Connector Setting Guide

PlanetCross Connector

This manual is currently reference document. Officially, please check the Japanese one.

PlanetCross Connector can easily create X-Road Services from SQL. It works with MySQL/MariaDB, PostgreSQL, SQLite, Oracle, IBM DB2 and Microsoft SQL Server.

Contents

1. Introduction

1.1 How it works

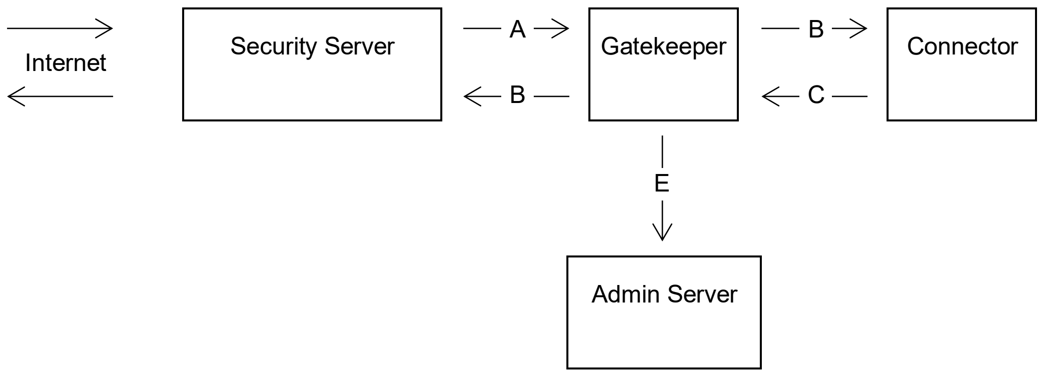


Fig.1 Connector related network

Connector runs two webservers, one SOAP server to handle SOAP requests and Admin server to provide UI for service configuration.

Security Server sends SOAP requests to Connector (A),
Connector sends SQL query to database (B) and receives rows (C),
generates a SOAP response and sends it back to the Security Server (D).

Admin server provides a web UI for configuring services, which X-Road service administrators use (E). After service configuration is done, Connector saves the service configuration in a file, generates a WSDL URL which Security Servers can fetch (A) to publish the service.

2. Overview how to use Connector

Connector setting overview is following.

Pre-Preparing Database

1. Prepare user and password to access from Connector.

Operate Connector

1. User sets connection parameters to connect with Database.
2. User sets data services.
3. Connector creates WSDL file automatically.

Operate Security Server

1. User register CLIENTS to Security Server.　(Security Server Clients => ADD CLIENTS)
2. User request CLIENTS registration to Planetway staff.
   User sends information: Member Name, Member Class, Member Code, Subsystem Code.
3. User clicks services tab.
4. User register services: copy the WSDL URL from connector and paste it to Security Server.
5. Clicks services and set Access rights.
   Clicks ADD SUBJECTS button and search the object, add the Access rights.

3. Connector configuration

Connector itself is installed in /opt/planetx/connector/ folder.

Edit `/etc/planetx/connector/main-conf.json` to configure the webservers' behaviour.

Default configration looks like this:

```
{
  "conf": "/etc/planetx/connector/services.json",

  "services":
    { "port": 8003
    , "host": "0.0.0.0"
    , "schema": "http"
    , "hostname" : null
    , "log": {
         "access"     : "/var/log/planetx/connector/pw-access.log"
    ,    "error"      : "/var/log/planetx/connector/pw-error.log"
    ,    "sql"        : "/var/log/planetx/connector/pw-sql.log"
    ,    "soapAll"    : ""
    ,    "soapHeader" : "/var/log/planetx/connector/pw-soapheader.log"
      }
    },

  "admin":
    { "port": 8002
    , "host": "127.0.0.1"
    }
}
```

Here are descriptions of config file.

- "conf": Full path to a json file which Admin server writes and SOAP server reads to handle SOAP requests.
- "services": Object to configure SOAP server.
- "services"."port": On which port it listens on.
- "services"."schema": Use either http or https.
- "services"."host": TCP host to bind (0.0.0.0 for public access).
- "services"."hostname": Hostname of the generated WSDL URL and the URL written inside the WSDL. If `null` or missing system configuration is used.
- "services"."log": Object to configure log behaviour.
- "services"."log"."access": SOAP server's HTTP access logs.
- "services"."log"."error": Error logs.
- "services"."log"."sql": SQL queriy logs.
- "services"."log"."soapAll": Whole SOAP request and response are logged to this file. If empty, logging is disabled.

- "services"."log"."soapHeader": SOAP header of request and response are logged to this file.
- "admin": Object to configure Admin server.
- "admin"."port": On which port it listens on.
- "admin"."host": TCP host to bind.

Admin server listens on localhost by default.
Change admin.host to the IP address of the network interface which you will use to access the admin server.

3.1 Apply configuration changes

Check current status of Connector.

```
$ systemctl status planetx-connector.service
 planetx-Connector.service - PlanetX Connector
   Loaded: loaded (/lib/systemd/system/planetx-connector.service; enabled;
vendor preset: enabled)
   Active: active (running) since Thu 2018-08-02 09:08:20 UTC; 34s ago
 Main PID: 1220 (node)
    Tasks: 10
   Memory: 50.9M
      CPU: 333ms
   CGroup: /system.slice/planetx-connector.service
           └─1220 /opt/planetx/connector/node/bin/node
/opt/planetx/connector/bin/planetx-Connector.js

Aug 02 09:08:20 ip-xxx-xxx-xxx-xxx systemd[1]: Started PlanetX Connector.
Aug 02 09:08:21 ip-xxx-xxx-xxx-xxx node[1220]: UI service running at ...
Aug 02 09:08:21 ip-xxx-xxx-xxx-xxx node[1220]: SOAP service running at ...
Aug 02 09:08:21 ip-xxx-xxx-xxx-xxx node[1220]: Services are not defined
```

Restart Connector to apply changes to the config file.

4. Services configuration

Open Admin server UI to configure services.
Admin server UI is at http://{host}:{admin.port}/ where {host} is the hostname or IP address which your browser can reach this server, and {admin.port} is the port number in configuration file.

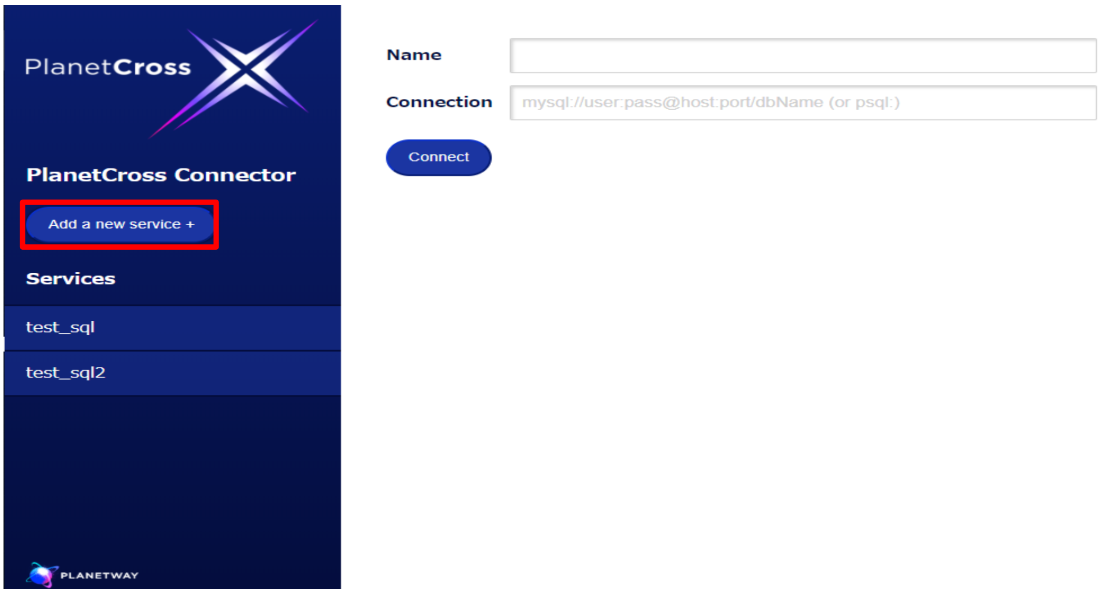
Admin server UI supports following browsers.

- Latest Safari
- Latest Google Chrome

NB! Internet Explorer is not supported.

4.1 Adding services

Press the "Add a new service" button on the left to start configuring a new service.



4.1.1 Name

The Name field is going to be used in the WSDL URL, and can only consist of following characters in single bytes (半角).

- Alphabets in uppercase or lowercase
- Numbers
- _ Underscores
- - Dashes, if not the first character

Name must be unique within a single Connector installation.

4.1.2 Connection string

Connection string should be one of:

- mysql://user:passwd@host:port/databasename
- psql://user:passwd@host:port/databasename
- oracle://user:passwd@host:port/databasename
- db2://user:passwd@host:port/databasename
- mssql://user:passwd@host:port/databasename
- sqlite:///filename

You can omit port if it is set to default.

For more advanced connection parameters use URL arguments. They are passed directly to database connection driver. Drivers documentations:

- mysql: https://www.npmjs.com/package/mysql
- postgresql: https://www.npmjs.com/package/pg
- oracle: https://www.npmjs.com/package/oracledb
- db2: https://www.npmjs.com/package/ibm_db
- mssql: https://www.npmjs.com/package/mssql
- sqlite: https://www.npmjs.com/package/sqlite3

For example, to configure pooling use following connection string:

- mysql://user:passwd@host:port/databasename?connectionLimit=10&acquireTimeout=2000
- psql://user:passwd@host:port/databasename?max=10&idleTimeoutMillis=1000
- oracle://user:passwd@host:port/databasename?poolMax=10

If there is subobject in connection, use dot to create subobject. Example case is to connect using TLS. URL to use to connect to mysql in AWS goes like this: mysql://root:pw@yourhost.amazonaws.com/db?ssl.ca=/home/me/rds-combined-ca-bundle.pem&connectionLimit=10 (read more about AWS at https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_MySQL.html#MySQL.Concepts.SSLSupport). And yes, you have to download rds-combined-ca-bundle.pem to your server and give full path for that file (files are read automatically).

Warning: Make sure that the database user has the minimal privileges to issue the SQL query you want.

4.1.2.1 Percent-encoding for connection string

The characters in the connection string should be percent encoded if necessary, according to the RFC3986.

Reserved characters according to RFC3986 should be percent encoded as follows.

Reserved characters	!	#	$	%	&	'	(	)	*	+	,	/	:	;	=	?	@	[	]
Percent encoded	%21	%23	%24	%25	%26	%27	%28	%29	%2A	%2B	%2C	%2F	%3A	%3B	%3D	%3F	%40	%5B	%5D

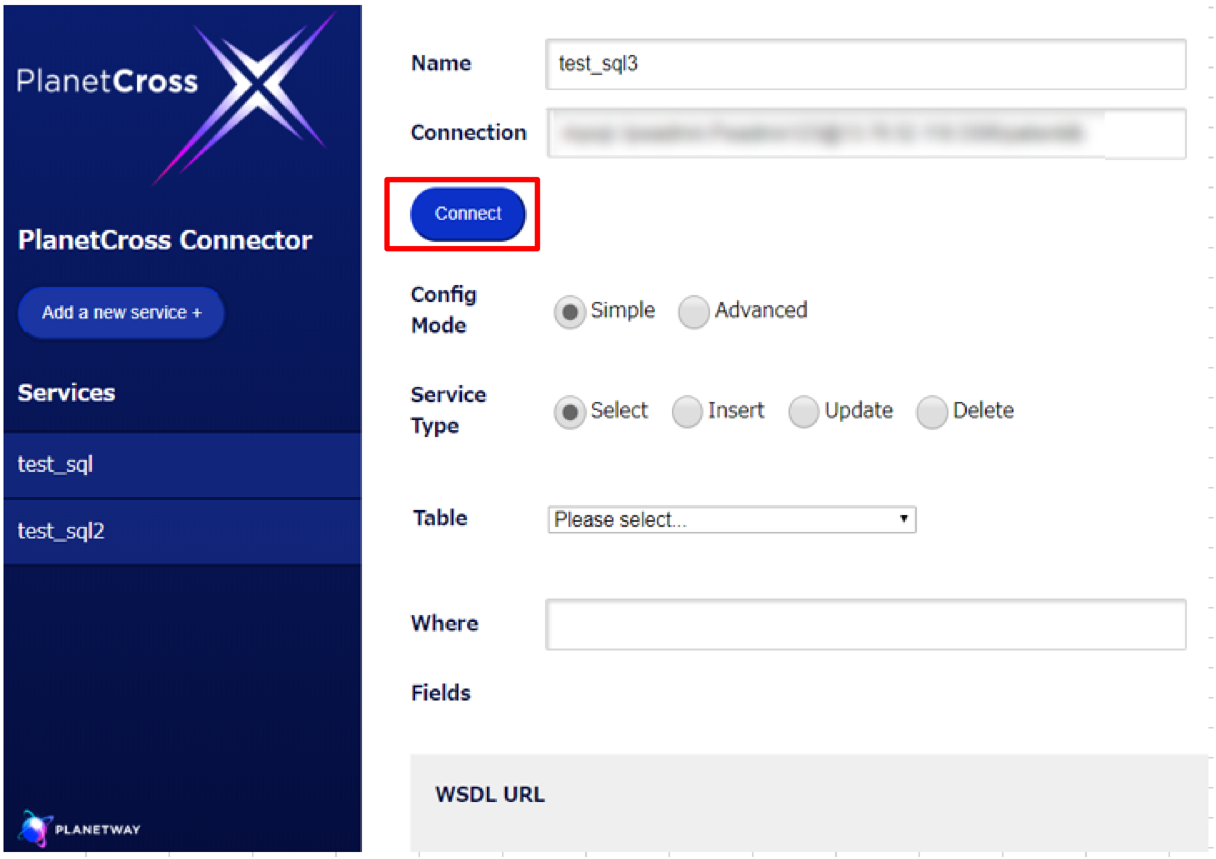
For example:

```
mysql://A1b}_.-%23:1Abc%40.-_~%28%29@mysql:3306/databasename
```

In above connection string, username is A1b}_.-# and password is 1Abc@.-_~().

Click [Connect] button after input Name and Connection.

When user will be connection success, user can configure following.



Name: test_sql3

Connection: [redacted]

Config Mode: [✓ Simple] Advanced

Service Type: [✓ Select] Insert Update Delete

Table: Please select...

Where:

Fields

4.1.3 Simple mode and advanced mode

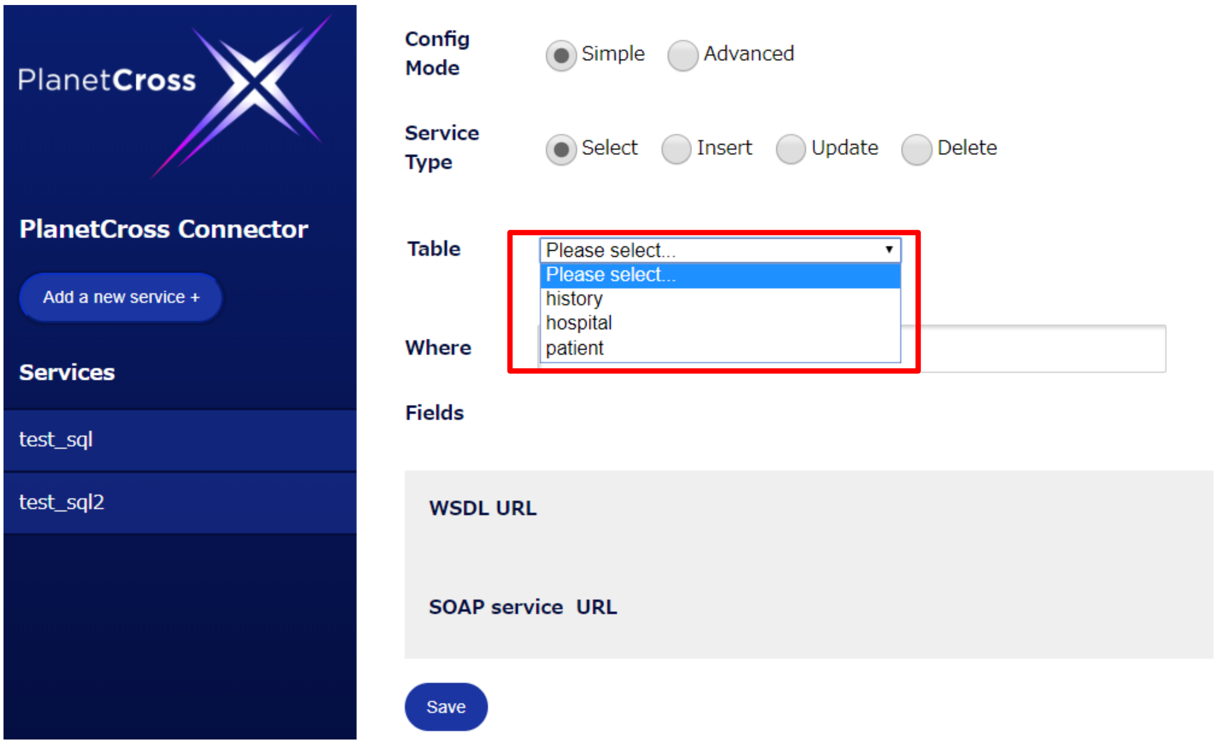
There are two configuration modes: Simple and Advanced.

4.1.3.1 Simple mode

Simple mode shows you tables and fields loaded from the database.
You can choose which fields are returned in response for the service.

You can select “Service Type”that you want to use.

When click “Table, pulldown that connected database table will be displayed.



Config Mode: [✓ Simple] Advanced

Service Type: [✓ Select] Insert Update Delete

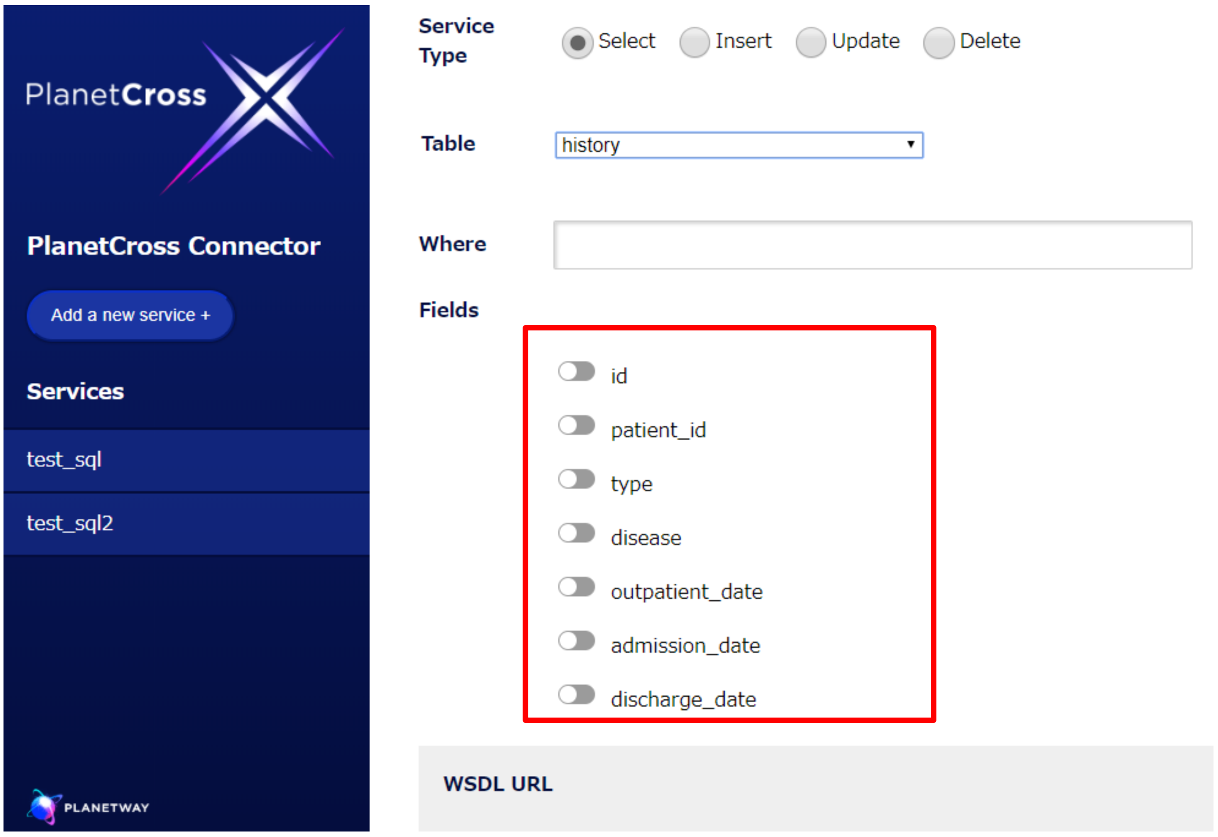
Table: Please select...
- Please select...
- history
- hospital
- patient

Where

Fields

WSDL URL

SOAP service URL



Service Type: [✓ Select] Insert Update Delete

Table: history

Where

Fields:
- id
- patient_id
- type
- disease
- outpatient_date
- admission_date
- discharge_date

WSDL URL

Where condition has to be written manually.
Input the where condition part of the SQL, for example as:

```
(where) patient_id = $patient_id1
```

Following operators are supported in the Simple mode where condition.

- AND, OR
- =, >, >=, <, <=

Simple mode is only supported in **MySQL**, **PostgreSQL** and **SQLite**. For other databases please use Advanced mode.

4.1.3.2 Advanced mode

In advanced mode you write the whole SQL query.

When you use variable, you need to input “$” mark.

LIKE Operator is supported in advanced mode only.

When you use LIKE query, please use the contact function.

About the syntax of the concat function, check the specifications of each database.

A description example in MySQL and postgreSQL is following,

MySQL

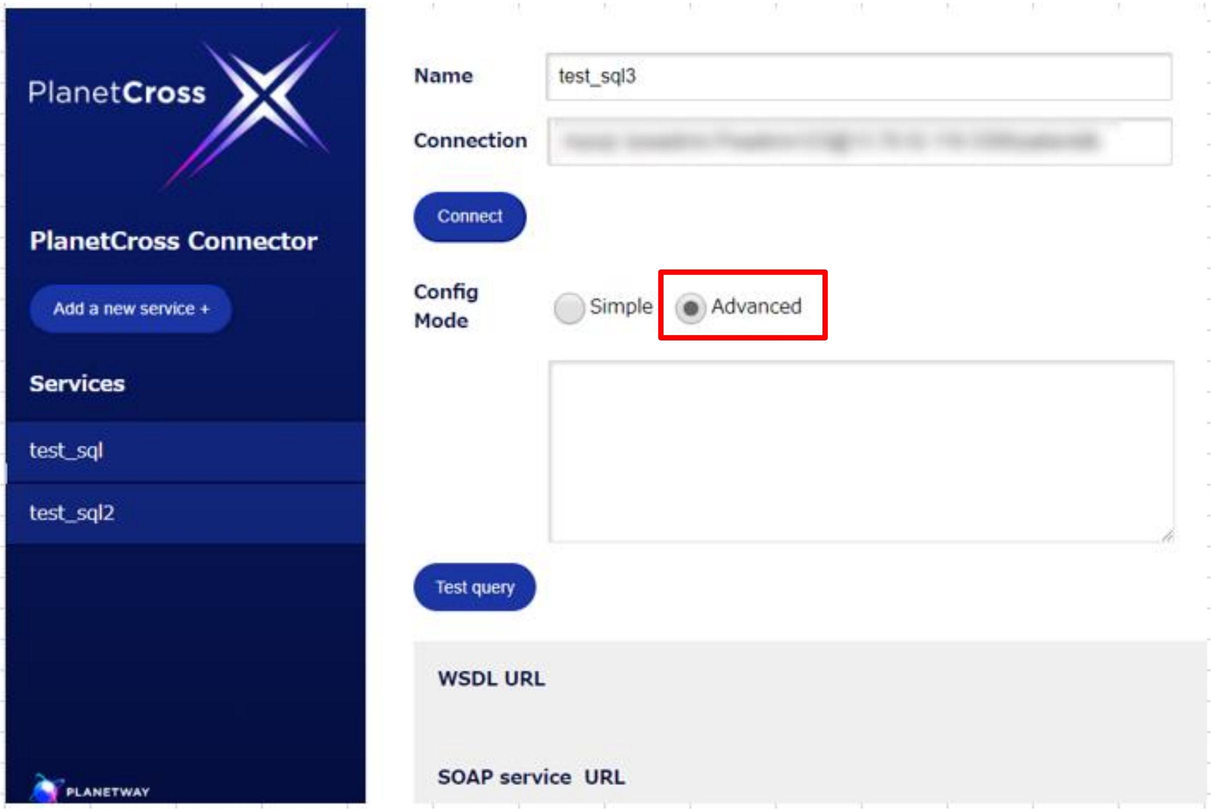
```
SELECT * FROM users WHERE username LIKE CONCAT($name, '%');
```

PostgreSQL

```
SELECT * FROM users WHERE username LIKE CONCAT($name::varchar, '%');
```

NB: Not supported following syntax about LIKE query.

```
SELECT * FROM users WHERE username LIKE '$name%';
```



Name: test_sql3

Connection: [redacted]

Config Mode: Simple [✓ Advanced]

4.1.3.3 Common between simple and advanced mode

In both modes **dollar symbol** is used to mark SOAP input parameters.
For example if you write `select * from foo where id = $id_input`,
then `id_input` automatically becomes the input parameter.

Be aware variables names are used as binding parameters in SQL and for Oracle you may not use reserved words (list of reserved words can be found from Oracle documentation: https://docs.oracle.com/cd/B19306_01/em.102/b40103/app_oracle_reserved_words.htm)

Queries for `insert`, `update` and `delete` can also be used. Field names are used as input parameter and affected rows count is returned.

4.1.3.4 Limited support for stored procedures

In advanced mode, Connector supports PostgreSQL stored procedures when the SELECT query is written in a particular way that returns each output parameters separately.

When we have following table and stored procedure,

```
-- Create table
CREATE TABLE person (
    id   integer,
    name varchar(100),
    sex  char(1)
);
```

```
-- Insert row
INSERT INTO person values(1,'suzuki','A');

-- Create stored procedure
CREATE TYPE PersonEntity AS (id integer, name varchar(100), sex char(1));

CREATE OR REPLACE FUNCTION SampleStoredProcedure(argId integer)
RETURNS PersonEntity AS $$
DECLARE
   returnValue RECORD;
BEGIN
   SELECT id, name, sex FROM Person WHERE id = argId
   INTO returnValue;
   RETURN returnValue;
END;
$$ LANGUAGE PLpgSQL;
```

Following SELECT SQL works in advanced mode and returns `id`, `name` and `sex` parameters separately in SOAP response.

```
SELECT * FROM SampleStoredProcedure($id)
```

Be aware that stored procedure support is limited and we do NOT plan to fully support all databases, all languages and all sorts of various ways you can write stored procedures.

4.1.3.5 Column names

The database column names define the SOAP response body XML element names, therefore the characters you can use in column names are limited, as follows.

All characters defined in Extensible Markup Language (XML) 1.0 (Fourth Edition) 2.3 Common Syntactic Constructs as "Name"
except `:` and `-` and `.` are allowed as column names.

Appendix B Character Classes is also worth to review.

If you use a column name that doesn't fit these criteria,
you can use the `AS` keyword in advanced mode as `SELECT column-name-with-dash AS column_name_without_dash FROM table WHERE ...` to rename and to avoid this problem.

Also when you use SQL functions, assign a valid name using the `AS` keyword,
because `(` and `)` are invalid.
ex: `SELECT COUNT(*) as c FROM ...`.

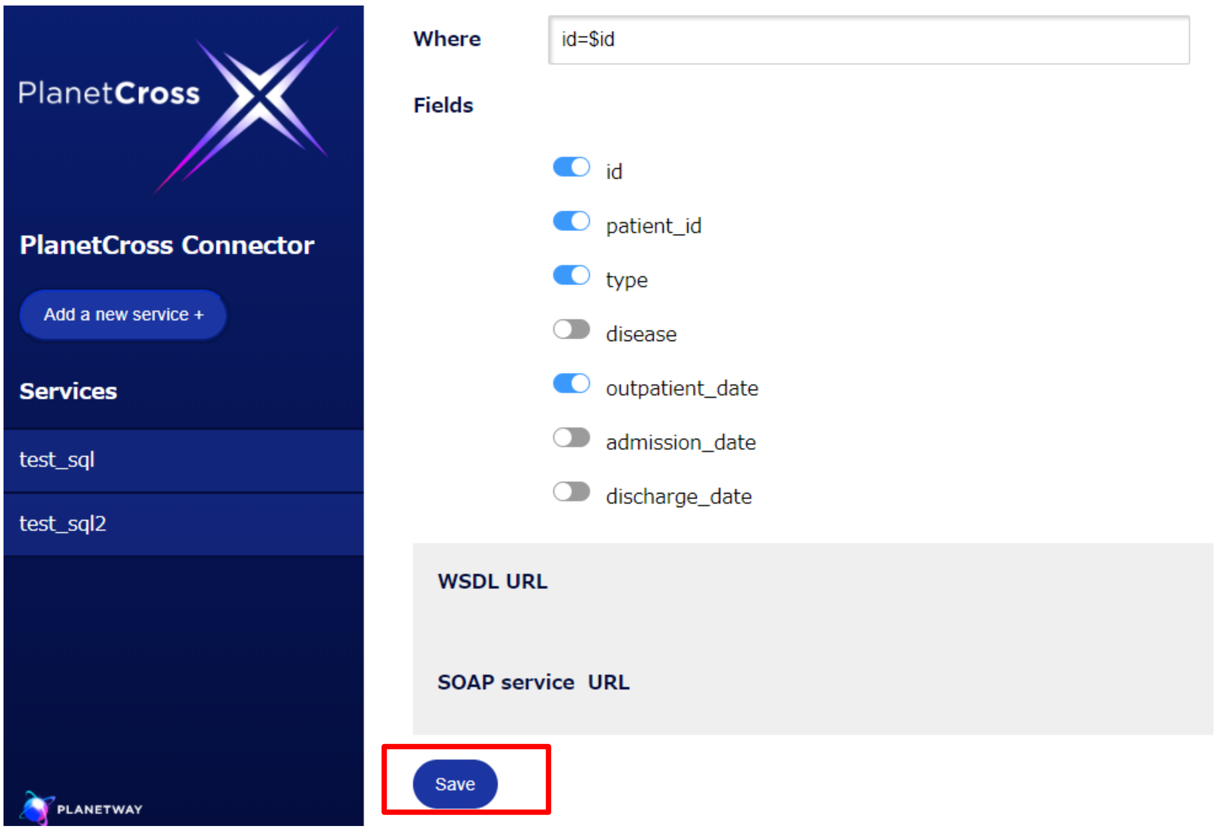
Services that access tables with unallowed column names cannot be saved.

We have some notes, please check "APPENDIX A" also.

4.1.4 Save

You must save service prior to using the service. After save you can see wsdl URL and you are ready to use the service from the Security Server.

Press the "Save" button to save the service.



Where: id=$id

Fields

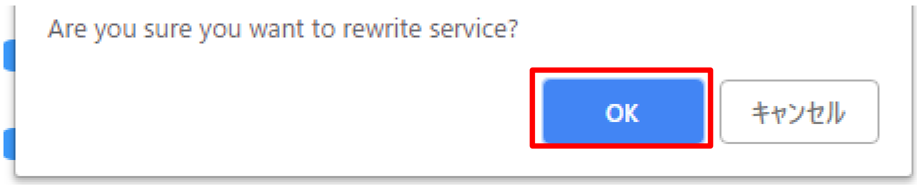
- [✓] id
- [✓] patient_id
- [✓] type
- [ ] disease
- [✓] outpatient_date
- [ ] admission_date
- [ ] discharge_date

WSDL URL

SOAP service URL

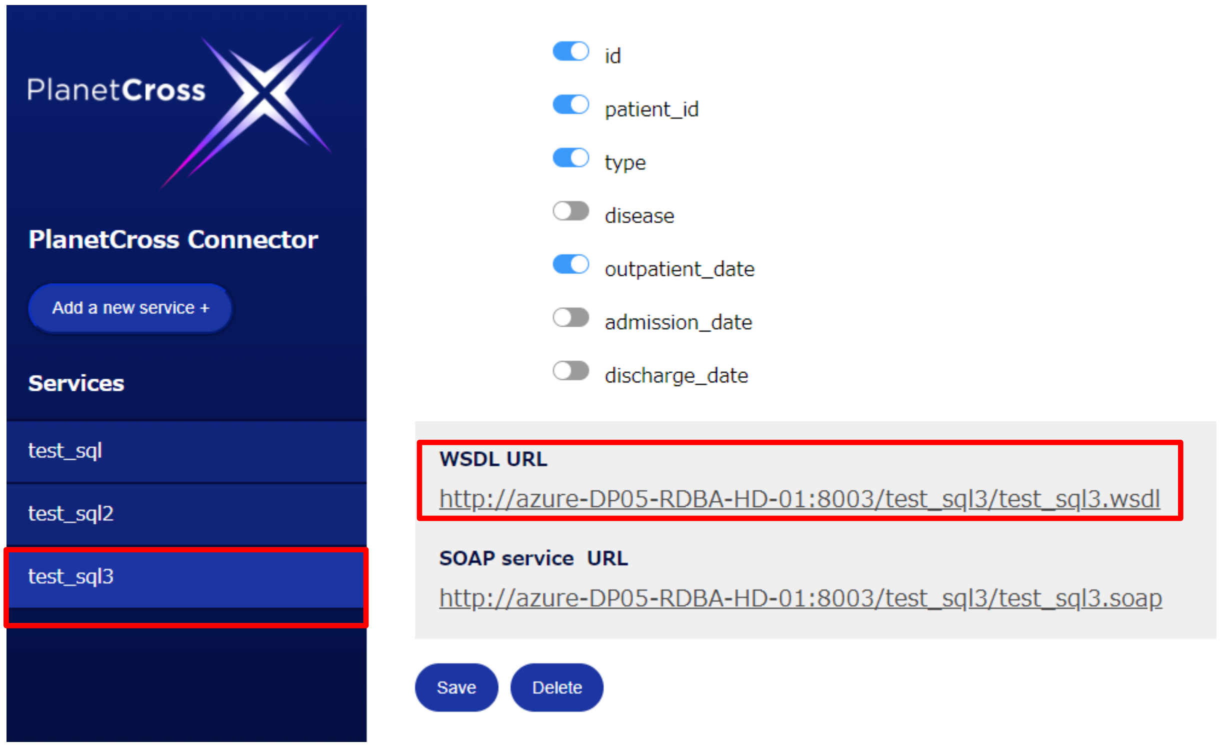
After click the "Save" button, it will be displayed "OK" button.

Please click it.



Are you sure you want to rewrite service?

After that WSDL URL will be displayed and added in Services field .



id
patient_id
type
disease
outpatient_date
admission_date
discharge_date

WSDL URL
http://azure-DP05-RDBA-HD-01:8003/test_sql3/test_sql3.wsdl

SOAP service URL
http://azure-DP05-RDBA-HD-01:8003/test_sql3/test_sql3.soap

4.2 Updating service

Connector doesn't support service versioning. If you update the service definition on Admin UI and press "Save", the previous service will be overwritten and new service will be available immediately.

To avoid this, for example when you're already providing service to a client, change the name and then press the "Save" button to save the service as a different name. The original service will be unchanged.

4.3 Deleting service

Click the "Delete" button to delete the service. After you confirm, the service will be deleted immediately.

4.4 Support consent function

`$__targetUserId` is a special variable you can use in the SQL that comes from the SOAP header `targetUserId` element. `targetUserId` is defined in `http://xsd.planetcross.net/planetcross.xsd` xsd. For example, if you have a service made in advanced mode with query:

```
SELECT * FROM users WHERE id = $__targetUserId;
```

Following SOAP request:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   ... snip ...
   xmlns:px="http://xsd.planetcross.net/planetcross.xsd"
   xmlns:xxforms="http://orbeon.org/oxf/xml/xforms">
  <SOAP-ENV:Header>
     <xrd:protocolVersion>4.0</xrd:protocolVersion>
     <xrd:id>c87e72c77f97581af1a7108b4f9d96b5e860a7f6</xrd:id>
     <xrd:userId>JP111111111</xrd:userId>
     <px:targetUserId>12345</px:targetUserId> <!-- this becomes the
$__targetUserId variable -->
     <xrd:issue />
     <xrd:service iden:objectType="SERVICE">
       ... snip ...
     </xrd:service>
     <xrd:client iden:objectType="SUBSYSTEM">
       ... snip ...
     </xrd:client>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
  ... snip ...
```

will result in following SQL:

```
SELECT * FROM users WHERE id = 12345;
```

In Simple mode you can also write where condition as `id = $__targetUserId`.

Appendix A Notes

This appendix is described notes when you setup the Connector.

A.1 When you use MySQL, you can’t set variable name same as column name.

[Contents]

When you use MySQL, you can’t set variable name same as column name.

[Workaround]

Please, refer following.

For example:

NG：id=$id AND patient_id=$patient_id

OK：id=$id AND patient_id=$patient_id1

A.2 When you use PostgreSQL, there is possibility that you can’t get a data in case of setting full-width digit with column name.

[Contents]

When you use PostgreSQL, there is possibility that you can’t get a data in case of setting full-width digit with column name.

[Workaround]

When you add “AS” option, you can get the data as the half-width digit.

For example:

select code, title, "２byte" as "2byte", character_varying_256, varchar_256, char_256, text from kanji where (code = $code);

Revision History

Version	Date	Details
V1.4	11/02/2019	Publish first edition.