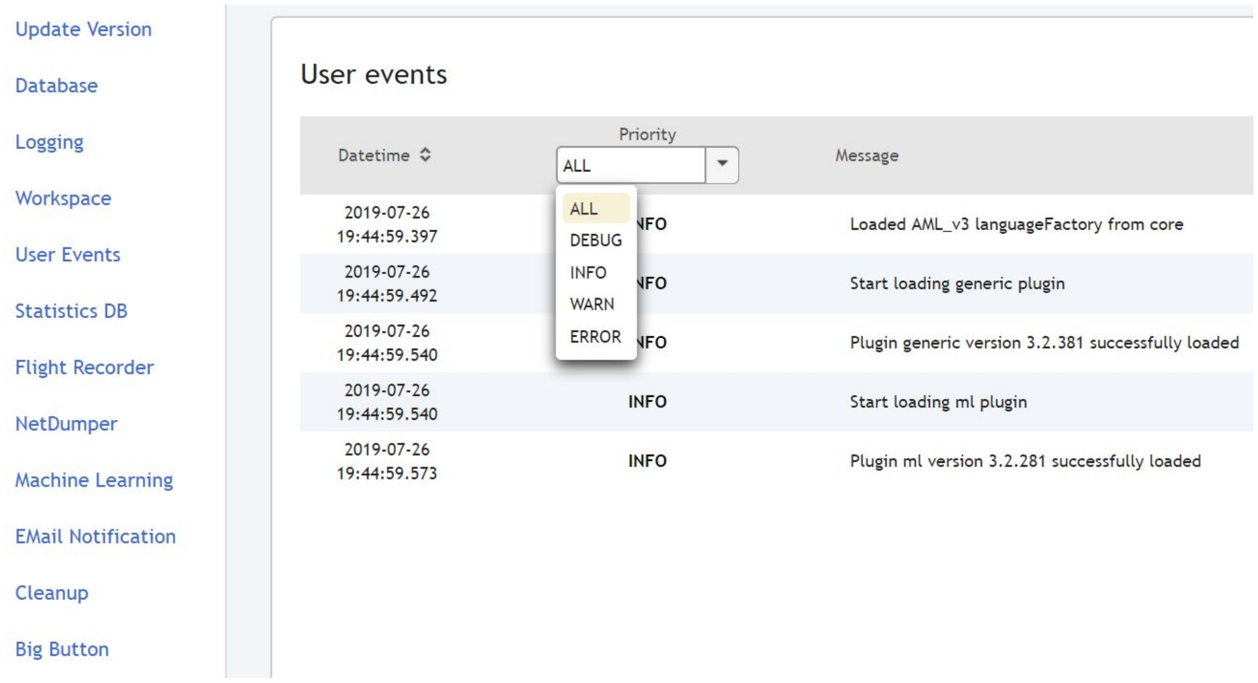
- Max Storage Queue Size
- Excluded messages
- Comparison Precision
- Matrix Compiler Priority
- Store Admin Messages
- Async Run Matrix
- File Storage Path
- Storage Type

5.7.5 User Events

By selecting the Priority from the drop-down menu, it is possible to see all the related events.



The messages will be displayed in the right part of the window.



Datetime	Priority	Message
2019-07-26 19:44:59.397	INFO	Loaded AML_v3 languageFactory from core
2019-07-26 19:44:59.492	INFO	Start loading generic plugin
2019-07-26 19:44:59.540	INFO	Plugin generic version 3.2.381 successfully loaded
2019-07-26 19:44:59.540	INFO	Start loading ml plugin
2019-07-26 19:44:59.573	INFO	Plugin ml version 3.2.281 successfully loaded

5.7.6 Statistics DB

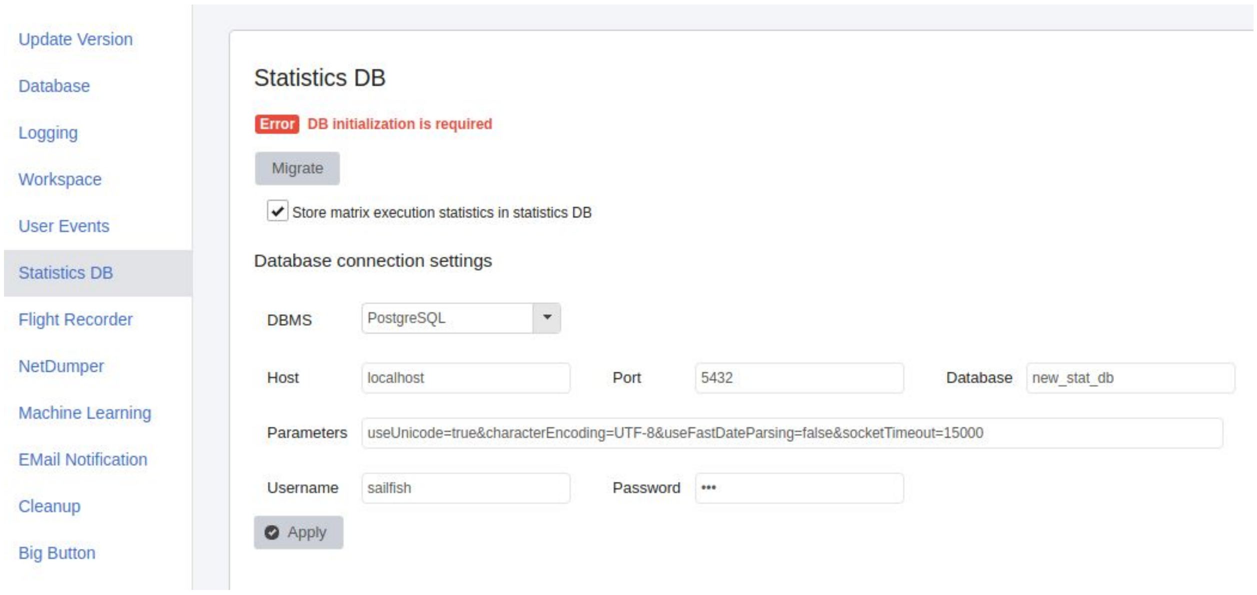
The collection of statistics allows to accumulate the test execution data, as information about matrices, timing, fails, etc., which allows the user to make test analysis, such as:

- Monitoring of the current work;
- Test runs history revision of each test case (which can be found by its unique #id);
- Analysis of test execution reports;
- Test suite runs comparison.

To save the information about matrices runs, a separate DBMS is used (MySQL and Postgresql are currently supported).

First, the user has to create a DB using one of the create_mysql_statistics_db/create_postgresql_statistics_db scripts, which can be found in the Sailfish core distributive, DB folder; or set up the connection to an existing DB from Configuration page > Statistics DB tab in Sailfish Web GUI.

If the user tries to connect to a newly created DB or a DB with the schema version below the current one, it will look as below.
The user will need to migrate the newly created DB.



Statistics DB

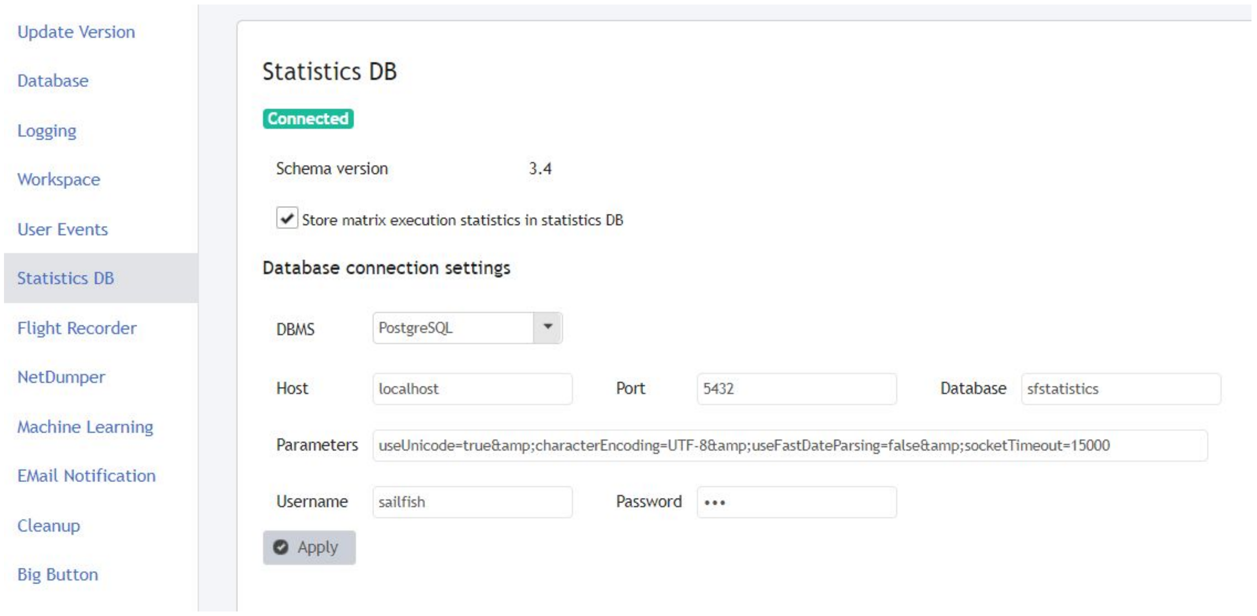
Error DB initialization is required

✓ Store matrix execution statistics in statistics DB

Database connection settings

DBMS	PostgreSQL				
Host	localhost	Port	5432	Database	new_stat_db
Parameters	useUnicode=true&characterEncoding=UTF-8&useFastDateParsing=false&socketTimeout=15000				
Username	sailfish	Password	•••		

Then, the user can change the Statistics database setup on the Sailfish Configuration page in the Statistics DB section.



Statistics DB

Connected

Schema version 3.4

✓ Store matrix execution statistics in statistics DB

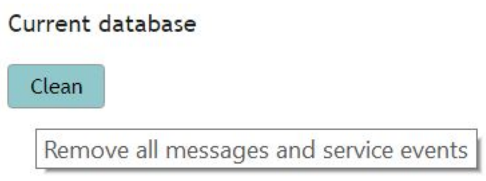
Database connection settings

DBMS	PostgreSQL				
Host	localhost	Port	5432	Database	sfstatistics
Parameters	useUnicode=true&amp;characterEncoding=UTF-8&amp;useFastDateParsing=false&amp;socketTimeout=15000				
Username	sailfish	Password	•••		

Check the "Store matrix execution statistics in statistics DB" and press “Apply”.

✓ Store matrix execution statistics in statistics DB

You can also remove all the messages and service events by pressing the ‘Clean’ button.

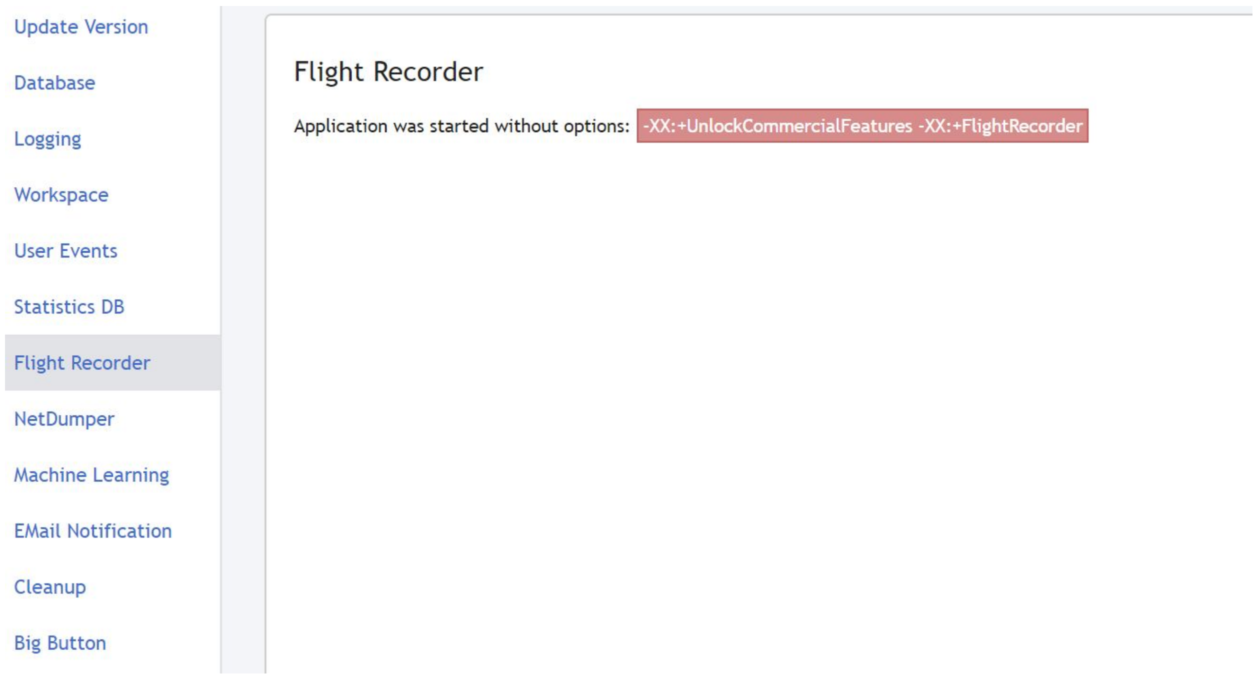


NOTE!: Several Sailfish instances on different hosts can be configured to use the same statistics database. Thus, all statistics from all configured hosts will be aggregated in the same database and can be retrieved from different hosts.

5.7.7 Flight Recorder

Flight Recorder is used for profiling recording (how many files have been run, how many resources have been spent, etc.)

Note: Flight Recorder is currently being incubated.



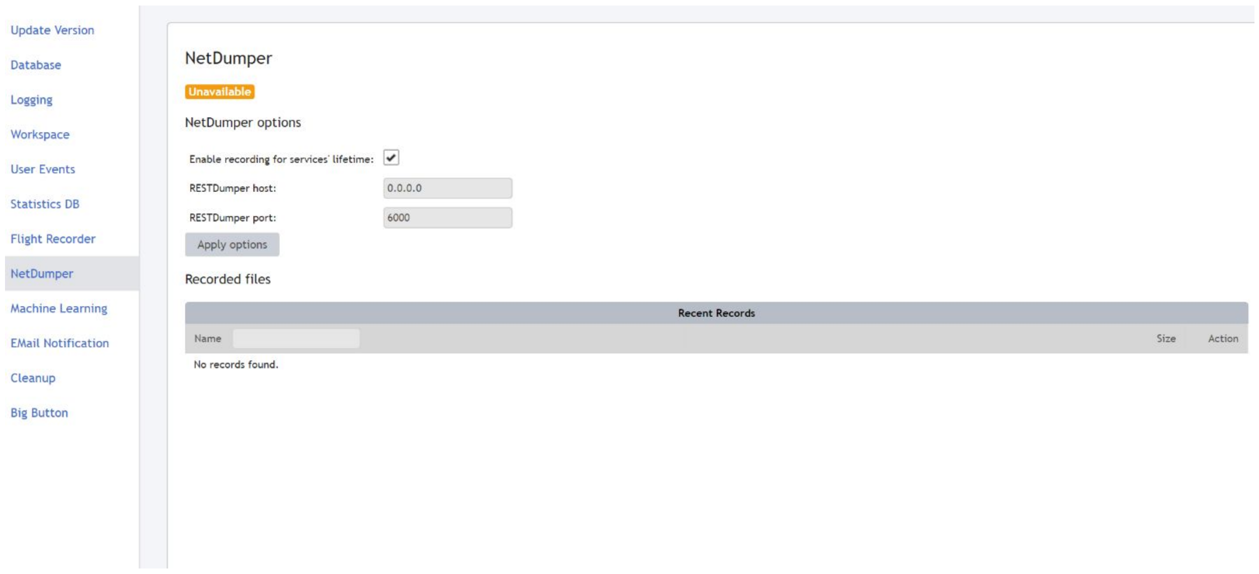
Application was started without options: -XX:+UnlockCommercialFeatures -XX:+FlightRecorder

5.7.8 NetDumper

NetDumper has been developed to record pickups for network traffic recording.

NetDumper has to be connected to a specific REST API instrument which is integrated into Sailfish before running a matrix, so that the network traffic is recorded. Pickups are attached to the execution reports.

Note: NetDumper is currently being incubated.



NetDumper

Unavailable

NetDumper options

Enable recording for services' lifetime: ✓

RESTDumper host: 0.0.0.0

RESTDumper port: 6000

Apply options

Recorded files

Recent Records		
Name	Size	Action
No records found.		

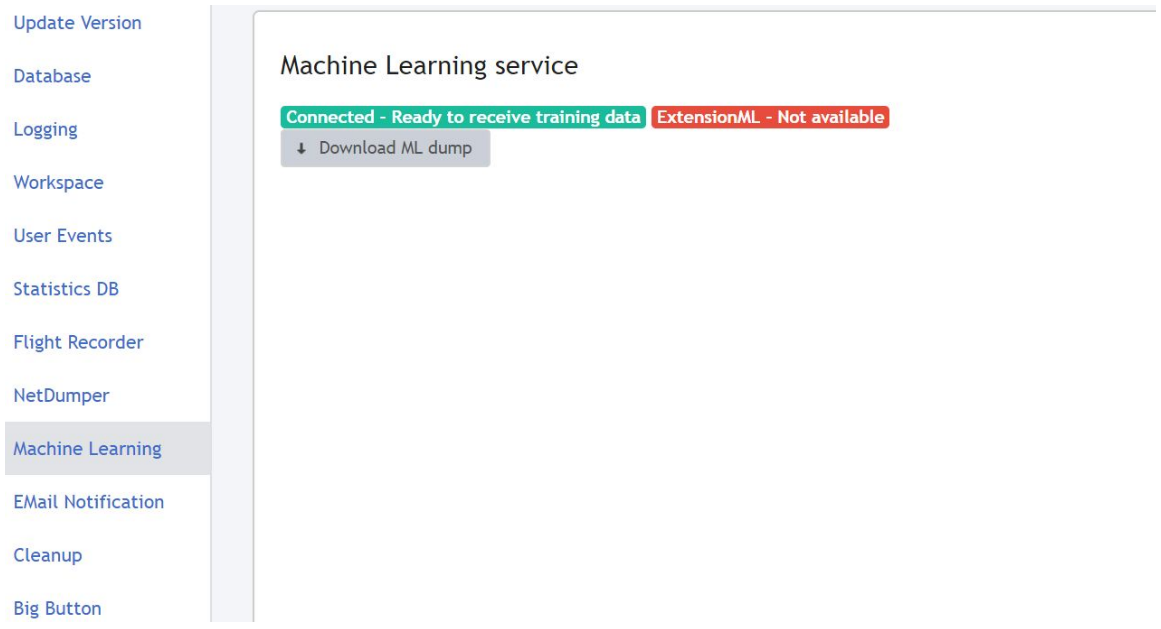
5.7.9 Machine Learning

The machine Learning functionality is used for getting machine learning data and analyzing the execution reports.

For analyzing the reports, an additional plugin must be integrated into Sailfish; the plugin depends on the project/client, etc. This plugin is developed based on the existing reports of a team and the logic of failing or passing the actions. Before this plugin is developed, test data should be collected.

Ready to receive the training data functionality – is used for collecting the test data, which, afterwards, will be used for the logic to be integrated in the plugin.

The status of ExtensionML shows if the plugin is integrated into Sailfish.



Machine Learning service

Connected - Ready to receive training data
ExtensionML - Not available

Download ML dump

By pressing Download ML dump, a .zip archive with training data will be downloaded.

5.7.10 Email Notification

On the Sailfish Configuration page in the EMail Notification section it is possible to enable Email Service by ticking

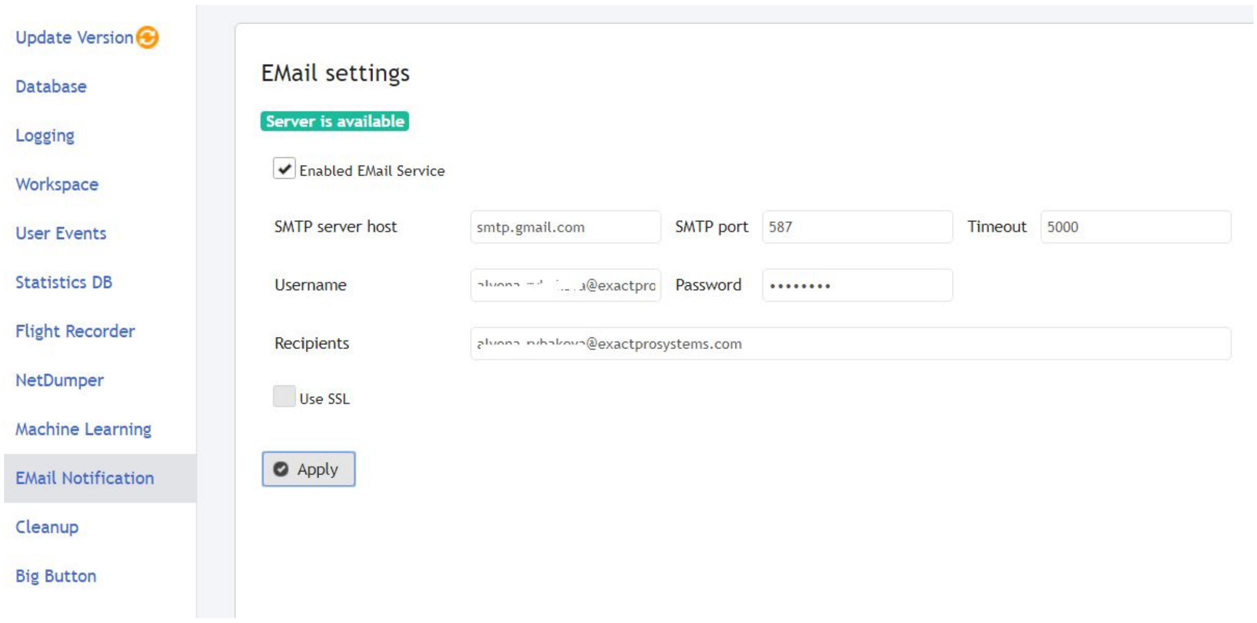
✓ Enabled EMail Service

checkbox and change the email settings.

Email Notification functionality is used for configuration of sending notifications with the execution results to the user's email.

1. First, Email service should be set up.

Example of Email Notification setup is shown below:



EMail settings

Server is available

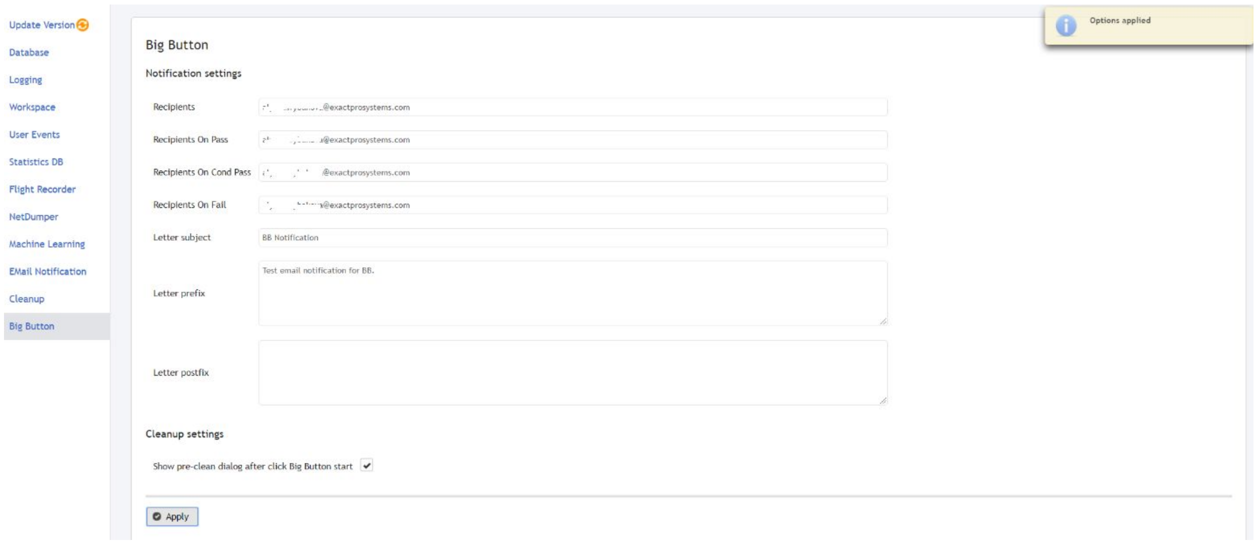
✓ Enabled EMail Service

SMTP server host	smtp.gmail.com	SMTP port	587	Timeout	5000
Username	[redacted]@exactpro	Password	••••••••		
Recipients	[redacted]@exactprosystems.com				

☐ Use SSL

2. Then, the Big Button notification should be set up.

Example is shown below:



Big Button

Notification settings

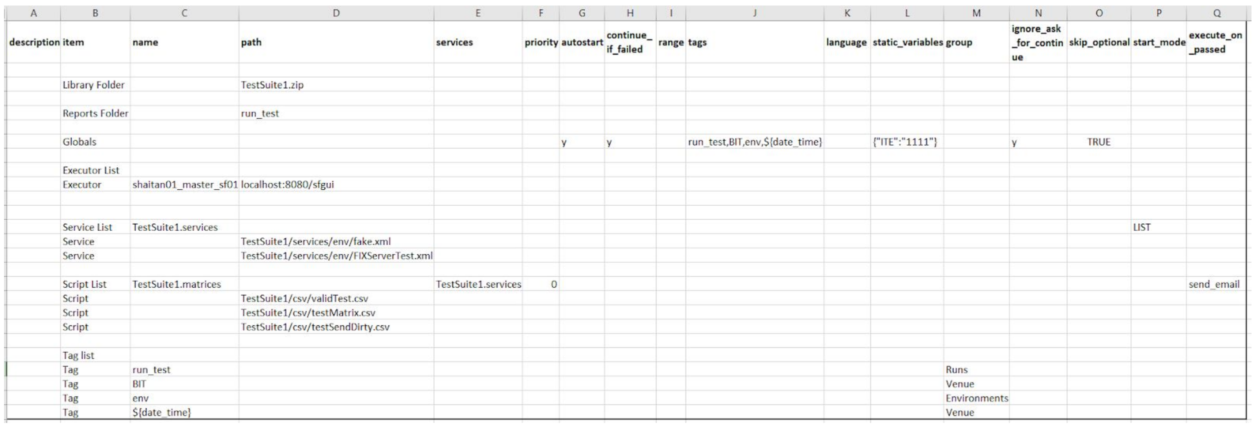
Recipients	[redacted]@exactprosystems.com
Recipients On Pass	[redacted]@exactprosystems.com
Recipients On Cond Pass	[redacted]@exactprosystems.com
Recipients On Fail	[redacted]@exactprosystems.com
Letter subject	BB Notification
Letter prefix	Test email notification for BB.
Letter postfix	

Cleanup settings

Show pre-clean dialog after click Big Button start ✓

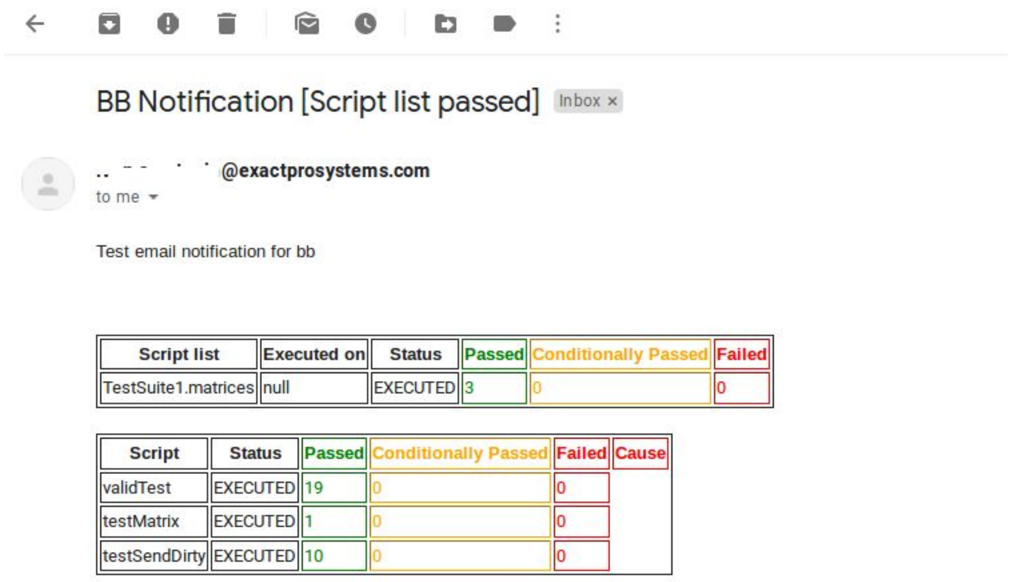
3. In Big Button config, the "send_email" action should be specified in the "execute_on_passed" and/or "execute_on_failed" and/or "execute_on_conditionally_passed" columns.

Example:



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
description	item	name	path	services	priority	autostart	continue_if_failed	range	tags	language	static_variables	group	ignore_ask_for_continue	skip_optional	start_mode	execute_on_passed
	Library Folder		TestSuite1.zip													
	Reports Folder		run_test													
	Globals					y	y		run_test,BIT,env,${date_time}		{"ITE":"1111"}		y	TRUE		
	Executor List															
	Executor	shaitan01_master_sf01	localhost:8080/sfgui													
	Service List	TestSuite1.services													LIST	
	Service		TestSuite1/services/env/fake.xml													
	Service		TestSuite1/services/env/FIXServerTest.xml													
	Script List	TestSuite1.matrices		TestSuite1.services	0											send_email
	Script		TestSuite1/csv/validTest.csv													
	Script		TestSuite1/csv/testMatrix.csv													
	Script		TestSuite1/csv/testSendDirty.csv													
	Tag list															
	Tag	run_test										Runs				
	Tag	BIT										Venue				
	Tag	env										Environments				
	Tag	${date_time}										Venue				

4. After running the Big Button config, you will receive the notification.



BB Notification [Script list passed]

[redacted]@exactprosystems.com

Test email notification for bb

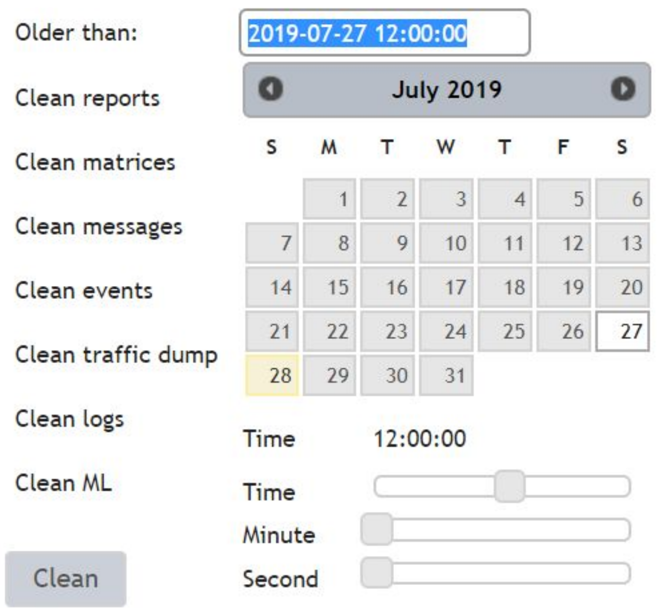
Script list	Executed on	Status	Passed	Conditionally Passed	Failed
TestSuite1.matrices	null	EXECUTED	3	0	0

Script	Status	Passed	Conditionally Passed	Failed	Cause
validTest	EXECUTED	19	0	0	
testMatrix	EXECUTED	1	0	0	
testSendDirty	EXECUTED	10	0	0	

5.7.11 Cleanup

Cleanup can be performed in the Cleanup tab on the Sailfish Configuration page.

First, select day/time from which the cleanup should be performed.


Older than: 2019-07-27 12:00:00

Clean reports

Clean matrices

Clean messages

Clean events

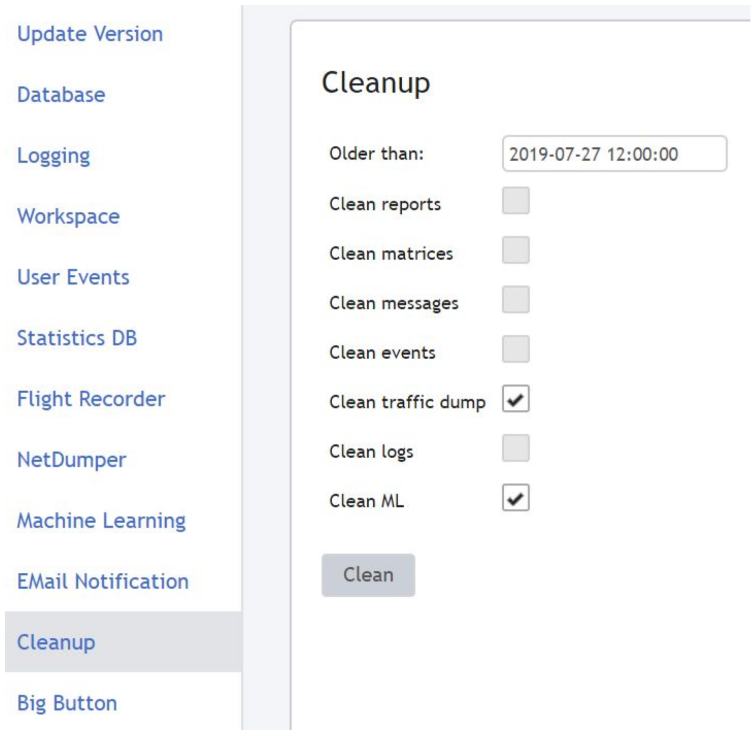
Clean traffic dump

Clean logs

Clean ML

Time 12:00:00

Then, tick the corresponding checkboxes depending on whether you need to clean the reports, matrices, etc.


Cleanup

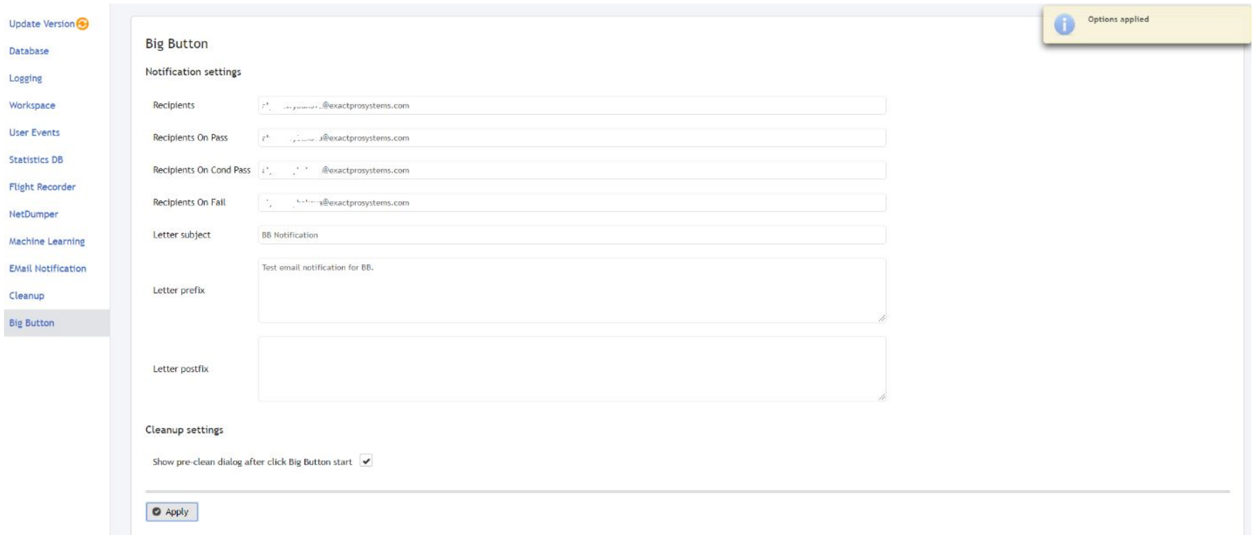
Older than:	2019-07-27 12:00:00
Clean reports	☐
Clean matrices	☐
Clean messages	☐
Clean events	☐
Clean traffic dump	✓
Clean logs	☐
Clean ML	✓

Clean

Once ticked, press Perform cleanup button to perform the cleanup.

5.7.12 Big Button

Using Big Button, the user can run several matrices on different instances of Sailfish, which are connected, and obtain execution reports on the main one.



Big Button

Notification settings

Recipients	[redacted]@exactprosystems.com
Recipients On Pass	[redacted]@exactprosystems.com
Recipients On Cond Pass	[redacted]@exactprosystems.com
Recipients On Fail	[redacted]@exactprosystems.com
Letter subject	BB Notification
Letter prefix	Test email notification for BB.
Letter postfix	

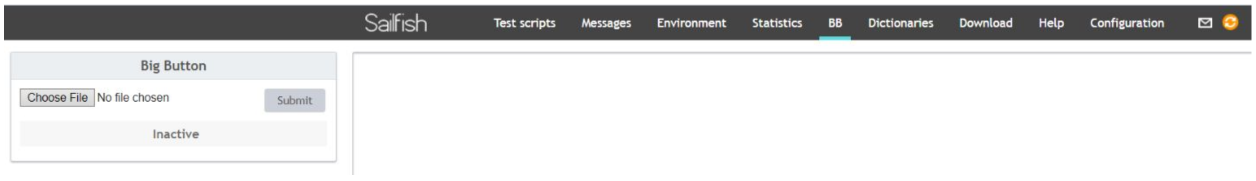
Cleanup settings

Show pre-clean dialog after click Big Button start ✓

5.8 Big Button Page

This is the page for automated execution of a structured test library.

Big Button functionality allows the user to run several matrices on different instances of Sailfish connected between each other and obtain execution reports on the main instance of Sailfish.



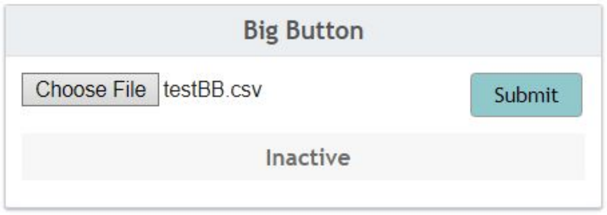
For working with BB the user will need:

- to have the BB configuration in .csv format;
- one or more started Sailfish instances;
- to have the library of scripts and services for the workspace.

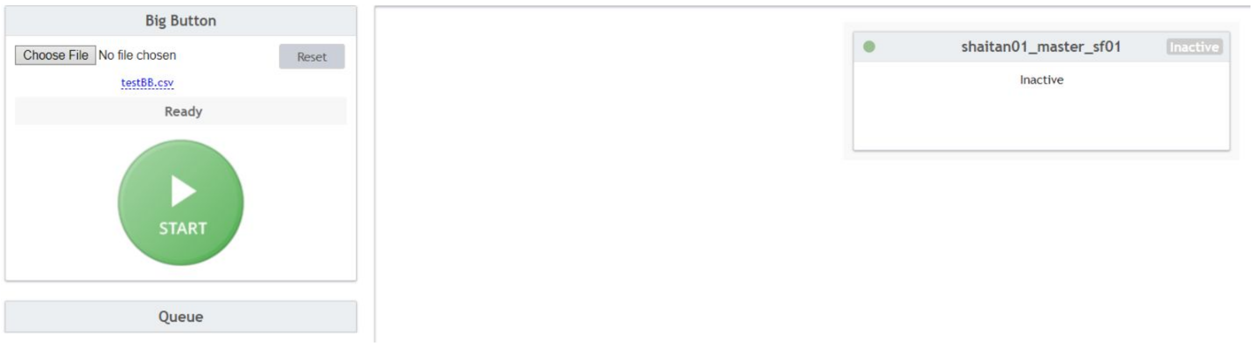
Main possibilities when working with BB are the following:

- individual and general setup of the parameters for the test set run;
- monitoring the test execution on several instances of Sailfish and changing the test script execution order, in case of unavailability of one Sailfish instance;
- collection of the statistics on test runs on all instances of Sailfish;
- REST API supported.

First, the user should load the configuration file from the local machine and, then, press Submit button.

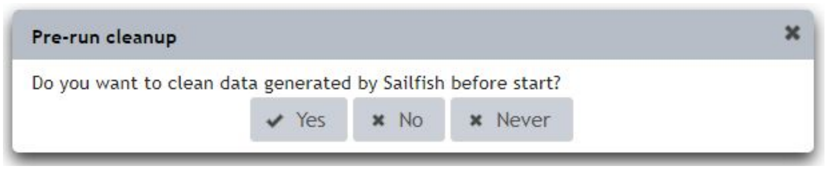


Then, the user can proceed with the execution of the script by pressing Start.



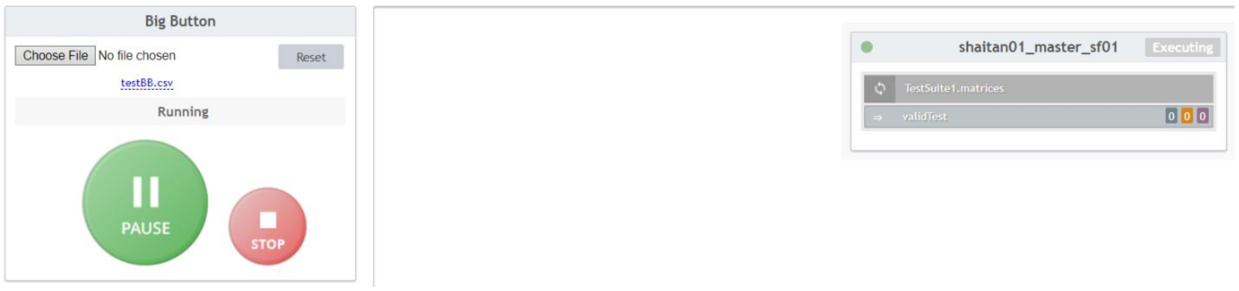
After the Start action is chosen and before the execution starts, a dialog is offered to the user requesting (optionally) to clean old data such as Sailfish logs, BB execution reports, etc.

Note: In case the user presses the Never button, once the corresponding setting in the Big Button tab in Sailfish is changed, the pop-up window with pre-run cleanup option will be shown again.

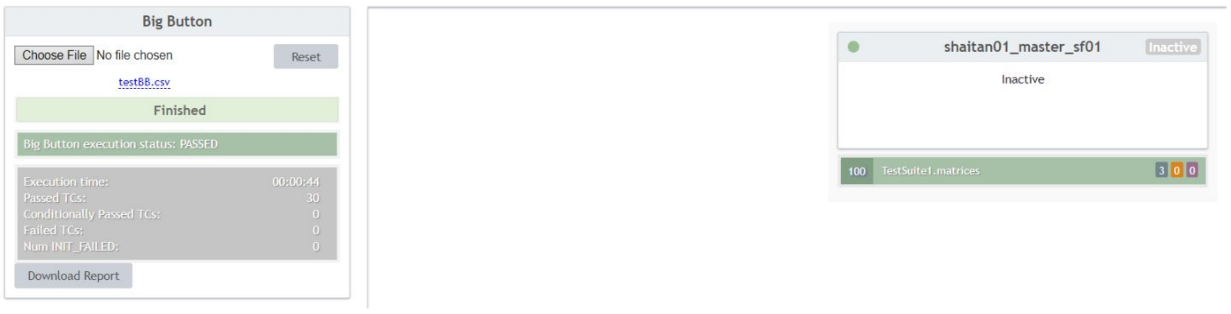

Pre-run cleanup

Do you want to clean data generated by Sailfish before start?

The process of running the BB test script will look as below:



Once the execution is done the Big Button execution status and the results can be observed.



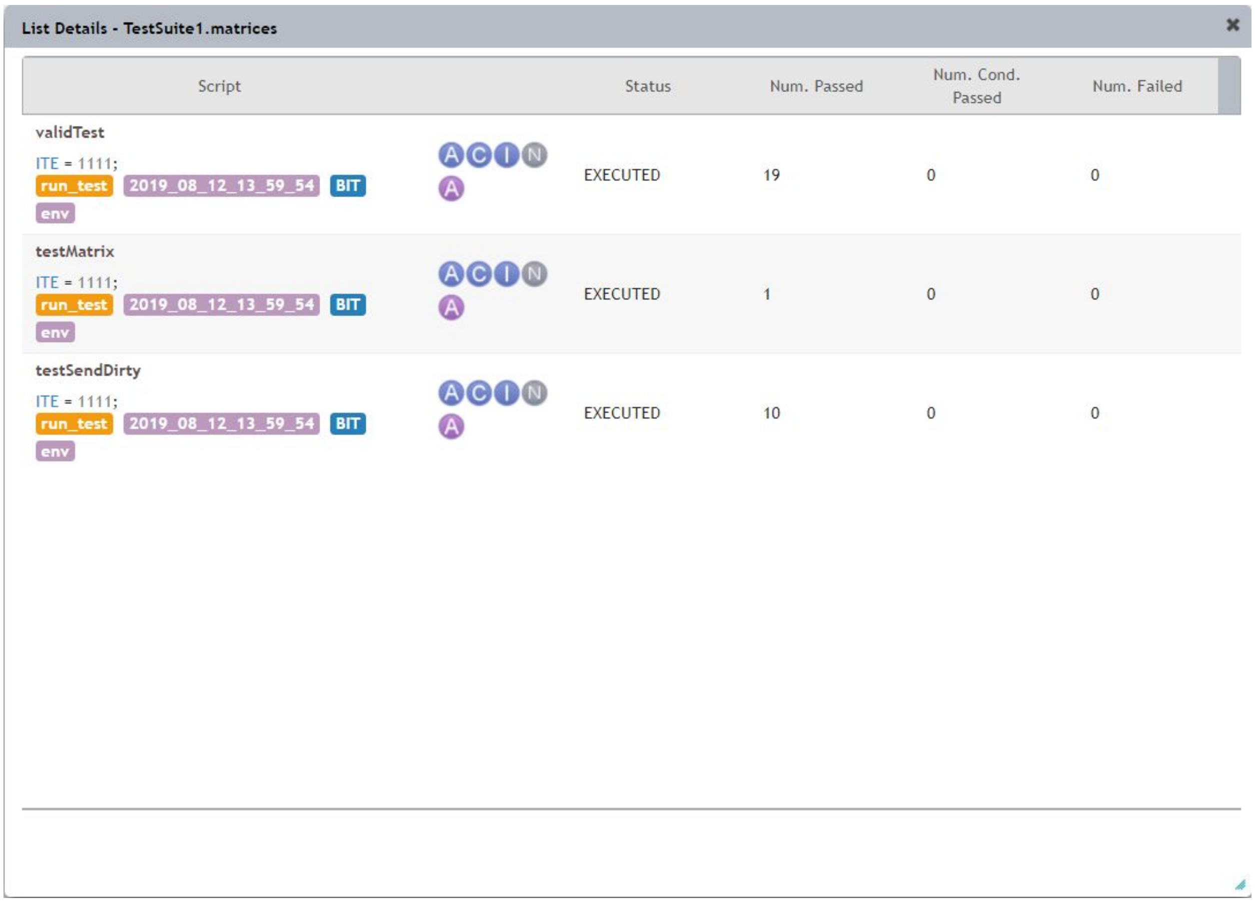
Big Button execution status: PASSED

Execution time:	00:00:44
Passed TCs:	30
Conditionally Passed TCs:	0
Failed TCs:	0
Num INIT_FAILED:	0

After execution, a high-level summary report is available for download in .csv format with the following information:

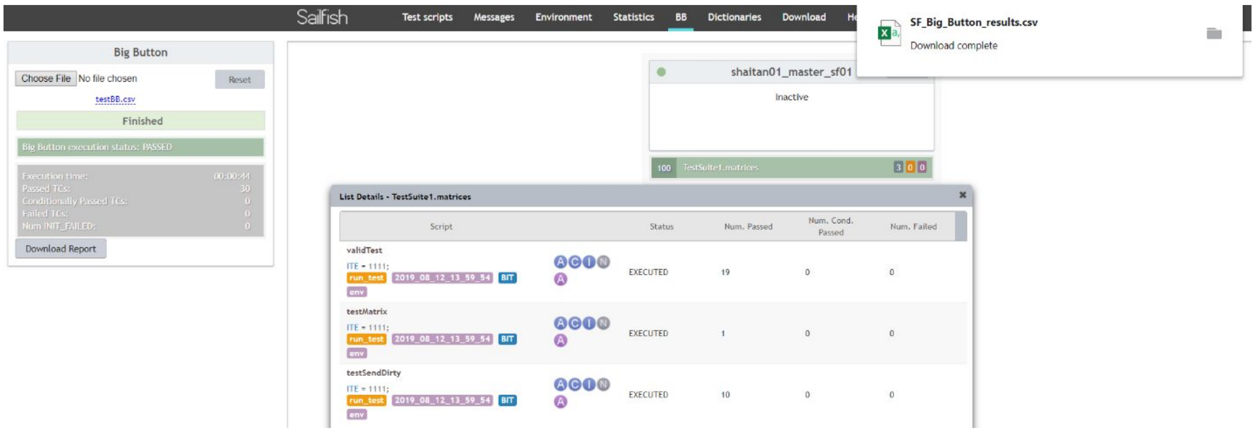
- Executed Scripts;
- Status (Executed, Interrupted or Init failed);
- The number of passed test cases per a script;
- The number of conditionally passed test cases per script;

- The number of failed test cases per a script.


List Details - TestSuite1.matrices

Script		Status	Num. Passed	Num. Cond. Passed	Num. Failed
validTest ITE = 1111; run_test 2019_08_12_13_59_54 BIT env	A C I N A	EXECUTED	19	0	0
testMatrix ITE = 1111; run_test 2019_08_12_13_59_54 BIT env	A C I N A	EXECUTED	1	0	0
testSendDirty ITE = 1111; run_test 2019_08_12_13_59_54 BIT env	A C I N A	EXECUTED	10	0	0

You can download the report with the results by pressing Download Report on the left part of the window:



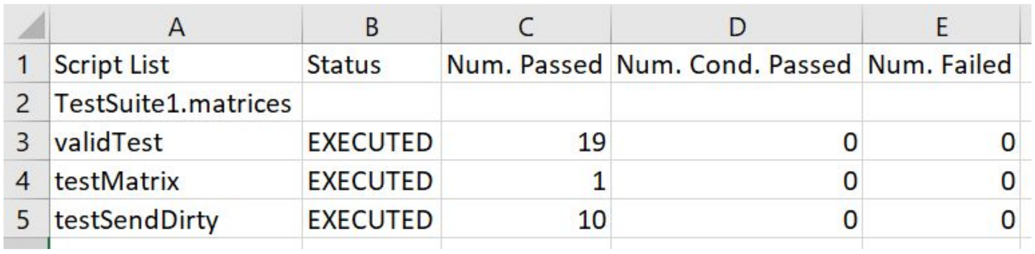
Big Button execution status: PASSED

Execution time:	00:00:44
Passed TCs:	30
Conditionally Passed TCs:	0
Failed TCs:	0
Num INIT_FAILED:	0

List Details - TestSuite1.matrices

Script		Status	Num. Passed	Num. Cond. Passed	Num. Failed
validTest ITE = 1111; run_test 2019_08_12_13_59_54 BIT env	A C I N A	EXECUTED	19	0	0
testMatrix ITE = 1111; run_test 2019_08_12_13_59_54 BIT env	A C I N A	EXECUTED	1	0	0
testSendDirty ITE = 1111; run_test 2019_08_12_13_59_54 BIT env	A C I N A	EXECUTED	10	0	0

The results will be shown in the .csv file as below:



	A	B	C	D	E
1	Script List	Status	Num. Passed	Num. Cond. Passed	Num. Failed
2	TestSuite1.matrices				
3	validTest	EXECUTED	19	0	0
4	testMatrix	EXECUTED	1	0	0
5	testSendDirty	EXECUTED	10	0	0

6 Sailfish Test Scripts basics

This chapter provides a brief overview of Sailfish test scripts and basic instructions about their support and update.

6.1 Test Actions

In simple terms, a Sailfish test script is a sequence of actions aimed either at sending a message to the system under test or at listening to the system's response. Typically, these actions go in pairs: an inbound test message is sent to the system and the corresponding outbound message is expected from the system. In addition to these input-output action pairs, there are auxiliary SF actions aimed at introducing pauses, sleep periods and counts.

Below is a brief overview of some of the available actions.

Example:

#Description	#action	#message_type	Side	OrderQty	Price
	test case start				
Sending a 'NewOrderSingle' client-initiated message to server according to the dictionary set up in the service or according to the value of #dictionary field.	send	NewOrderSingle	BUY	1000	%{Price}
Receiving an 'ExecutionReport' server-initiated message to client with status 'New'.	receive	ExecutionReport	${buyside_order.Side}	${buyside_order.OrderQty}	${buyside_order.Price}
Sending a 'NewOrderSingle' client-initiated message to server according to the dictionary set up in the service or according to the value of #dictionary field.	send	NewOrderSingle	SELL	1000	%{Price}
Receiving an 'ExecutionReport' server-initiated message by server to client with status 'New'.	receive	ExecutionReport	${sellside_order.Side}	${sellside_order.OrderQty}	${sellside_order.Price}
Receiving an 'ExecutionReport' server-initiated message to client with status 'Filled'.	receive	ExecutionReport	${buyside_order.Side}	${buyside_order.OrderQty}	${buyside_order.Price}
Receiving an 'ExecutionReport' server-initiated message to client with status 'Filled'.	receive	ExecutionReport	${sellside_order.Side}	${sellside_order.OrderQty}	${sellside_order.Price}
	test case end				

6.1.1 'Send' action

The 'Send' action is used to send an **inbound** test message into the system under test.

- The message will be constructed according to the rules of the selected interface dictionary;
- The message will be routed to the server via an interface channel and on behalf of the user according to the selected Sailfish service;
- The message will contain all fields and values that were selected for this operation.

6.1.2 'Receive' action

The 'Receive' action is used to define the parameters of an **outbound** system message so that it could be filtered and found among the others. This is a verification message which can have a Passed or a Failed result. To Pass this action, the outbound system message should comply with all three conditions listed below:

1. The message should be received via the service and by the user that were defined for this verification;
2. The message should be received within a defined timeout;
3. The message should contain exactly the same pairs of fields/values that were defined for this verification (However, the incoming messages may contain more optional fields than was defined in the matrix).

If any one of the above conditions was not met, the 'Receive' action will fail.

Actually, the 'Receive' action defines the parameters of an outbound system message, which is expected as a response to an inbound ('Send' action) message.

6.1.3 'Retrieve' action

The 'Retrieve' action works the same way as the 'Receive' action does, except for one difference:

- For the 'Receive' action, Sailfish will be searching for a message with predefined parameters in the Sailfish's memory (RAM storage; which contains only the messages that correspond to the current test case) and the search range will be limited by the action timeout and checkpoint;
- For the 'Retrieve' action, Sailfish will be searching for a message with predefined parameters in the Sailfish's storage (HDD storage, which contain all the messages received by the service), and the search range is not restricted.

6.1.4 'Count' action

The 'Count' action works like the 'Receive' action, with an extra condition: to Pass the action, the number of received messages should be equal to the parameter that was defined for this verification.

6.1.5 Other actions

There is a number of auxiliary and service actions that can be used to improve test script usability and performance. The most popular actions are:

- **GetCheckPoint** - to set the time marker in memory of Sailfish where all outbound system messages for all enabled services are stored during test case execution; the marker can be further used by the 'Receive' action to start the limit of the range in Sailfish cache in which the expected message should be searched;
- **Sleep** - to set delays – where and if needed - in milliseconds before other actions;

- **AskForContinue** - is used to pause the test case. It offers the user to "stop" or "continue" test execution;
- **SetVariables** - is used to define variables. It should be used with the following columns:
  - #reference — variable name;
  - with any other columns (tags) that contain values of the declared variable;
- **SetStatic** - to pass the parameters between test cases. It uses the following columns:
  - #reference — static variable name;
  - #static_type — variable type; Java simple types and BigDecimal, LocalDate, LocalTime, LocalDateTime are supported.
  - #static_value — static variable value.

A static variable can be set by a constant (e.g. 100 or VALUE) or by reference to a previously defined ${order.field_name} field. A static variable cannot be set by reference to the #{function_name} function. A static variable can be referenced from an action using the following syntax: %{static_variable_name}

For a complete list of available actions, refer to Sailfish HELP.

6.2 Test Case Structure

Every test script consists of one or more test cases.

Every test case consists of one or more test actions.

Every test case should begin with “Test case start” and should end with “Test case end” statements in the #Action column. The rest of the test case’s content should be placed between these two operators; it completely depends on the testing scenario and may contain:

- Repeating groups;

#description	#reference	#action	#message_type	TradingParty	Side	OrderQty	Price
Start of test case definition. Basic type of blocks. Send Limit orders: buy 1000@105 + sell 1000@105 -> trade (Test case status - Passed)		test case start					
Repeating group	TradingParty_ref	[NoPartyIDs_ref]					
Get a checkpoint to limit the search scope to the Messages received after this checkpoint.	check1	GetCheckPoint					
Sending a 'NewOrderSingle' client-initiated message to server according to the dictionary set up in the service or according to the value of the #dictionary field.	buyside_order	send	NewOrderSingle	[TradingParty_ref]	BUY	1000	1000
Receiving an 'ExecutionReport' server-initiated message to client with status 'New'	exr_buy	receive	ExecutionReport	[TradingParty_ref]	${buyside_order.Side}	${buyside_order.OrderQty}	${buyside_order.Price}
End of test case definition.		test case end					

- Static Variables;

#reference	#action	#message_type	#static_type	#static_value	SecurityID	Price
	test case start					
SecurityID	SetStatic		String	ABC		
Price	SetStatic		BigDecimal	1000		
buyside_order	send	NewOrderSingle			%{SecurityID}	%{Price}
exr_buy	receive	ExecutionReport			${buyside_order.SecurityID}	${buyside_order.Price}
sellside_order	send	NewOrderSingle			%{SecurityID}	%{Price}
exr_sell	receive	ExecutionReport			${sellside_order.SecurityID}	${sellside_order.Price}
	test case end					

- Checkpoints;

#description	#reference	#check_point	#action
			test case start
Get a checkpoint to limit the search scope to the Messages received after this checkpoint.	check1		GetCheckPoint
Sending a 'NewOrderSingle' client-initiated message to server according to the dictionary set up in the service or according to the value of the #dictionary field.	buyside_order		send
Receiving an 'ExecutionReport' server-initiated message to client with status 'New'	exr_buy	check1	receive
Get a checkpoint to limit the search scope by only Messages received after this checkpoint.	!check2		GetCheckPoint
Sending a 'NewOrderSingle' client-initiated message to server according to the dictionary set up in the service or according to the value of the #dictionary field.	sellside_order		send
Receiving an 'ExecutionReport' server-initiated message to client with status 'New'	exr_sell	!check2	receive
Receiving an 'ExecutionReport' server-initiated message to client with status 'Filled'	exr_buy_f1	!check2	receive
Receiving an 'ExecutionReport' server-initiated message to client with status 'Filled'	exr_buy_f2	!check2	receive
			test case end

(*) – When a reference for checkpoint starts with ! - that means it is a SmartCheckpoint which automatically moves to the position after the found message upon a passed received action.

- Functions;

#action	#message_type	TransactTime
test case start		
send	NewOrderSingle	#{TransactTime("")}
receive	ExecutionReport	*
test case end		

- Actions.

#description	#action
	test case start
Sending a 'NewOrderSingle' client-initiated message to server according to the dictionary set up in the service or according to the value of the #dictionary field.	send
Receiving an 'ExecutionReport' server-initiated message to client with status 'New'	receive
	test case end

Repeating groups can be specified by using the following syntax:

[reference_1, reference_2]

[${reference_1}, ${reference_2}]

Static Variables can be referenced from the action's fields using the following syntax: %{static_variable_name}.

References can be specified by using the following syntax:

- ${reference} – this is a reference to a message. This syntax is used for specifying a repeating group;
- ${reference.field} – this is a reference to a field of a message or a sub-message.

Functions can be specified by using the following syntax: #{function_name}

6.3 Test Case Syntax

Sailfish Test Scripts have flexible syntax with just a few mandatory rules, which are listed below:

- "Test case start" / "Test case end" statements should be set for every test case;
- In the first row, the test script (.csv/.xls/.xlsx file) should contain a Header.

It is also recommended that the Sailfish system fields (marked with #) precede optional interface tags.

Please see the descriptions of Sailfish service fields in the test script Header below:

- #execute – an optional flag to control the execution of a current action or test case. It is possible to switch off the execution of a particular test case/cases by specifying "N". Valid values: "Y" and "N", blank value defaults to "Y";
- #comment – an option field used for the comments or the scenario description. If the value of the column is 'y', then other columns may have any text. Sailfish ignores similar lines.
- #description – used to describe a row. It is displayed in the report;
- #reference – allows to refer to parameters of a certain row later in the test script;
- #service_name – the name of a service which should send or receive the message;
- #timeout – the maximum time for waiting for the expected message in milliseconds (messages are expected in the received actions); the next action will not be processed until this time is elapsed or the expected message is received;
- #check_point – defines a starting point for waiting for the expected message. Together with the #timeout, the value defines the period for waiting for the expected message;
- #action – action name (case sensitive);
- #message_type – defines the type of message from the dictionary specified in the service;
- #add_to_report – an optional flag to control the appearance of an action in the report. Valid values: "Y" and "N", blank value defaults to "Y".

6.4 Test Case Formulae

Sailfish Test Script formulae may include the following components and their combinations:

- arithmetic operations, such as +; -; *; /;
- numbers (e.g. 1000);

- static variable such as %{static_variable_name};
- references to a previously defined fields such as ${reference.field_name} e.g. ${ord1.OrderQty};
- functions such as #{function_name}.

Note: The operation becomes calculable only having ${...}, %{...}, #{...} before them.

E.g.

{ord0.OrderQuantity}-${ord2.OrderQuantity}-${ord1.OrderQuantity}; ${ord0.OrderQuantity}-1000; ${itch_es_1.Turnover}+100000*103)/(${itch_es_1.Volume}+100000) etc.

7 Common services

Sailfish uses **Services** in order to communicate with the system under test or any other external application. Services are usually provided as a part of a plugin, but a few dummy ones are provided with SF itself.

Services are typically responsible for:

1. Setting up a connection with an external app (e.g. system under test);
2. Maintaining a connection;
3. Converting messages from an app's native protocol to the SF representation;

The user can manage SF Services via **Environment** page. Please refer to 5.1 Environment Page section.

SF services can be accessed from a test script using **actions**. To check the Test Actions, please refer to section 6.1.

7.1 Integration with DBMS

It's possible to execute custom SQL queries from the test script. To do that, you need a service that supports the **DB_Query** action.

The test script should contain:

```
#action - DB_Query
#message_type – HashMap
```

- The query should be specified in the 'DB_QUERY' field;
- The query should be written using native SQL dialect for the target DBMS.

Fields from the returned result set can be used via field reference syntax (${<message>.<field>}. Field names will correspond to the ones specified in the SQL SELECT statement.

If you want to pass some variables from a test script as query arguments, you need to:

1) Define the required variables as "**static variables**", e.g. "AdminName";

2) Pass the values to a **DB_QUERY** action by referencing **static variables** in the corresponding columns like this: "%{AdminName}" ;

3) Reference the passed values from the SQL code via this syntax: '@{<custom field name>}', e.g. @{USER_NAME}.

Below is an example of using a DB query in a script.

#description	#reference	#service_name	#action	#static_value	#static_type
define admin name	AdminName		SetStatic	JoeTheAdmin	String

#description	#reference	#service_name	#action	DB_QUERY	USER_ID	USER_NAME
first query	firstResultSet	DB_connection	DB_Query	SELECT USER_ID FROM USER_LIST WHERE USER_NAME = '@{USER_NAME}'		%{AdminName}
second query	secondResultSet	DB_connection	DB_Query	SELECT GROUP_NAME FROM GROUP_LIST WHERE ADMIN_ID = '@{USER_ID}'	${firstResultSet:USER_ID}	

The USER_ID field that was obtained from a database can be referenced later with this statement: ${firstResultSet:USER_ID}.

<u>Note:</u> To work with an object-oriented BD, you need to have a jar with a jdbc driver as a plugin library.

The service settings are below.
E.g. Service settings to connect to MySQL server:

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<serviceDescription>
    <type>**ORACLE-DB-Client**</type>
    <name>**Service_name**</name>
<serviceHandlerClassName>com.exactprosystems.testtools.services.CollectorServiceHandler</serviceHandlerClassName>
    <oracleClientSettings>
        <expectedTimeOfStarting>2000</expectedTimeOfStarting>
        <waitingTimeBeforeStarting>0</waitingTimeBeforeStarting>
        <idleTimeout>1000</idleTimeout>
        <jdbcDriver>**org.mariadb.jdbc.Driver**</jdbcDriver> - applicable jdbc driver (e.g. for mariadb).
        <password>**Password**</password> - password to connect to DB.
        <url>jdbc:mariadb://<host>[:<port>]/<databaseName></url> - url to connect to DB where the host address, port and DB name should be specified.
        <username>**User**</username> - user to connect to DB.
    </oracleClientSettings>
</serviceDescription>

E.g. Service settings to connect to Oracle server:

<serviceDescription>
    <type>**ORACLE_DB_Client**</type>
    <name>**Service_name**</name>

```
  <serviceHandlerClassName>com.exactprosystems.testtools.services.CollectorServiceHandler</servic
  eHandlerClassName>
    <oracleClientSettings>
      <expectedTimeOfStarting>2000</expectedTimeOfStarting>
      <performDump>false</performDump>
      <waitingTimeBeforeStarting>0</waitingTimeBeforeStarting>
      <idleTimeout>1000</idleTimeout>
      <jdbcDriver>oracle.jdbc.driver.OracleDriver</jdbcDriver>
      <password>Password</password>
      <url>jdbc:oracle:thin:@host:port/DB_name</url>
      <username>User</username>
    </oracleClientSettings>
  </serviceDescription>
```

E.g. Service settings to connect to Microsoft SQL server:

```
  <serviceDescription>
    <type>ORACLE-DB-Client</type>
    <name>Service_name</name>
  <serviceHandlerClassName>com.exactprosystems.testtools.services.CollectorServiceHandler</servic
  eHandlerClassName>
    <oracleClientSettings>
      <expectedTimeOfStarting>2000</expectedTimeOfStarting>
      <waitingTimeBeforeStarting>0</waitingTimeBeforeStarting>
      <idleTimeout>1000</idleTimeout>
      <jdbcDriver>com.microsoft.sqlserver.jdbc.SQLServerDriver</jdbcDriver>
      <password>Password</password>
      <url>jdbc:sqlserver://host:password;databaseName=<DB_name>;</url>
      <username>User</username>
    </oracleClientSettings>
</serviceDescription>
```

8 Test run results

Each Sailfish test script executed either automatically or manually will result in one of the following:

- Initialization failed (if Sailfish was not able to find a java source from a script due to wrong syntax);
- Run failed (if at least one service was not started by the “auto start” option);
- Passed (e.i. all verifications in the script matched the expected results);
- Failed (e.i. at least one verification in the test script did not match the expected result);
- Conditionally passed (e.i. at least one verification in the test script matched a result that was marked as a known issue in the system under test).

In each case (except script initialization failure, which will only provide an AML error), a detailed report will be generated automatically and stored in the default Sailfish ‘reports’ folder (the default location is <Deployed Tomcat>\webapps\sfgui\reports).

The report can be accessed either from the Sailfish UI ‘Test Scripts’ tab (the panel on the right) or from the “Script Run History” section of the Sailfish UI ‘Statistics’ tab.

The report can be downloaded as a .zip archive to the hard disk and browsed offline as a standard HTML file.

9 Test failures investigation

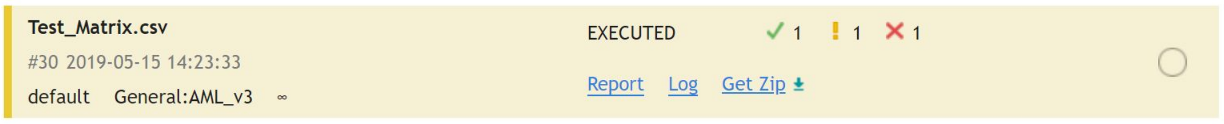
The Sailfish Regression Test library usually has the following hierarchy:

- Test Library consisting of
  - Test Scripts consisting of
    - Test Cases consisting of
      - Test Actions.

If any Test Action fails, its Test Case and Test Script fail too.

On the Test Script page, the report on the Sailfish ‘Test scripts’ tab is highlighted with:

- Red if the script failed due to INIT FAILED / RUN FAILED;
- Yellow if the script was EXECUTED but has some failed cases;
- Green if the script was EXECUTED, and all the test cases passed or conditionally passed.



Test_Matrix.csv EXECUTED ✓ 1 ! 1 ✗ 1

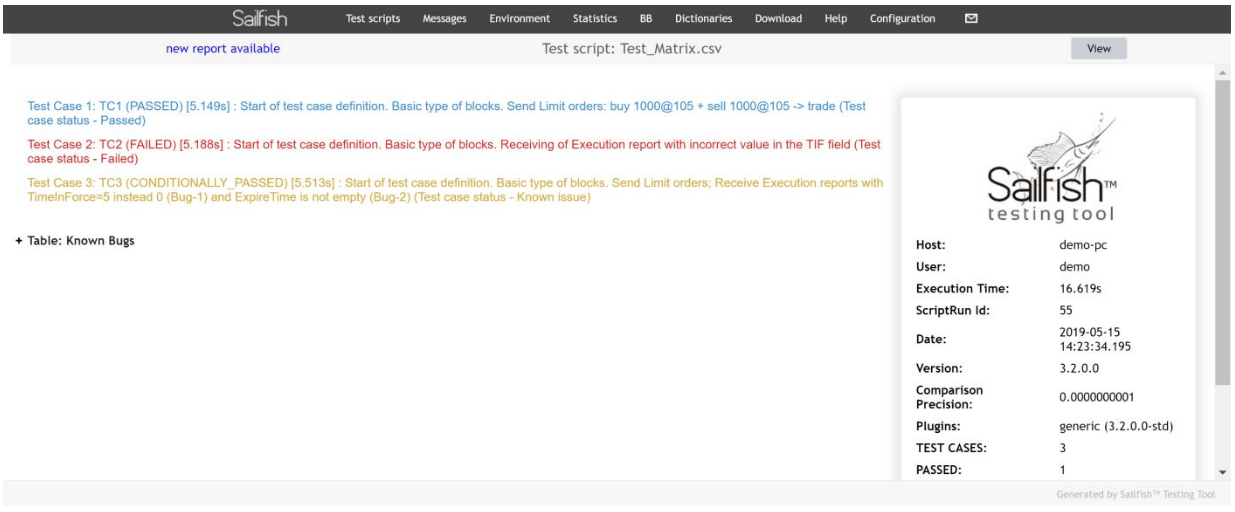
#30 2019-05-15 14:23:33 Report Log Get Zip

default General:AML_v3

The report can be downloaded as a .zip archive or opened on a new web page via the ReportLink.

On the report page, there is a list of all test cases that are a part of the script, with their descriptions. They are highlighted with:

- Red for failed TEST cases;
- Blue for passed test cases;
- Yellow for conditionally passed test cases (with known issues).



Test script: Test_Matrix.csv

Test Case 1: TC1 (PASSED) [5.149s] : Start of test case definition. Basic type of blocks. Send Limit orders: buy 1000@105 + sell 1000@105 -> trade (Test case status - Passed)

Test Case 2: TC2 (FAILED) [5.188s] : Start of test case definition. Basic type of blocks. Receiving of Execution report with incorrect value in the TIF field (Test case status - Failed)

Test Case 3: TC3 (CONDITIONALLY_PASSED) [5.513s] : Start of test case definition. Basic type of blocks. Send Limit orders; Receive Execution reports with TimeInForce=5 instead 0 (Bug-1) and ExpireTime is not empty (Bug-2) (Test case status - Known issue)

+ Table: Known Bugs

Host:	demo-pc
User:	demo
Execution Time:	16.619s
ScriptRun Id:	55
Date:	2019-05-15 14:23:34.195
Version:	3.2.0.0
Comparison Precision:	0.0000000001
Plugins:	generic (3.2.0.0-std)
TEST CASES:	3
PASSED:	1

Generated by Sailfish™ Testing Tool

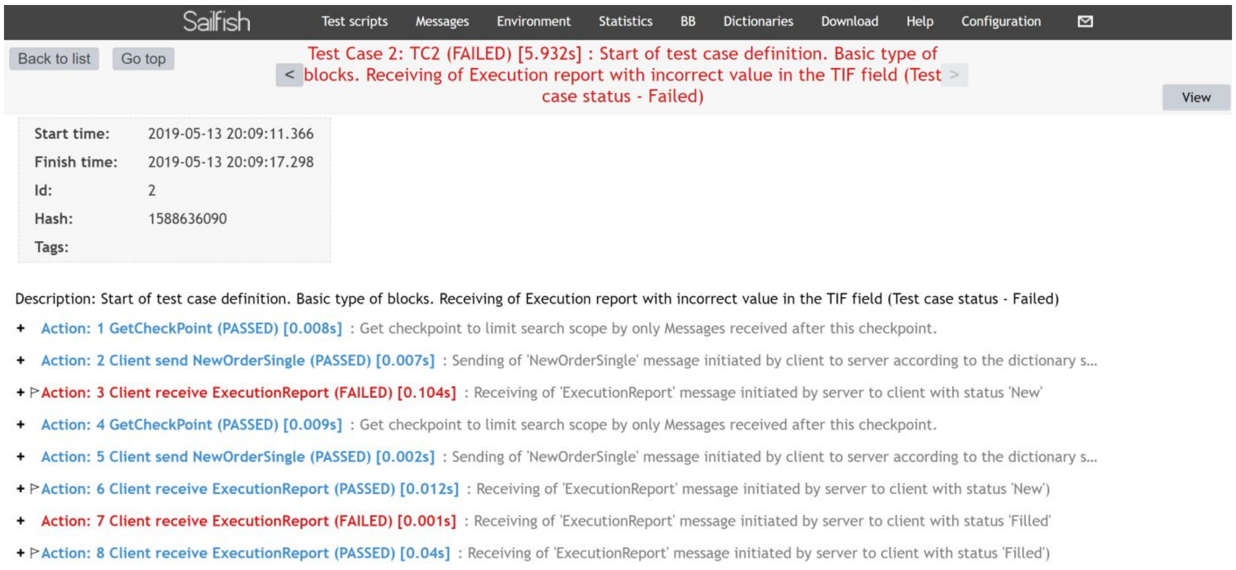
Left click on a row with a Test Case name. A detailed report will open with information as follows:

- Start Time;
- Finish Time;
- Unique Case ID (if filled in matrix);

- Hash of Unique Case ID;
- View menu which provides an ability to choose if all cases/fields should be displayed or only the fields;
- The list of Actions in the Test Case with their descriptions.

Left click on a row with a Test Action name. A detailed report will be opened with information as follows:

1. A table with Action details (list of Action Parameters);
2. Action Status with the failure reason;
3. A table with Similar messages (for Receive or Retrieve action);
4. Tables with parameters of Similar messages with Expected and Actual values for each parameter (Highlighted with red if there is a discrepancy).



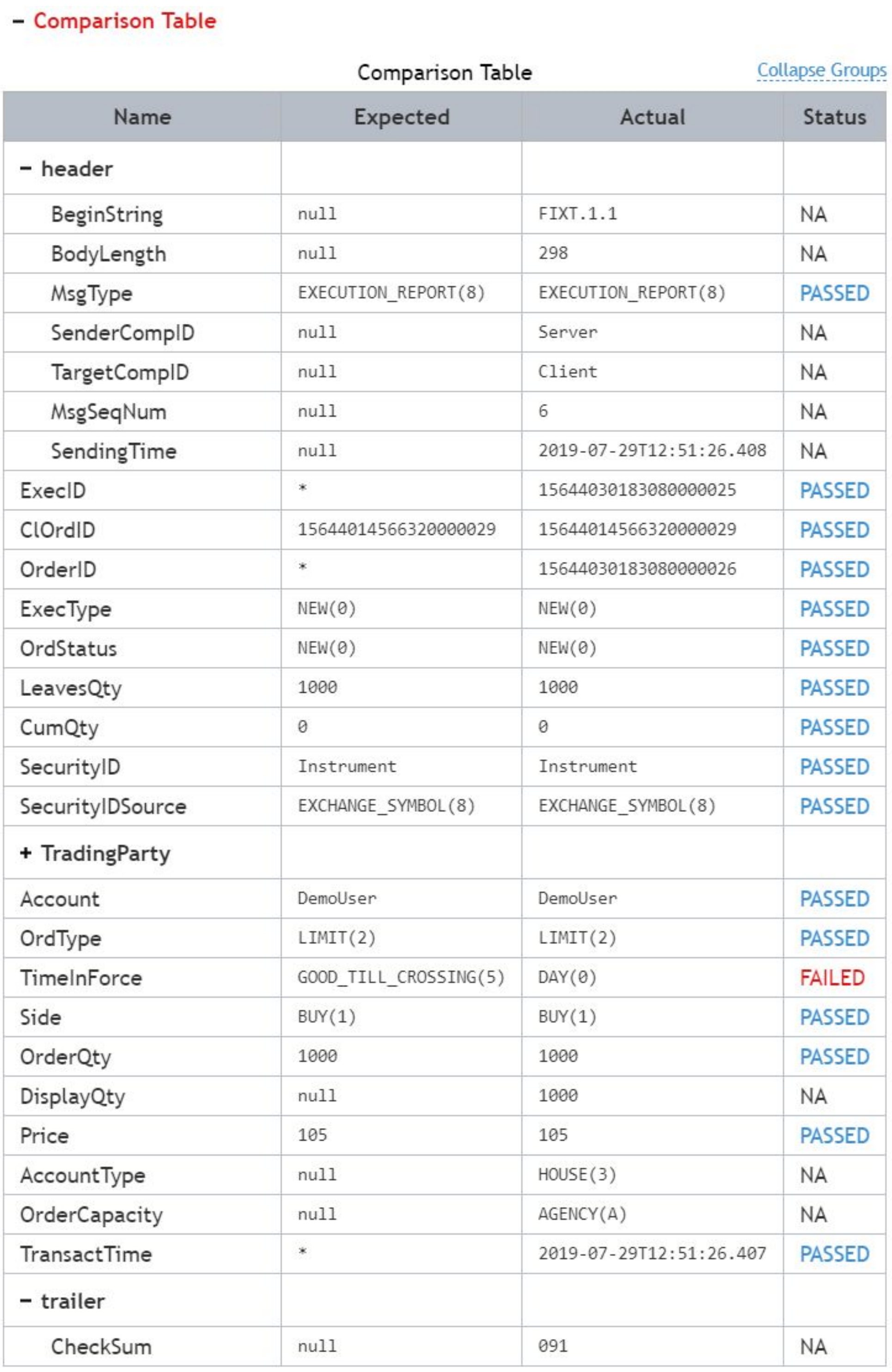
Test Case 2: TC2 (FAILED) [5.932s] : Start of test case definition. Basic type of blocks. Receiving of Execution report with incorrect value in the TIF field (Test case status - Failed)

Start time:	2019-05-13 20:09:11.366
Finish time:	2019-05-13 20:09:17.298
Id:	2
Hash:	1588636090
Tags:	

Description: Start of test case definition. Basic type of blocks. Receiving of Execution report with incorrect value in the TIF field (Test case status - Failed)

+ Action: 1 GetCheckPoint (PASSED) [0.008s] : Get checkpoint to limit search scope by only Messages received after this checkpoint.
+ Action: 2 Client send NewOrderSingle (PASSED) [0.007s] : Sending of 'NewOrderSingle' message initiated by client to server according to the dictionary s...
+ Action: 3 Client receive ExecutionReport (FAILED) [0.104s] : Receiving of 'ExecutionReport' message initiated by server to client with status 'New'
+ Action: 4 GetCheckPoint (PASSED) [0.009s] : Get checkpoint to limit search scope by only Messages received after this checkpoint.
+ Action: 5 Client send NewOrderSingle (PASSED) [0.002s] : Sending of 'NewOrderSingle' message initiated by client to server according to the dictionary s...
+ Action: 6 Client receive ExecutionReport (PASSED) [0.012s] : Receiving of 'ExecutionReport' message initiated by server to client with status 'New')
+ Action: 7 Client receive ExecutionReport (FAILED) [0.001s] : Receiving of 'ExecutionReport' message initiated by server to client with status 'Filled'
+ Action: 8 Client receive ExecutionReport (PASSED) [0.04s] : Receiving of 'ExecutionReport' message initiated by server to client with status 'Filled')

For the failed receive action the Comparison table with Expected and Actual values will be displayed as below.



− Comparison Table

Comparison Table

Collapse Groups

Name	Expected	Actual	Status
− header			
BeginString	null	FIXT.1.1	NA
BodyLength	null	298	NA
MsgType	EXECUTION_REPORT(8)	EXECUTION_REPORT(8)	PASSED
SenderCompID	null	Server	NA
TargetCompID	null	Client	NA
MsgSeqNum	null	6	NA
SendingTime	null	2019-07-29T12:51:26.408	NA
ExecID	*	15644030183080000025	PASSED
ClOrdID	15644014566320000029	15644014566320000029	PASSED
OrderID	*	15644030183080000026	PASSED
ExecType	NEW(0)	NEW(0)	PASSED
OrdStatus	NEW(0)	NEW(0)	PASSED
LeavesQty	1000	1000	PASSED
CumQty	0	0	PASSED
SecurityID	Instrument	Instrument	PASSED
SecurityIDSource	EXCHANGE_SYMBOL(8)	EXCHANGE_SYMBOL(8)	PASSED
+ TradingParty			
Account	DemoUser	DemoUser	PASSED
OrdType	LIMIT(2)	LIMIT(2)	PASSED
TimeInForce	GOOD_TILL_CROSSING(5)	DAY(0)	FAILED
Side	BUY(1)	BUY(1)	PASSED
OrderQty	1000	1000	PASSED
DisplayQty	null	1000	NA
Price	105	105	PASSED
AccountType	null	HOUSE(3)	NA
OrderCapacity	null	AGENCY(A)	NA
TransactTime	*	2019-07-29T12:51:26.407	PASSED
− trailer			
CheckSum	null	091	NA