

# Using The Postman® & SoapUI® Application with Sage X3

Raheel Khan

Sage



# Contents

## SoapUI® Application

- Sage X3 Prerequisites for SOAP Web Services
- Introduction to the SoapUI® Application
- Installation of SoapUI®
- The Sage X3 SOAP WSDL
- How to send a request to X3
  - Loading the X3 WSDL
  - Importing & Saving Projects
- Load Testing with SoapUI®
  - Creating a test case
  - Defining data and assertions
  - Configuration

## Postman Application

- Sage X3 and Graph QL API
- Introduction to the Postman Application
- Installation of Postman
- How to send a Request to X3 using Postman
  - Authentication
  - SOAP Request
  - Graph QL Request
- Tips
  - Request Tips
  - Troubleshooting when no WS response
  - Graph QL Network inspector

# Sage X3 Prerequisites for SOAP

# Sage X3 Prerequisites for SOAP

SOAP Web pool needs to be configured and channels available so that the web service is available externally.

All > Administration > Administration > Web services

Pool configuration: WSSEED

Information X3 connection Channels

**Information**

Alias \*  
WSSEED  Auto start  Stopped manually

**X3 connection**

Endpoint \* SEED X3 runtime tags Locale \* English (United Kingdom)  
Endpoints describe services locations Comma-separated tags can be used to select your preferred X... Locale preferences  
User \* admin  
Super administrator

**Channels**

Maximum size \* 4 Initialization size \* 2 Unused lifetime (mn) 20 Lifetime (mn) 720

Status

Host	Port	Process	PID	Status	User login	Locale	Last action	Action By	WS queue	WS count
No data to display										

- Alias that identifies the pool
- Auto start to start the pool automatically when Sage X3 starts
- Default startup language
- Default startup Sage X3 user code
- Max size (maximum number of connections)
- Initial size (number of connections at startup).

SOAP pool status

Clear filter

Alias	Host	Port	Process	PID	Status	User login	Locale	Last action	Action By	WS queue
WSSEED	X3ERP12SQLVM	0	W0	4260	Started	admin	en-GB	Auto start		

X3 session	X3 PID	Status	X3 host	X3 port	X3 user login	X3 locale code	Solution	Folder	Web Service	Log	Creation date	Last access	Expiration date	WS count
42832	8784	Available	x3erp12sqlvm	20100	admin	en-GB	X3V12	SEED			2024-04-17T12:58:40.548Z	2024-04-17T12:58:40.548Z	2024-04-17T13:18:49.615Z	0
42842	2708	Available	x3erp12sqlvm	20100	admin	en-GB	X3V12	SEED			2024-04-17T12:58:50.042Z	2024-04-17T12:58:50.042Z	2024-04-17T13:18:58.466Z	0

# Sage X3 Prerequisites for SOAP

**Web service needs to be published** for the Web services a simple function exists under Scripts in the Development > Script dictionary > Web Services GESAWE to transform a subprogram or object into a Web service with a public name.

The screenshot shows the Sage X3 Web services configuration interface. The 'Definition' tab is active, showing the 'Object' field set to 'BPC' and the 'Program' field set to 'WJZBPC'. The 'Information' section shows the 'Published on' date as 20240320170106. The 'Mapping' section shows a list of objects with checkboxes and codes.

A file explorer window shows the directory path: D:\Sage\X3V12\folders\SEED\TRT. The files listed are WJZBPC.adx and WJZBPC.src.

A file explorer window shows the directory path: This PC > Sage (D:) > Sage > X3V12 > folders > X3\_PUB > SEED > GEN > ALL > WEBS. The files listed are ZBPC.xml (XML Document, 105 KB) and ZBPC.xsd (XSD File, 753 KB).

A green arrow points from the 'WJZBPC.adx' file in the file explorer to the 'WJZBPC' program in the Sage X3 interface.

A XML description, stored in the following directory on the applications server: X3\_PUB/FolderName/GEN/ALL/WEBS is generated

# Sage X3 Prerequisites for SOAP

A user needs to be configured for authentication

This screenshot shows the 'Users' configuration page for the 'admin' user. The 'Login' tab is active, displaying a table with user details. The 'Active' status is checked, and the authentication method is 'Standard'. There are 'x' marks next to 'Password never expires' and 'Require password change'.

Login	Active	Authentication	New password	Password never expires	Require password change
admin	✓	Standard	*****	x	x

Factory Owner: SAGE  
LDAP instance for synchronization: SAGE  
LDAP directory: SAGE

Syracuse user setup

This screenshot shows the 'Information' and 'Administration' tabs for the 'admin' user. The 'Information' tab displays the user's title as 'Mr' and email as 'admin@sagex3.com'. The 'Administration' tab shows links for 'Groups' (Super administrators) and 'Endpoints login'.

This screenshot shows the 'General' settings for the 'System Administrator' user. A blue arrow points to the 'Web services connection' checkbox, which is checked. Other settings include 'X3 connection' (checked), 'No directory check' (unchecked), and various profile and contact information fields.

Sage X3 Classic user setup with web services connection allowed

# SoapUI® Application

# What is the SoapUI® Application

SoapUI® by SmartBear is a widely used open-source tool for testing SOAP (Simple Object Access Protocol) and REST (Representational State Transfer) web services. It was first launched in 2006.

## Key Features:

**Versions:** SoapUI is available in both open-source and commercial versions. The open-source version offers robust features for most testing needs, while the commercial version provides additional enterprise-level features and support.

**Supported Technologies:** SoapUI supports various technologies, including SOAP, REST, HTTP, JMS, AMF, JDBC, and more, making it versatile for testing a wide range of APIs and web services.

•**Functional Testing:** SoapUI allows users to create and execute functional tests for SOAP and REST APIs.

•**Automated Testing:** It supports automated testing with features like assertions, data-driven testing, and scriptable testing scenarios.

•**Load Testing:** It provides load testing capabilities to assess the performance and scalability of web services under various conditions.

•**Mocking:** SoapUI enables the creation of mock services, allowing developers to simulate APIs and test client applications without accessing the service.

•**Security Testing:** SoapUI supports security testing by configuring security protocols and conducting vulnerability assessments.



# SoapUI® With Sage X3

SoapUI can be used with Sage X3

1. Using a Projects in SoapUI to save your requests
2. Testing SOAP web services outside side of Sage X3
3. Testing payloads that may not be suitable for the internal tester
4. Automated testing
5. Load testing

# Installing SoapUI® Application

# SoapUI® Application Download

SoapUI application can be downloaded from <https://www.SoapUI.org/downloads/SoapUI/>

## Download the Best API Testing Tool for your Needs

Try ReadyAPI for advanced API testing (security, load, & virtualization)  
or download SoapUI to get started with the fundamentals



### ReadyAPI

Get the most advanced functional testing tool for  
REST, SOAP and GraphQL APIs.

Download ReadyAPI



### SoapUI Open Source

Get the open source version of the most widely  
used API testing tool in the world.

Download SoapUI Open Source

There are two versions available for Windows,  
Linux, MacOS

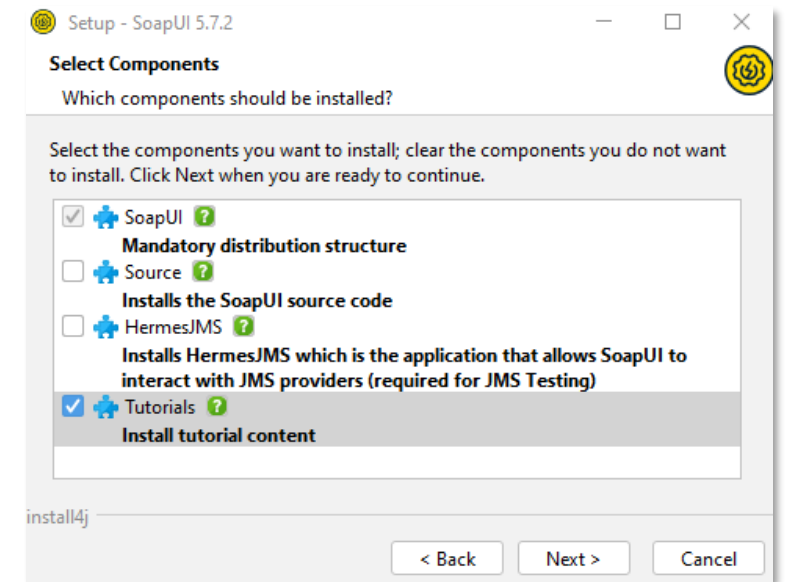
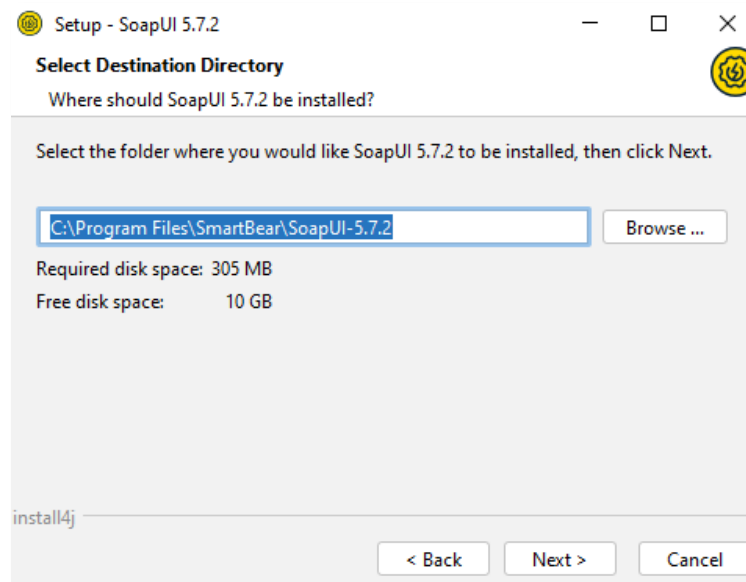
Ready API is the commercial version that has  
extra functionality

- Improved interface
- Team collaboration features (Jira, GitHub, Slack etc)
- More advanced testing features
- Pre-built reporting features



SoapUI-x64-5.7.2.exe  
SoapUI 5.7.2  
SmartBear Software

# Installing SoapUI® Application



1. Execute the.exe file for Windows
2. Choose the installation directory
3. Choose components

# Installing SoapUI® Application

The image displays four sequential screenshots of the SoapUI 5.7.2 installation wizard, connected by green arrows indicating the flow of the process:

- Step 1: Tutorials location** - The user is prompted to select a target directory for SoapUI Tutorials. The path `C:\Users\X3admin\SoapUI-Tutorials` is entered.
- Step 2: Select Start Menu Folder** - The user is asked where to place the program's shortcuts. The folder `SmartBear\SoapUI 5.7.2` is selected from a list of options.
- Step 3: Select Additional Tasks** - The user is asked which additional tasks should be performed. The options `Create a desktop icon` and `Create shortcuts for all users` are checked.
- Step 4: Completing the SoapUI 5.7.2 Setup Wizard** - The wizard is finished. The user is informed that the application has been installed and can be launched by selecting the installed icons. The options `View the release notes` and `Run SoapUI 5.7.2` are checked. The `Finish` button is visible.

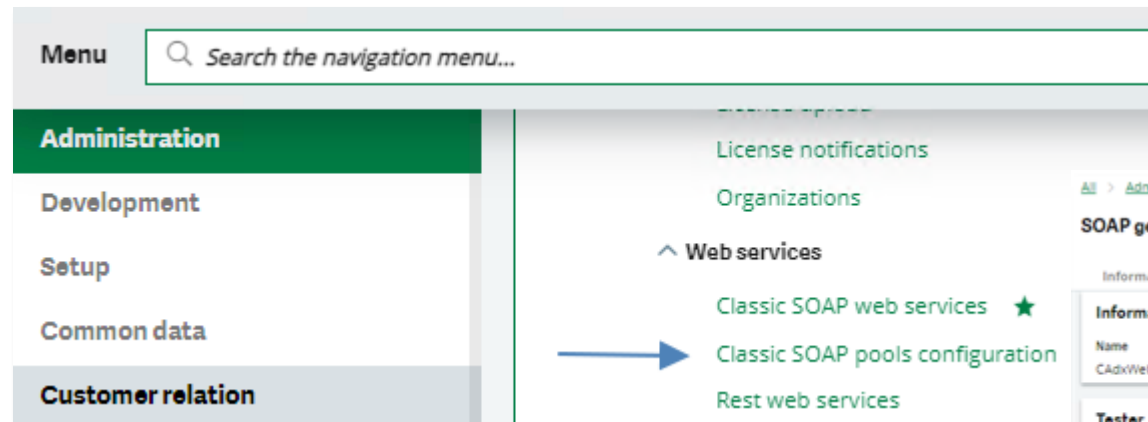
4. Specify the destination directory for tutorials
5. Select Windows start menu Folder
6. Create a desktop icon
7. Finish install

# Sage X3 SOAP WSDL

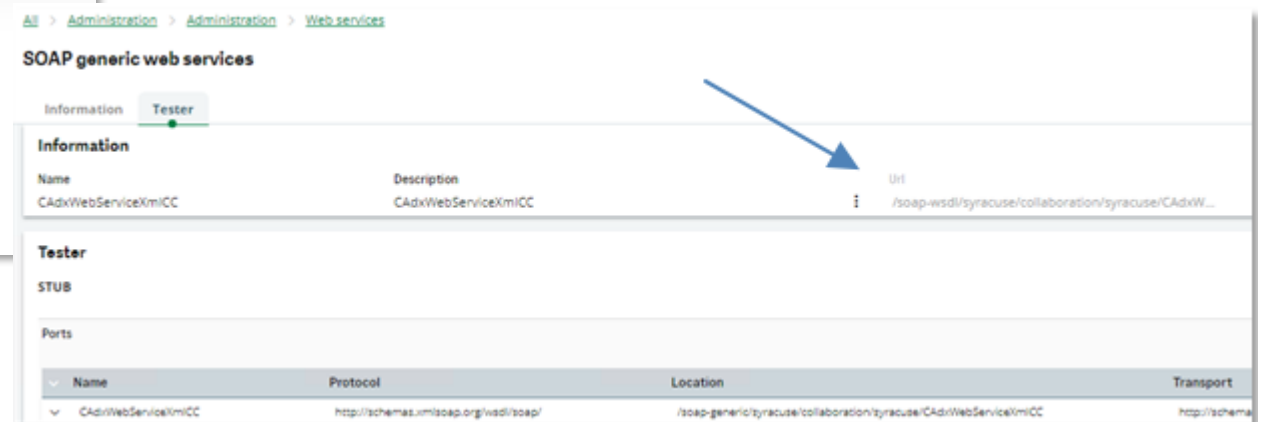
# The Sage X3 Soap WSDL

A SOAP (Simple Object Access Protocol) WSDL (Web Services Description Language) is a document written in XML that describes the functionalities offered by a SOAP-based web service.

WSDL defines the methods available, the parameters required for each method, the data types used, and the communication protocols (typically HTTP and HTTPS) supported by the web service. It acts as a contract between the service provider, in our case Sage X3, and the service consumer (External application), enabling communication between different systems and platforms.



Access the SOAP WSDL from the Classic SOAP pools configuration function



# The Sage X3 Soap WSDL

When accessing the URL you can see the XML definition for the WSDL, Sage also has an internal SOAP testing utility which can be accessed from the same function

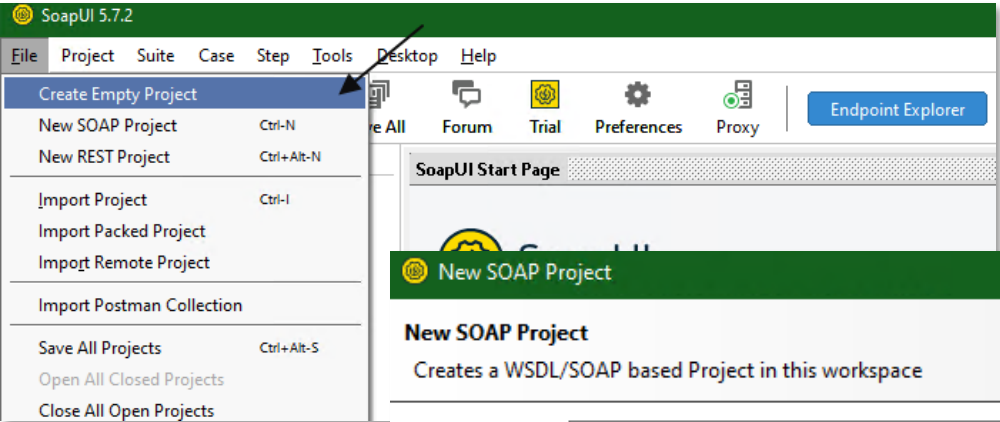
The screenshot shows the Sage X3 Administration console. The breadcrumb navigation is 'All > Administration > Administration > Web services'. The page title is 'SOAP generic web services'. There are two tabs: 'Information' and 'Tester', with 'Tester' selected. Below the tabs, there is a table with columns 'Name', 'Description', and 'Url'. The first row shows 'CAdxWebServiceXmlCC' with a description 'CAdxWebServiceXmlCC' and a URL starting with '/soap-wsdl/syracuse/collaboration/syracuse/CAdxW...'. A blue arrow points to the 'Url' column. Below this is a 'Ports' section with a table showing 'Name', 'Protocol', 'Location', and 'Transport'. The 'Operations' section is expanded, showing a table with 12 results. The table has columns 'Name', 'Description', 'SOAP action', and 'Style'. The operations listed include 'run', 'save', 'delete', 'read', 'query', 'getDescription', 'modify', 'actionObject', 'actionObjectKeys', 'getDataXmlSchem', 'insertLines', and 'deleteLines'.

**URL**  
<http://x3erpv12sqlvm:8124/soap-wsdl/syracuse/collaboration/syracuse/CAdxWebServiceXmlCC?wsdl>

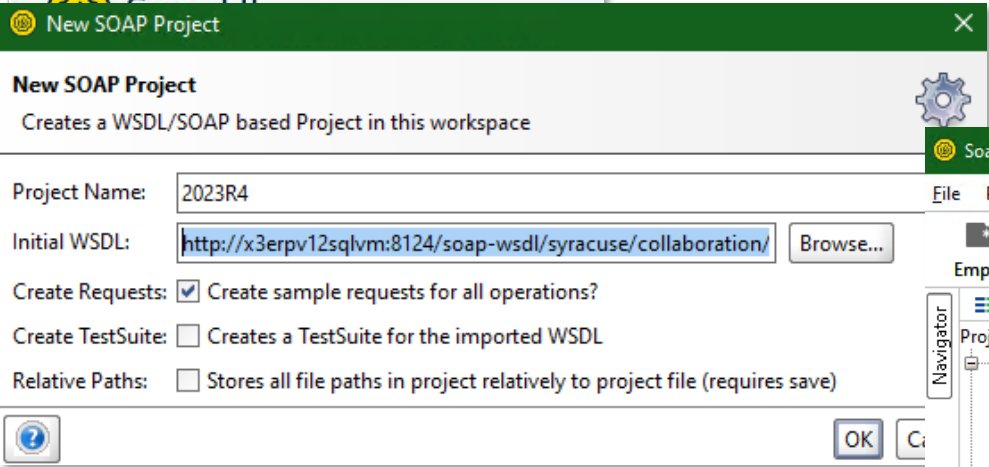
The screenshot shows a web browser displaying the WSDL XML definition for the CAdxWebServiceXmlCC service. The browser address bar shows the URL 'x3erpv12sqlvm:8124/soap-wsdl/syracuse/collaboration/syracuse/CAdxWebServiceXmlCC?wsdl'. The page title is 'Sage X3'. The browser tabs include 'Sage X3', 'ElasticCloud', 'X3 Handheld', 'GraphQL Explorer', and 'Imported'. The main content area shows the XML definition, which is a WSDL document. The XML starts with a root element 'definitions' and contains several complex types and messages. The document tree is shown below the text. The XML includes elements like 'sequence', 'complexType', 'restriction', and 'element'.



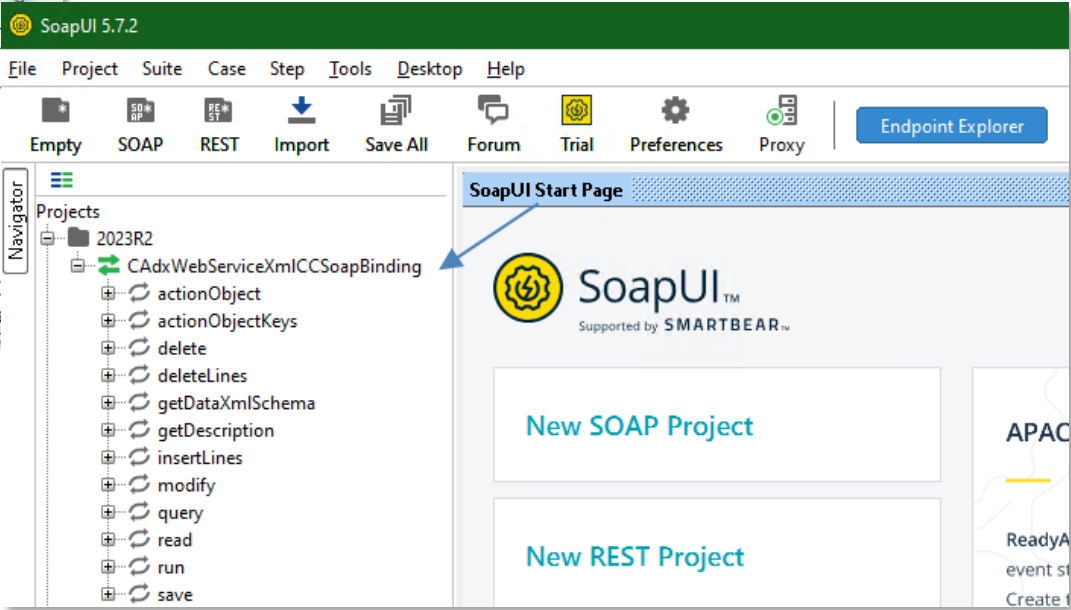
# Loading The Sage X3 SOAP WSDL



1. Use the File option to create a new empty project



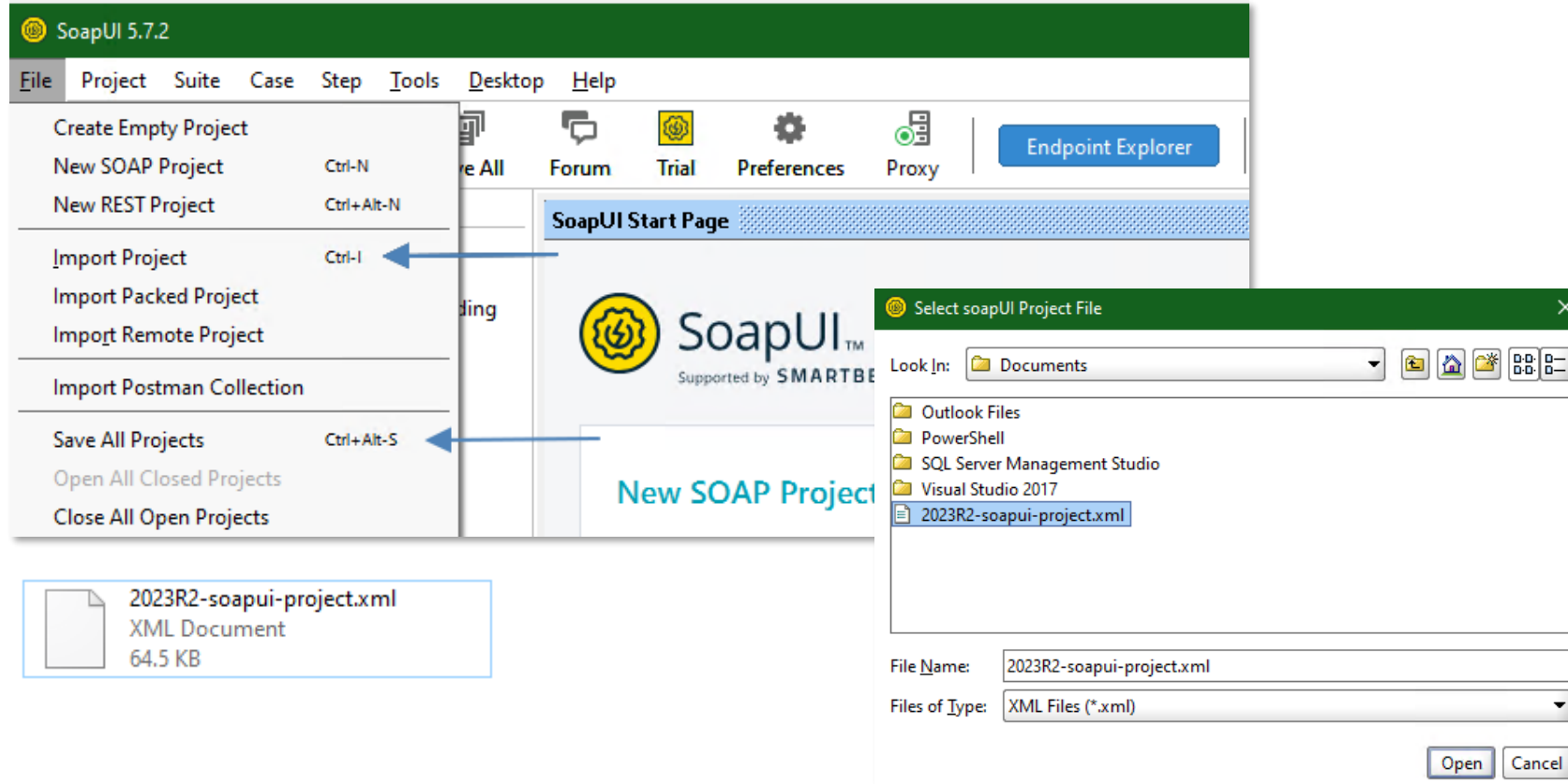
2. Enter a project name & Populate the WSDL URL



3. SoapUI® creates the new project and loads the WSDL methods

# Loading & Saving Your Project

SoapUI® Projects are saved in XML format



- When you open SoapUI®, existing projects are auto-loaded
- Use 'Import project' if you want to load an existing project from an XML file
- Use 'Save All' to save any changes to your project

Note that the project file will include any passwords/authentication details that are used in the project

# Sending Requests Using SoapUI®

# Sending SOAP Request To Sage X3

To send a SOAP request to Sage X3 using SoapUI®

- **Create your SOAP request** in the correct request category for example 'Read' or 'Query'

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:wss="http://www.adonix.com/WSS"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
  <soapenv:Header/>
  <soapenv:Body>
    <wss:read soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <callContext xsi:type="wss:CAdxCallContext">
        <codeLang xsi:type="xsd:string">BRI</codeLang>
        <poolAlias xsi:type="xsd:string">WSSEED</poolAlias>
        <poolId xsi:type="xsd:string"></poolId>
        <requestConfig xsi:type="xsd:string">adxwss.beautify=true</requestConfig>
      </callContext>
      <publicName xsi:type="xsd:string">ZSOH</publicName>
      <objectKeys xsi:type="wss:ArrayOfCAdxParamKeyValue" soapenc:arrayType="wss:CAdxParamKeyValue[]">
        <key>SOHNUM</key><value>SONGB0230015</value>
      </objectKeys>
    </wss:read>
  </soapenv:Body>
</soapenv:Envelope>
```

**Web service class/method** → `<wss:read>`

**Name of the web service pool** → `<poolAlias>`

**Web service public name** → `<publicName>`

**Language code (optional)** → `<codeLang>`

**Object key (for Read, Modify and Delete methods). It has two public properties: key (the Sage X3 field name or rank in the main index) and Value (the key value)** → `<key>SOHNUM</key><value>SONGB0230015</value>`

- **Set your authentication settings** to authenticate pre-emptively (pre-emptive authentication refers to the process of sending authentication credentials with every request without waiting for the server to challenge the client. Unlike the standard authentication process where the client waits for the server to respond with a challenge before sending the credentials)

# Sending SOAP Request To Sage X3

Submit request

Request SOH Read created under the 'read' method

The screenshot displays the SoapUI 5.7.2 interface. On the left, the 'Navigator' pane shows a tree structure of test cases and methods. The 'read' method is expanded, and 'SOH READ' is selected. Below this, the 'Request Properties' table is visible:

Property	Value
Name	SOH READ
Description	
Message Size	1032
Encoding	UTF-8
Endpoint	http://x3erpv12sq...
Redirects	true
me	admin
rd	*****
n	
itication Ty...	Preemptive
assword Ty...	
meToLive	
SSL Keystore	
Skip SOAP Action	false
Enable MTOM	false
Force MTOM	false
Inline Response A...	false
Expand MTOM At...	false

The main workspace shows the 'SOH READ' request configuration. The 'Authorization' dropdown is set to 'Basic'. The 'Username' field contains 'admin' and the 'Password' field contains a masked password. The 'Pre-emptive auth' radio button 'Authenticate pre-emptively' is selected. The 'Outgoing WSS' and 'Incoming WSS' dropdowns are also visible. The 'Raw' tab is active, displaying the following XML content:

```
<?xml version='1.0' encoding='UTF-8'>
<soapenv:Envelope xmlns:xs1='http://www.w3.org/2001/XMLSchema-instance' xmlns:xs2='http://schemas.xmlsoap.org/soap/envelope/'>
  <soapenv:Header/>
  <soapenv:Body>
    <wss:read soapenv:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'>
      <callContext xs1:type='xsd:CAdxCallContext'>
        <codeLang xs1:type='xsd:string'></codeLang>
        <poolAlias xs1:type='xsd:string'>WSSEED</poolAlias>
        <poolId xs1:type='xsd:string'></poolId>
        <requestConfig xs1:type='xsd:string'>adxwss.beautify=true</requestConfig>
      </callContext>
      <publicName xs1:type='xsd:string'>ZSOH</publicName>
      <objectKeys xs1:type='wss:ArrayOfCAdxParamKeyValue'>
        soapenc:arrayType='wss:CAdxParamKeyValue[]'
        <key>SOHNUM</key><value>SONGB0230015</value>
      </objectKeys>
    </wss:read>
  </soapenv:Body>
</soapenv:Envelope>
```

Annotations with arrows point to the 'Submit request' button, the 'SOH READ' method in the Navigator, the 'Authenticate pre-emptively' radio button, and the XML content in the Raw tab.

The Request content

Response Display in XML or RAW Formats

Authentication Username & password

Authenticate pre-emptively selected

# Receiving SOAP Response From Sage X3

The screenshot shows the SoapUI 5.7.2 interface. The main window displays the SOAP request and response in XML format. The request is for the 'SOH READ' operation, and the response is an XML document with a 'RESULT' section containing various product details. The response includes fields like 'SALFCY', 'ZSALFCY', 'SOHTYP', 'ZSOHTYP', 'SOHNUM', 'REVNUM', 'CUSORDREF', 'SOHNUMEND', 'ORDDAT', 'BPCORD', 'BPCNAM', 'CUR', 'ZCUR', 'WSOHCAT', 'BPCINV', 'BPCINAM', 'BPAADD', 'BPDNAM', 'VACBPR', 'ZVACBPR', 'MENULAB', 'HLDSTA', 'HLDSTN', 'HLCOD', 'ZHLDCOD', 'SOHCFMFLG', and 'SOHVALDATC'.

Request Properties:

Property	Value
Name	SOH READ
Description	
Message Size	1032
Encoding	UTF-8
Endpoint	http://x3erpv12sqlv...
Timeout	
Bind Address	
Follow Redirects	true
Username	admin
Password	*****
Domain	
Authentication Type	Preemptive
WSS-Password Type	
WSS TimeToLive	
SSL Keystore	
Skip SOAP Action	false
Enable MTOM	false
Force MTOM	false
Inline Response Atta...	false
Expand MTOM Attac...	false

Authorization: Basic

Username: admin  
Password: \*\*\*\*\*  
Domain:   
Pre-emptive auth:  Use global preference  Authenticate pre-emptively  
Outgoing WSS:   
Incoming WSS:   
Auth (Basic) Headers (0) Attachments (0) WS-A WS-RM JMS Headers JMS Property (0)  
response time: 988ms (11145 bytes)

SOHNUM=SONGB0230015

The request response is displayed in the SoapUI Interface

# Load Testing With SoapUI®

# Load Testing With SoapUI®

Load testing in SoapUI® refers to the process of simulating multiple users accessing a web service to assess its performance under various load conditions. This can help to identify bottlenecks, determine system behaviour under peak load, and ensure that the service can handle the expected number of users without degradation in performance.

## The main steps to create a simple load test in SoapUI®

1. Creating the test case
2. Define Test Data and Assertions
3. Configure Load Test Settings
  1. Threads (virtual users)
  2. Set the total number of requests or time to run requests
  3. Set strategy
  4. Execute the test and review results



# Creating Test Case

A test case typically contains a sequence of requests or a single request to be executed during the load test. For example, we have a SOAP request which uses the query method to list 5 customer records.

The screenshot displays a SOAP client interface with three main panes. The left pane shows a project tree with 'BPC QUERY' selected. The middle pane shows the SOAP request XML, which includes a query method call with parameters for pool alias, pool ID, and list size (5). The right pane shows the SOAP response XML, which contains a 'queryResponse' element with a 'status' of 1 and a 'technicalInfos' block containing various performance metrics. An arrow points from the text on the right to the 'status' field in the response XML.

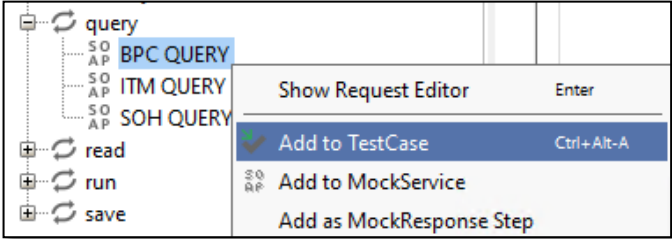
```
<?xml version="1.0" encoding="UTF-8" ?>
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <wss:query soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <callContext xsi:type="wss:CAdxCallContext">
        <codeLang xsi:type="xsd:string">ENG</codeLang>
        <poolAlias xsi:type="xsd:string">WSSEED</poolAlias>
        <poolId xsi:type="xsd:string"></poolId>
        <requestConfig xsi:type="xsd:string">adxwss.beautify=true</requestConfig>
      </callContext>
      <publicName xsi:type="xsd:string">ZBPC</publicName>
      <objectKeys xsi:type="wss:ArrayOfCAdxParamKeyValue" soapenc:arrayType="wss:CAdxParamKeyValue[]">
      </objectKeys>
      <listSize xsi:type="xsd:int">5</listSize>
    </wss:query>
  </soapenv:Body>
</soapenv:Envelope>
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<RESULT>
  <resultXml>
    <status xsi:type="xsd:int">1</status>
    <technicalInfos xsi:type="wss:CAdxTechnicalInfos">
      <busy xsi:type="xsd:boolean">>false</busy>
      <changeLanguage xsi:type="xsd:boolean">>false</changeLanguage>
      <changeUserId xsi:type="xsd:boolean">>false</changeUserId>
      <flushAdx xsi:type="xsd:boolean">>false</flushAdx>
      <loadWebsDuration xsi:type="xsd:double">23</loadWebsDuration>
      <nbDistributionCycle xsi:type="xsd:int">1</nbDistributionCycle>
      <poolDistribDuration xsi:type="xsd:double">2</poolDistribDuration>
      <poolEntryIdx xsi:type="xsd:int">5560</poolEntryIdx>
      <poolExecDuration xsi:type="xsd:double">181</poolExecDuration>
      <poolRequestDuration xsi:type="xsd:double">-1</poolRequestDuration>
      <poolWaitDuration xsi:type="xsd:double">0</poolWaitDuration>
      <processReport xsi:type="xsd:string" xsi:nil="true"/>
      <processReportSize xsi:type="xsd:int">1</processReportSize>
      <reloadWebs xsi:type="xsd:boolean">>false</reloadWebs>
      <resunitAfterDBOpen xsi:type="xsd:boolean">>false</resunitAfterDBOpen>
      <rowInDistribStack xsi:type="xsd:int" xsi:nil="true"/>
      <totalDuration xsi:type="xsd:double">234</totalDuration>
      <traceRequest xsi:type="xsd:string"/>
      <traceRequestSize xsi:type="xsd:int">0</traceRequestSize>
    </technicalInfos>
  </queryReturn>
</wss:queryResponse>
</soapenv:Body>
</soapenv:Envelope>
```

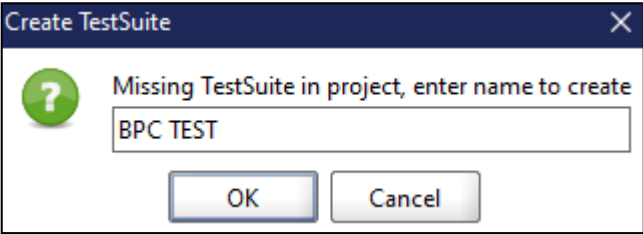
Testing the query response returns the correct results status 1 is returned = successful request & response

# Define Test Data

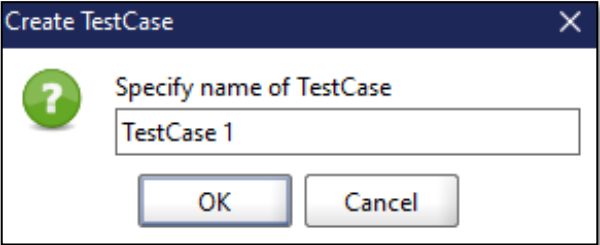
Our test data will be a single SOAP request and response. Next, we need to set up the assertions within the test case to validate the correctness of the responses during the load test.



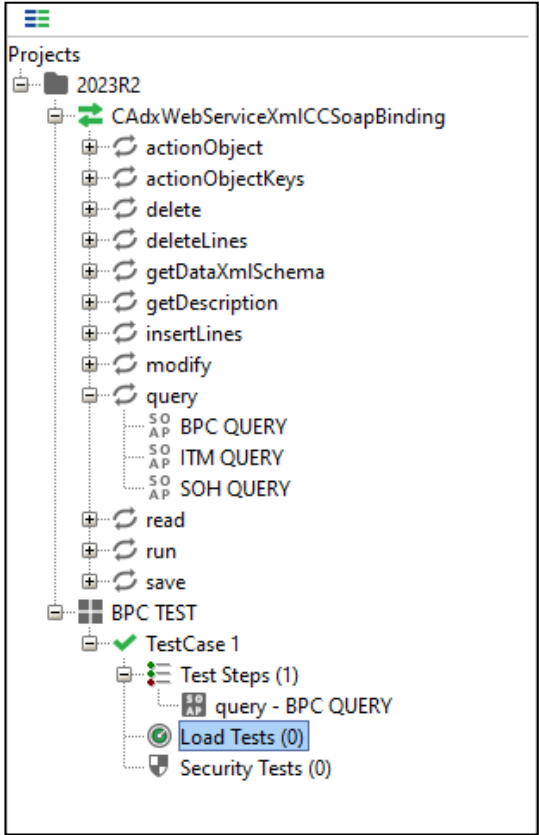
Right-click on the query and select add to test case



Create a test suite in the current project.



Give your test case a name



The test suite is created with the selected request.

# Define The Assertions

Assertions in SoapUI are validation checkpoints that allow you to verify the correctness of responses received from web services during testing. They are used to define criteria that must be met for a test step or a test case to pass. Assertions help ensure the web service behaves as expected and meets the specified requirements.

The screenshot shows the SoapUI interface. On the left is the Navigator pane with a tree view of projects and test steps. The main area displays a SOAP query in XML format. Below the query, there is a list of test steps, including 'query - BPC QUERY'. A dialog box titled 'Contains Assertion' is open, showing the configuration for an assertion. The 'Content' field contains the XPath expression: `<status xsi:type='xsd:int'>1</status>`. The 'Ignore Case' checkbox is checked, and the 'Regular Expression' checkbox is unchecked. The 'OK' button is highlighted.

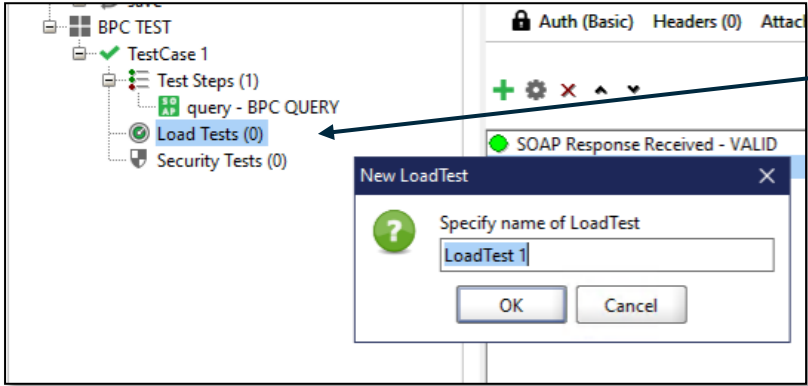
```
<soapenv:Envelope xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance' xmlns:xsd='http://www.w3.org/2001/XMLSchema'>
  <soapenv:Header/>
  <soapenv:Body>
    <wss:query soapenv:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'>
      <callContext xsi:type='wss:CAdxCallContext'>
        <codeLang xsi:type='xsd:string'>ENG</codeLang>
        <poolAlias xsi:type='xsd:string'>WSSSEED</poolAlias>
        <poolId xsi:type='xsd:string'></poolId>
        <requestConfig xsi:type='xsd:string'>adxwss.beautify=true</requestConfig>
      </callContext>
      <publicName xsi:type='xsd:string'>ZBPC</publicName>
      <objectKeys xsi:type='wss:ArrayOfCAdxParamKeyValue' soapenv:arrayType='wss:ArrayOfCAdxParamKeyValue'>
        <listSize xsi:type='xsd:int'>5</listSize>
      </objectKeys>
    </wss:query>
  </soapenv:Body>
</soapenv:Envelope>
```

Press the + to add your assertions in this case we can add an assertion to check the status returned in the response is '1'

The screenshot shows the SoapUI assertion results pane. It displays two assertions, both with green status icons and the text 'VALID'. The first assertion is 'SOAP Response Received - VALID' and the second is 'Status 1 Returned - VALID'. At the bottom, there is a summary bar showing 'Assertions (2)' and 'Request Log (1)'. A play button icon is visible at the top left of the pane.

Press play to launch the query and confirm the assertion status is working

# Create/Configure Load Test Settings



The last step is creating the load test and then configuring the test parameters. Right-click on load tests and select the option to create a new test giving it a name.

Cnt is the number of requests sent and if any of our defined assertions fail are logged in the err column

Test Step	min	max	avg	last	cnt	tps	bytes	bps	err	rat
query - BPC QUERY	95	4178	238.92	274	304	5.03	1103193	18259	0	0
TestCase:	95	4178	238.92	274	304	5.03	1103193	18259	0	0

time	type	step	message
2024-04-29 19:02:12.361	Message		LoadTest started at Mon Apr 29 19:02:12 CEST 2024
2024-04-29 19:03:12.728	Message		LoadTest ended at Mon Apr 29 19:03:12 CEST 2024

1. Number Threads: Specify how many simultaneous users should be simulated during the test.
2. The different test strategies allow you to configure different test options simple making available test delay and randomisation
3. Test delay sets the delay between each test run in milliseconds
4. Random amount of randomisation in the delay 0 = no randomisation
5. Limit sets the total number of tests or time in seconds for the test to run

# Postman® Application

# What Is The Postman Application

Postman is an API (Application Programming Interface) development and testing tool that simplifies the process of building, testing, and documenting APIs. It provides an intuitive user interface for creating and sending HTTP requests to APIs, allowing developers to interact with APIs without writing code manually.

## Key features of Postman include:

**Request Building:** Postman allows users to easily create HTTP requests such as GET, POST, PUT, DELETE, etc., and customize request headers, parameters, and payloads.

**Collection Management:** Users can organise requests into collections, making managing and sharing related API endpoints and test scenarios easier.

**Automated Testing:** Postman supports the creation of automated test scripts using JavaScript. Developers can write tests to verify API responses, status codes, and data integrity.

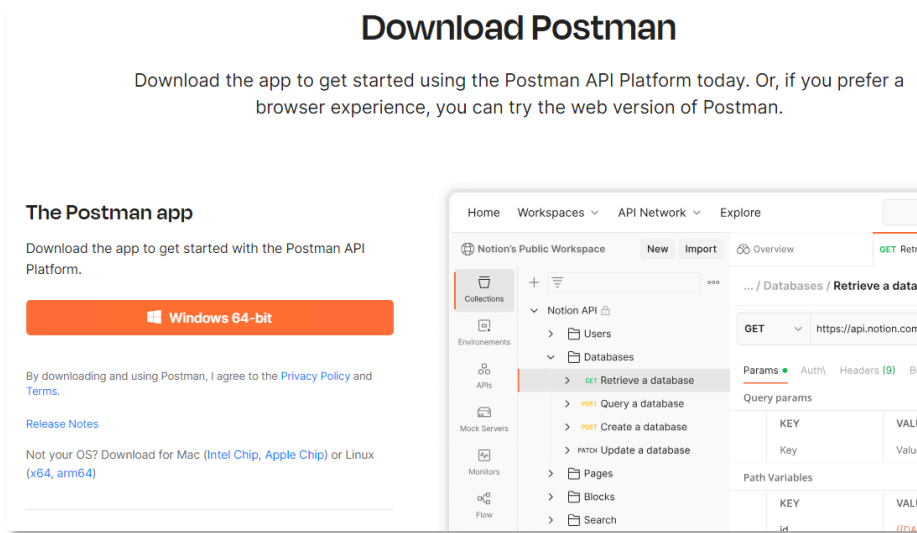
**API Documentation:** Postman enables the generation of interactive API documentation from API requests and responses, making it easier for developers to understand and use APIs.

**Collaboration:** Postman provides features that allow team members to share collections, collaborate on API development, and work together on API testing and debugging.

Overall, Postman streamlines the API development lifecycle by providing a comprehensive set of tools for designing, testing, and debugging APIs, making it a valuable tool for developers and teams working with APIs.

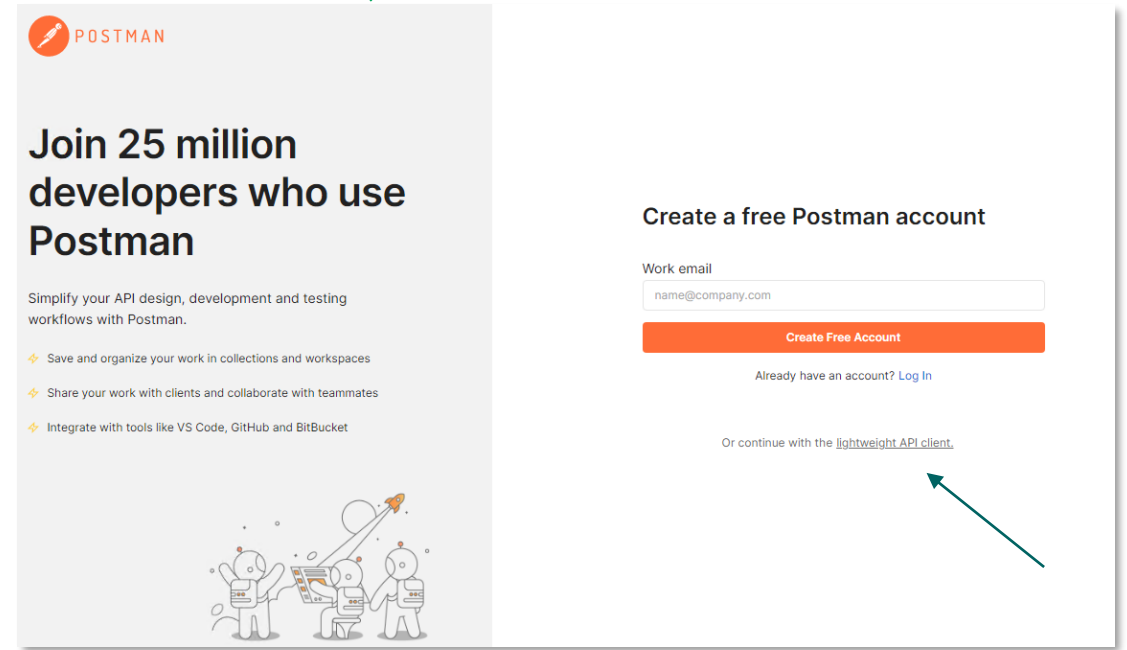
# Postman<sup>®</sup> Installation

# Postman® Installation



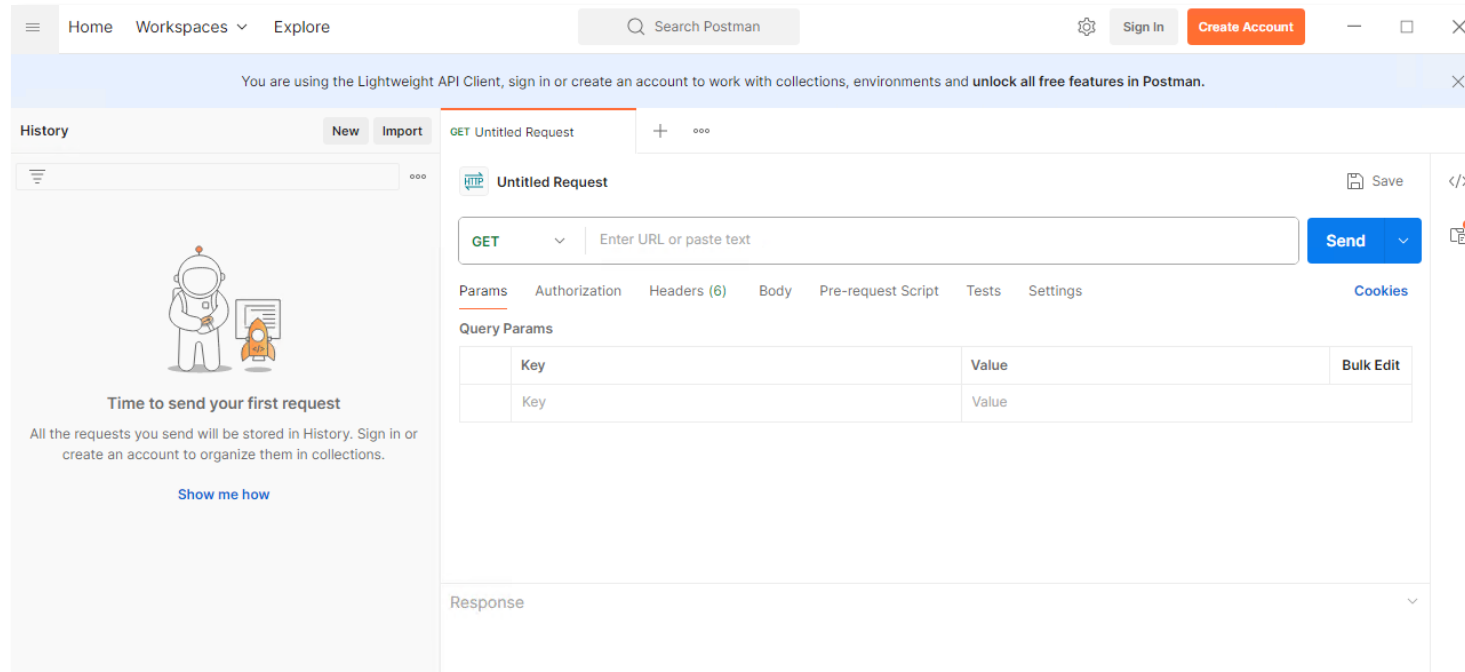
1. Visit the Postman website at <https://www.postman.com/>.
2. click the "Download for Windows" button once the download page opens.
3. Download The Executable

3. Postman offers two versions: 64-bit and 32-bit. Choose the version that matches your system architecture.
4. Click on the download button to start the download.
5. Once the installation is completed, you have the option of signing in or continuing with the lightweight API Client





# Postman® Installation

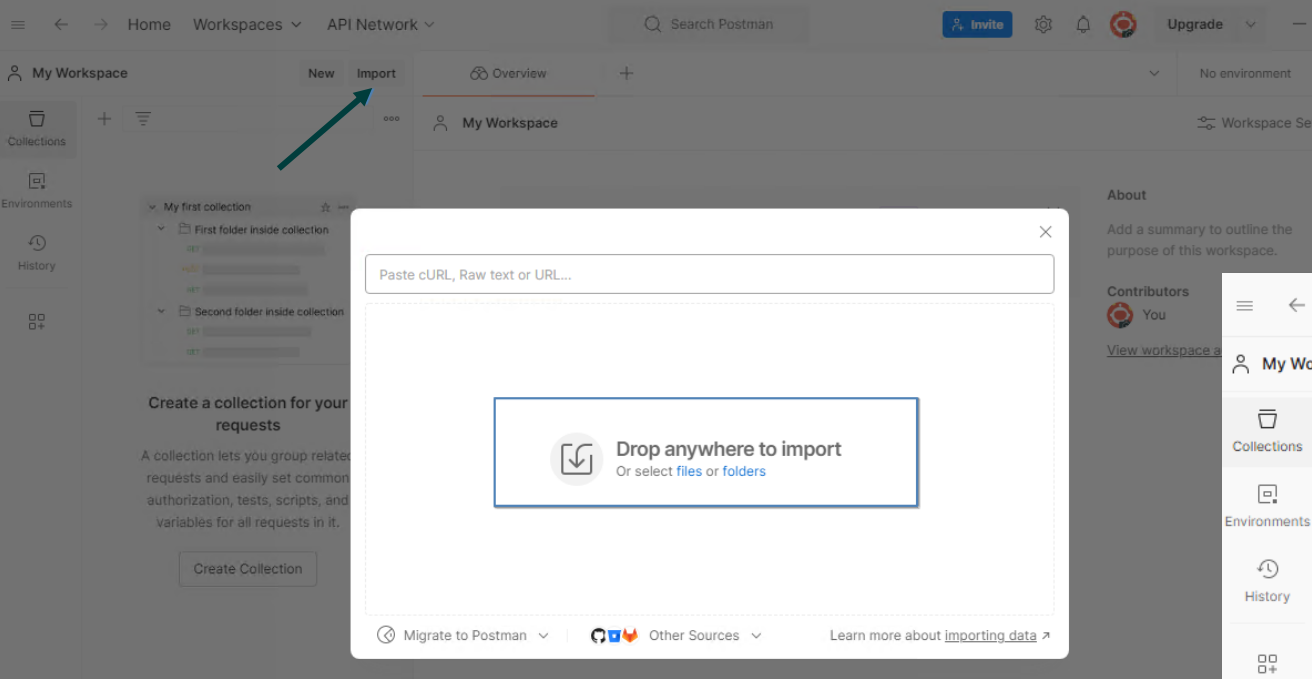


1. Once the installation is complete, you can launch Postman by finding it in your Start menu or by double-clicking the desktop shortcut if you choose to create one during installation.
2. Upon launching Postman, you may be prompted to sign in or create a Postman account. This step is optional but provides access to additional features, such as syncing your collections across devices and importing collections

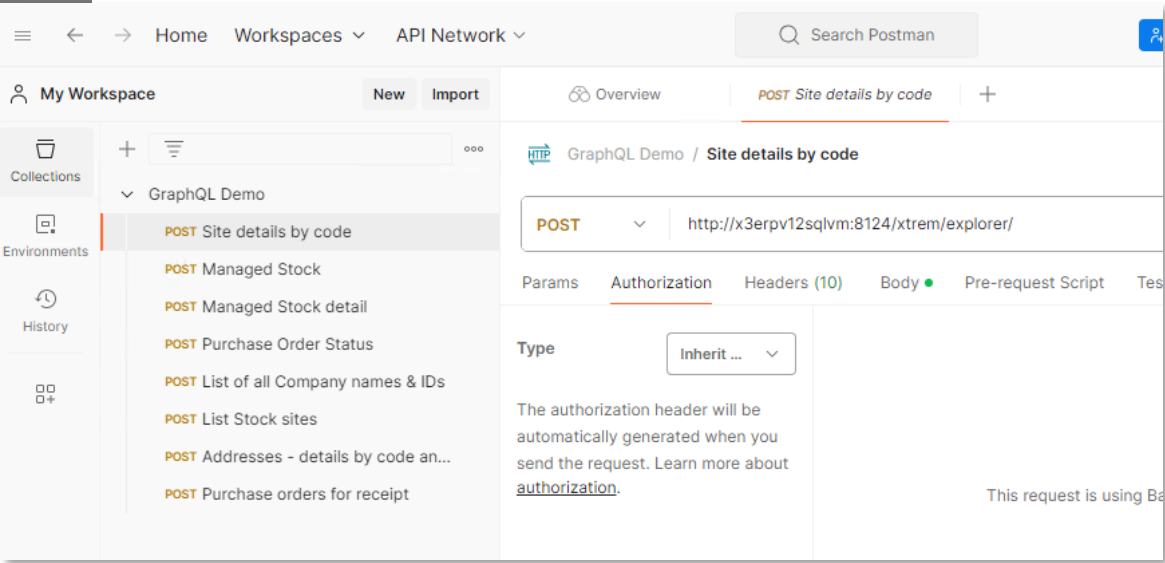
# Postman<sup>®</sup> Sending Requests

# Importing Collections

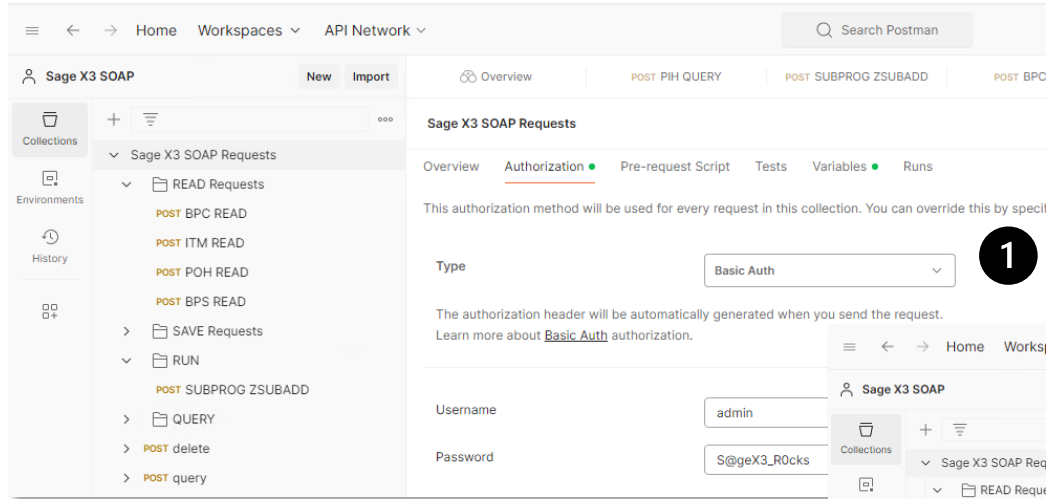
With the lightweight client, you are unable to import any existing collections; once you have created an account, you can import collections. Other advantages include syncing your collections and sharing with other approved users.



GraphQL  
Demo.postman\_collection.json  
JSON File

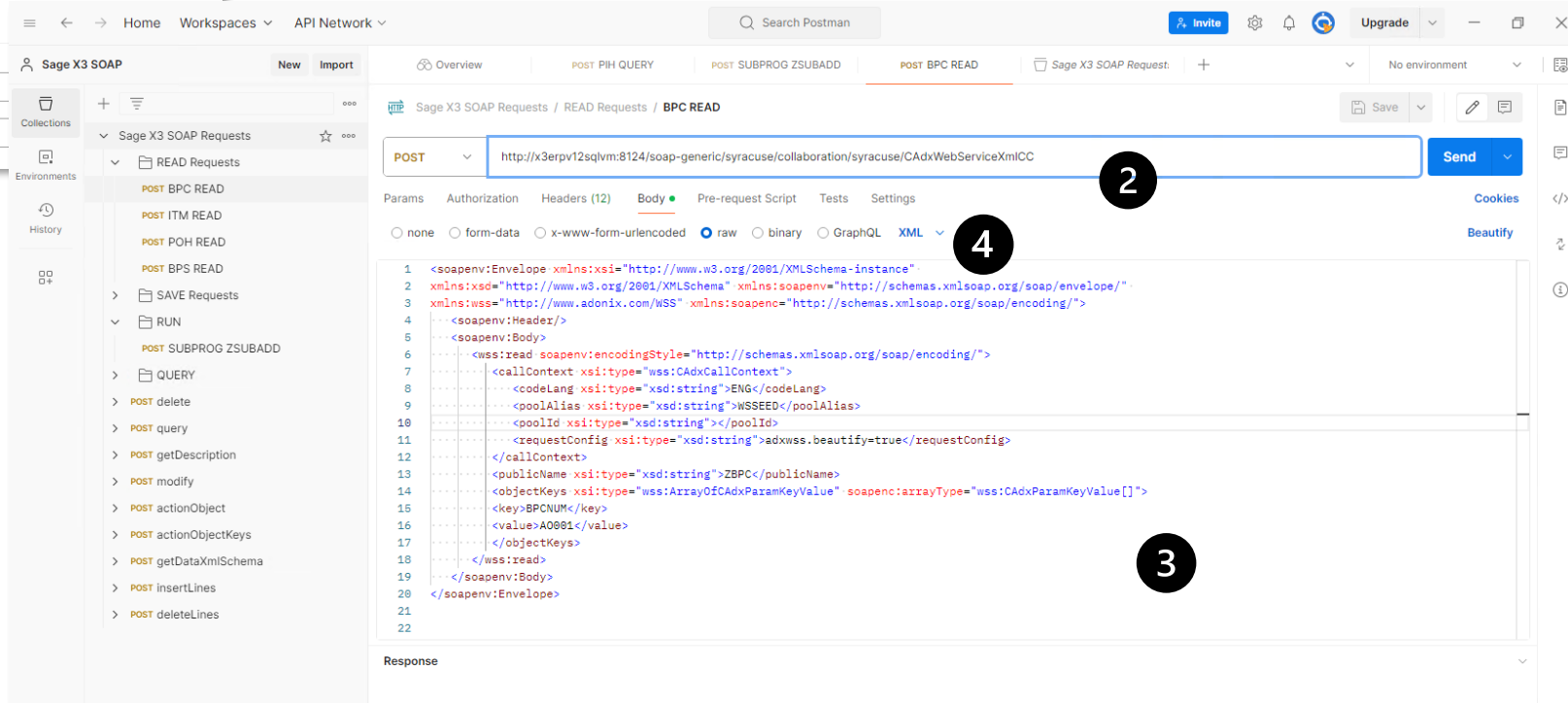


# Sending SOAP Request To X3



1. Configure the authentication this can be done at the individual request level or at the workspace level, and all requests can be set to inherit the authentication settings

2. Populate the address of the web services WSDL and set the request type to POST



3. Populate the body of your SOAP request all parameters will be identical to those used in SoapUI

4. Set the request format in this case, raw XML

# Sending SOAP Request To X3

The screenshot shows the Postman interface for a SOAP request. The request is a POST to the URL `http://x3erpv12sqlvm:8124/soap-generic/syracuse/collaboration/syracuse/CAdxWebServiceXmlCC`. The request body is in raw XML format, containing a SOAP envelope with a `readResponse` element. The response status is 200 OK, with a time of 862 ms and a size of 17.36 KB. The response body is also in raw XML format, containing a `readReturn` element with a `messages` array. The interface includes a left sidebar with collections and environments, a top navigation bar, and a bottom status bar.

5. Press Send and the Response from the SOAP request is displayed

# Sage X3 and GraphQL

# Sage X3 and GraphQL

GraphQL is a query language for APIs and a runtime for executing those queries by a server. It was developed by Facebook in 2012 and later open-sourced in 2015. From Sage X3 2021 R3, we now have the possibility to use the GraphQL API to execute queries and modify Sage X3 data.

## Prerequisites

- Sage X3 services for GraphQL and ADC needs to be installed and configured to utilise the GraphQL API
  - Installing Services
  - Configuring the database connection in Syracuse
  - Detailed installation step & configuration steps - [https://online-help.sagex3.com/erp/12/en-us/Content/V7DEV/getting-started\\_Sage-X3-Services-installation.html](https://online-help.sagex3.com/erp/12/en-us/Content/V7DEV/getting-started_Sage-X3-Services-installation.html)
- For ADC access, you need to have the ADC Badges
- Badges need to be assigned to the appropriate role that will be used by the ADC user

# Sage X3 and GraphQL

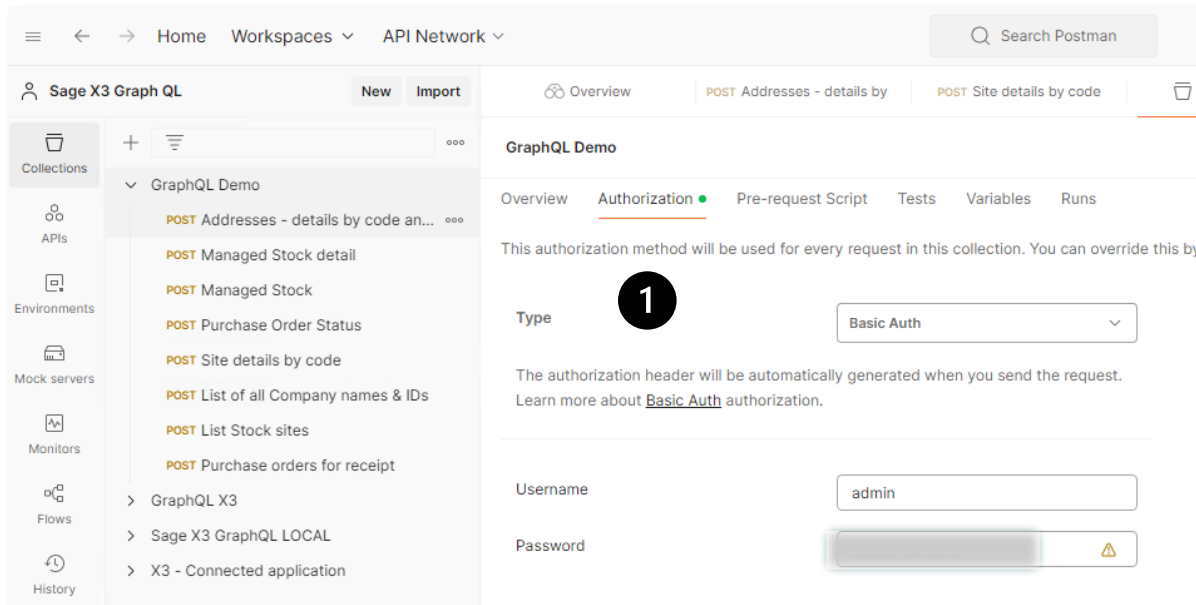
With Sage X3 Services installed, you can access the inbuilt GraphQL explorer, allowing you to interact with your Sage X3 data for the configured folder.

```
1 # GraphQL is an in-browser integrated development environment (IDE) for
2 # writing, validating, and testing GraphQL queries.
3 #
4 # You can use it here to explore and interact with the Sage X3 GraphQL API
5 # and demo data provided.
6 #
7 # Type queries on this side of the screen. Typeahead support lists available
8 # choices based on the text entered. GraphQL also highlights syntax and
9 # validation errors in the code.
10 #
11 # GraphQL queries typically start with a { character.
12 # Comment lines starting with a # are ignored.
13 #
14 # Let's get you started with a simple query to return a list of customers.
15 {
16   xtremX3MasterData {
17     customer {
18       query {
19         edges {
20           node {
21             _id
22             companyName
23           }
24         }
25       }
26     }
27   }
28 }
29 # Note: All queries are limited to 20 results, and mutations to create
30 # and update data are not supported on this demo endpoint.
31 #
32 # Keyboard shortcuts:
33 #   Prettify a query: Shift + Ctrl + P (or select the Prettify button above)
34 #   Run a query: Ctrl + Enter (or select the Run button above)
35 #   Auto complete: Ctrl + Space (or start typing)
36 #
37
```

```
{
  "data": {
    "xtremX3MasterData": {
      "customer": {
        "query": {
          "edges": [
            {
              "node": {
                "_id": "AE001",
                "companyName": "Al Rostamani Communications "
              }
            },
            {
              "node": {
                "_id": "AE002",
                "companyName": "Al Zahra Computers"
              }
            },
            {
              "node": {
                "_id": "AE003",
                "companyName": "Cosmos Computer Co LLC"
              }
            },
            {
              "node": {
                "_id": "AE011",
                "companyName": "Computer Products"
              }
            },
            {
              "node": {
                "_id": "AE001",
                "companyName": "Luanda BTT."
              }
            }
          ]
        }
      }
    }
  }
}
```



# Sending GraphQL Request To X3

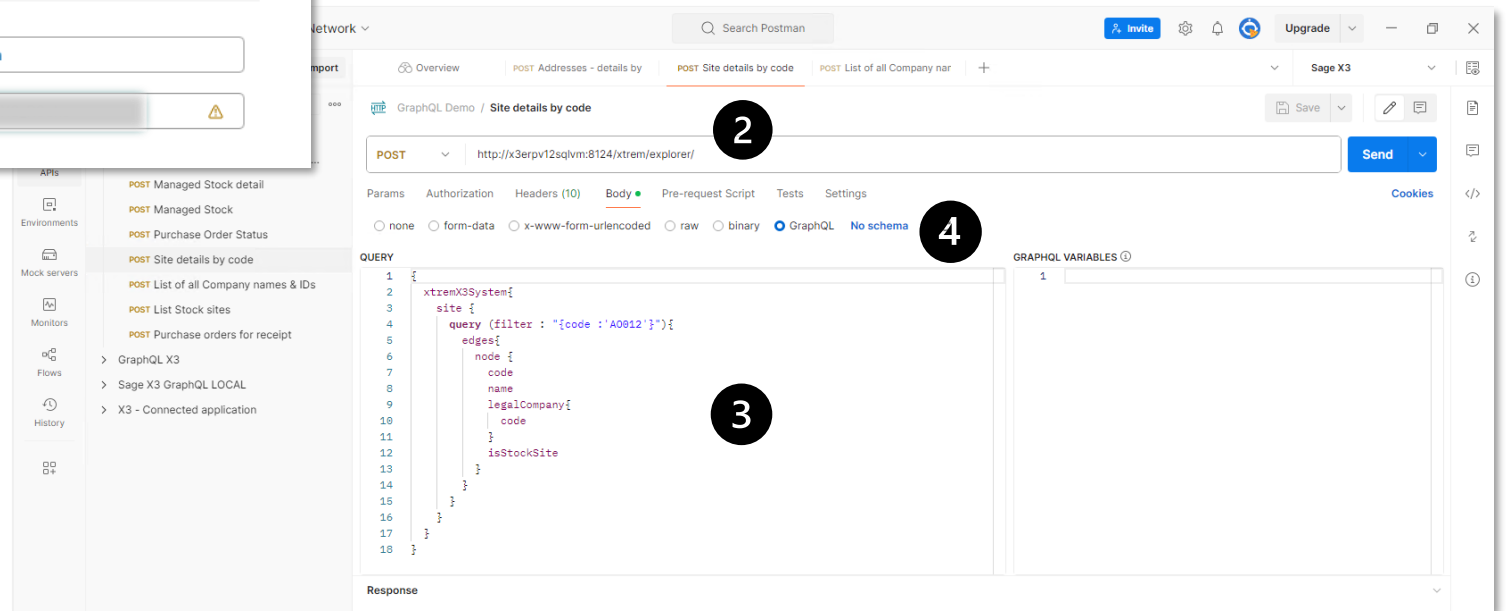


1. Configure the authentication this can be done at the individual request level or at the workspace level, and all requests can be set to inherit the authentication settings

2. Populate the address of the GraphQL service endpoint

3. Populate the body of your GraphQL request

4. Set the request format in this case GraphQL



# Sending GraphQL Request To X3

The screenshot displays the Sage X3 GraphQL interface. The top navigation bar includes 'Overview', 'POST Addresses - details by', 'POST Site details by code', 'POST List of all Company nar', and 'POST List Stock sites'. The main area shows a 'POST' request to 'http://x3erpv12sqlvm:8124/xtrem/explorer/'. The 'Body' tab is selected, showing a GraphQL query:

```
1 {
2   xtremX3System{
3     site {
4       query (filter : "{code : 'A0012'}"){
5         edges{
6           node {
7             code
8             name
9             legalCompany{
10              code
11            }
12            isStockSite
13          }
14        }
15      }
16    }
17  }
```

The 'GRAPHQL VARIABLES' section is empty. Below the query, the 'Body' tab shows the response in 'Pretty' format:

```
1 {
2   "data": {
3     "xtremX3System": {
4       "site": {
5         "query": {
6           "edges": [
7             {
8               "node": {
9                 "code": "A0012",
10                "name": "Manufaturas do Soyo",
11                "legalCompany": {
12                  "code": "A010"
13                },
14                "isStockSite": true
15              }
16            }
17          ]
18        }
19      }
20    }
21  }
```

The status bar indicates 'Status: 200 OK', 'Time: 79 ms', and 'Size: 831 B'. An arrow points from the text '5. Press Send and query results are displayed' to the 'Send' button and the response area.

5. Press Send and query results are displayed

# Demo

# Demo

# Tips

# SOAP Request

## Request Config property

In the request config property, we can specify different settings separated by the ‘&’ character to activate processing settings, which can be used for troubleshooting using the response to the web service call.

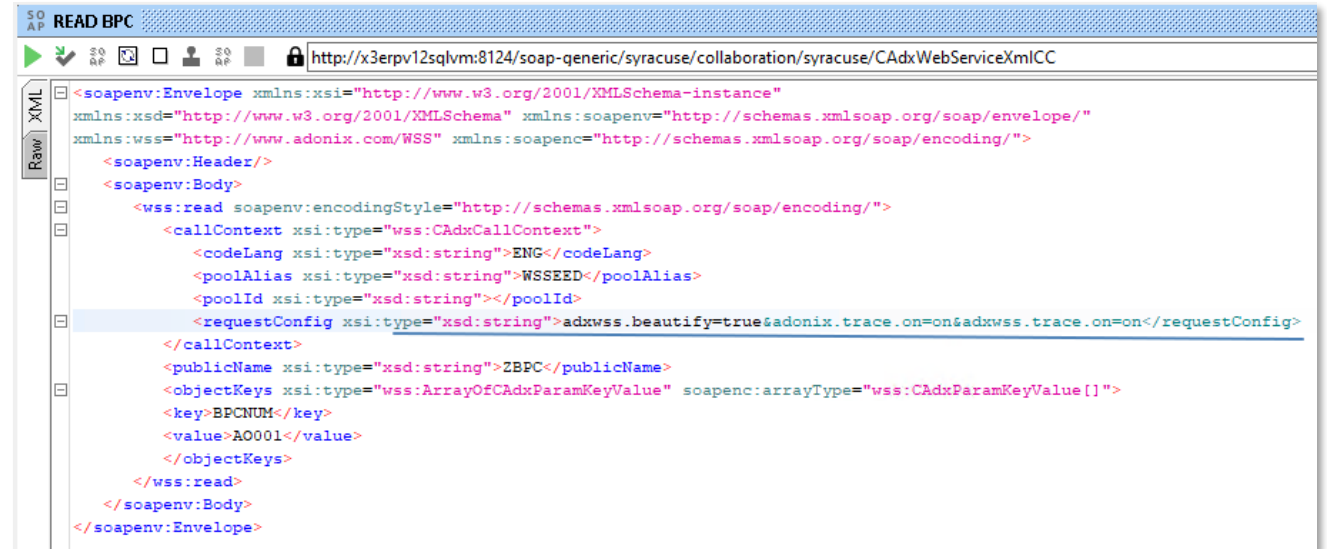
adxwss.trace.on=[on/off] – Activates web server (Syracuse layer) trace and embeds the trace in the response

adonix.trace.on=[on/off] – X3 (classic layer) server trace

adonix.trace.level=(1/2/3) – Specify server trace levels

adxwss.beautify=true – format the response

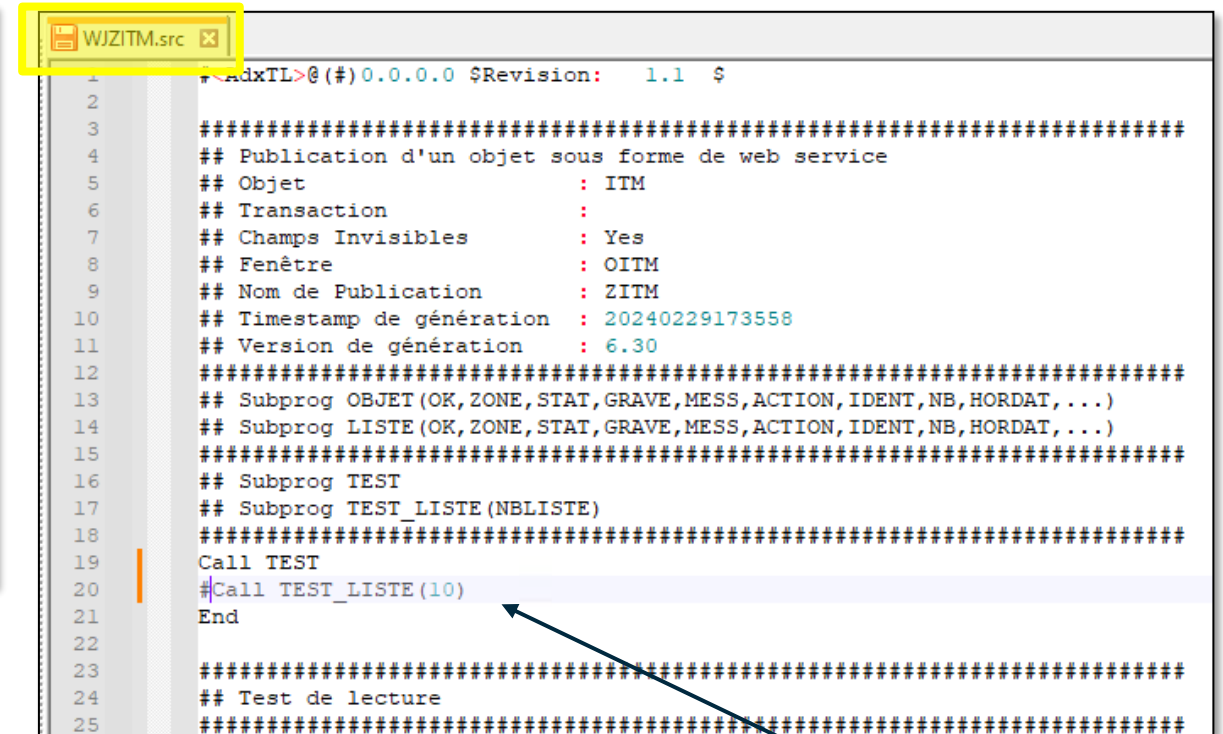
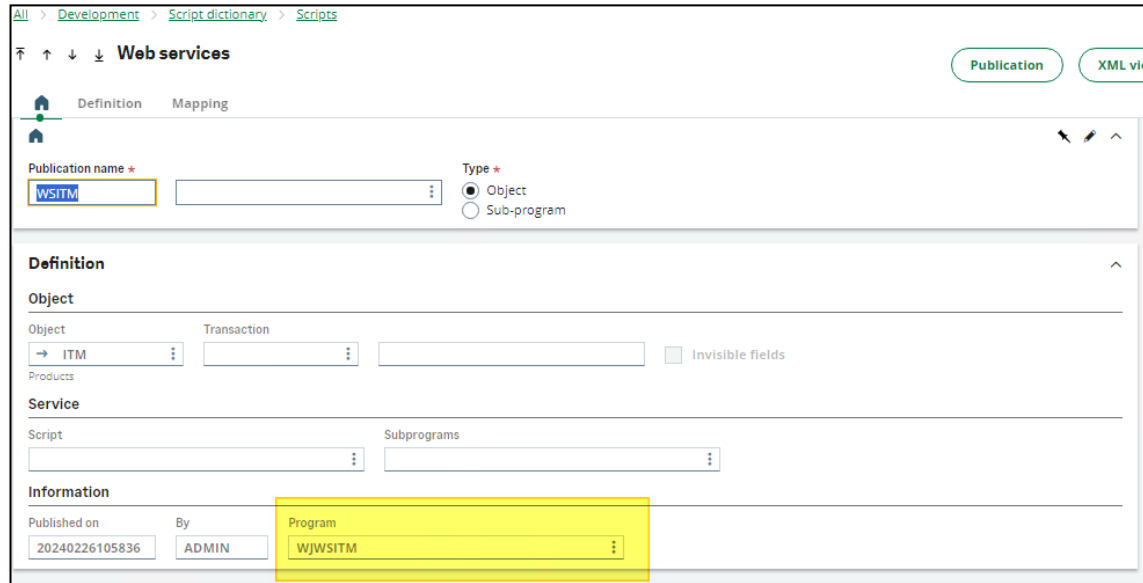
adxwss.optreturn=(XML/JSON) – Response Format



```
<?xml version='1.0' encoding='UTF-8'>
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wss="http://www.adonix.com/WSS" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
  <soapenv:Header/>
  <soapenv:Body>
    <wss:read soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <callContext xsi:type="wss:CAdxCallContext">
        <codeLang xsi:type="xsd:string">ENG</codeLang>
        <poolAlias xsi:type="xsd:string">WSSEED</poolAlias>
        <poolId xsi:type="xsd:string"></poolId>
        <requestConfig xsi:type="xsd:string">adxwss.beautify=true&adonix.trace.on=on&adxwss.trace.on=on</requestConfig>
      </callContext>
      <publicName xsi:type="xsd:string">ZBPC</publicName>
      <objectKeys xsi:type="wss:ArrayOfCAdxParamKeyValue" soapenc:arrayType="wss:CAdxParamKeyValue[]">
        <key>BPCNUM</key>
        <value>A0001</value>
      </objectKeys>
    </wss:read>
  </soapenv:Body>
</soapenv:Envelope>
```

# When there is no SOAP Response

In a situation where your SOAP request is returning no response or not the expected response, we can trace the 4GL program the web service uses.



Open the file which will be in the folder TRT folder, alternatively you could modify using the script editor for example "D:\Sage\X3V12\folders\SEED\TRT\WJZITM.src"

Comment out #Call TEST\_LISTE(10) enable Call TEST

# When there is no SOAP Response

Add the variable being passed in the request; this particular request was a READ on the ITMREF for the value BIKE, so in this case, we can add value BIKE to WW\_IDENT in the web service program source.

```
WZITM.src
10  ## Timestamp de génération : 20240229173558
11  ## Version de génération : 6.30
12  #####
13  ## Subprog OBJET(OK,ZONE,STAT,GRAVE,MESS,ACTION,IDENT,NB,HORDAT,...)
14  ## Subprog LISTE(OK,ZONE,STAT,GRAVE,MESS,ACTION,IDENT,NB,HORDAT,...)
15  #####
16  ## Subprog TEST
17  ## Subprog TEST_LISTE(NBLISTE)
18  #####
19  Call TEST
20  #Call TEST_LISTE(10)
21  End
22
23  #####
24  ## Test de lecture
25  #####
26  Subprog TEST
27  Local Char TEXTE(20)
28  Local Integer I , J , K
29  Call OUVRE_TRACE("") From LECFIC
30  Gosub DEFVAR
31  WW_ACTION = "READ" : # READ, CREATE, MODIFY, DELETE, SUPLIG, INSLIG
32  WW_IDENT = "BIKE" : # Clé de 1 Objet Vall~Val2
33
```

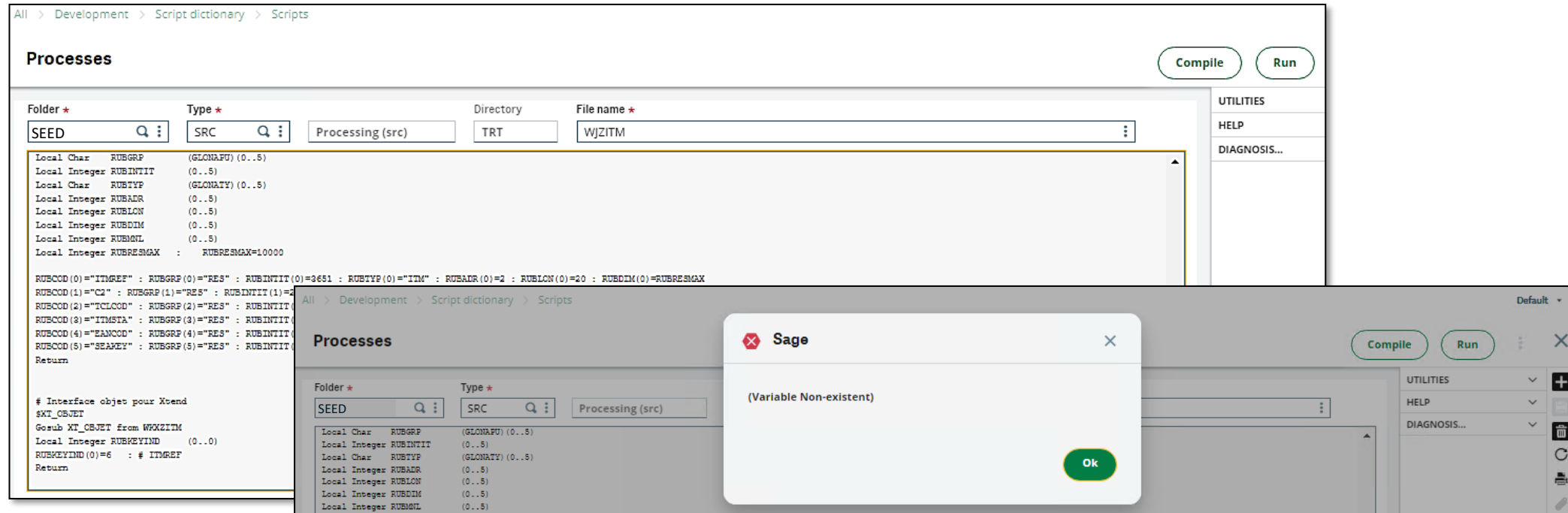
```
WZITM.src
25  #####
26  Subprog TEST
27  Local Char TEXTE(20)
28  Local Integer I , J , K
29  Call OUVRE_TRACE("") From LECFIC
30  Gosub DEFVAR
31  WW_ACTION = "READ" : # READ, CREATE, MODIFY, DELETE, SUPLIG, INSLIG
32  WW_IDENT = "BIKE" : # Clé de 1 Objet Vall~Val2
33
34  Onerrgo ERREURWS
35  Gosub INITMAJ From ZZWSITM
36  Onerrgo
37
38  Gosub WEBSERV
39  Gosub ANALRESOBJ
40
41  Onerrgo ERREURWS
42  Gosub RESMAJ From ZZWSITM
43  Onerrgo
44
45  #Gosub DETAILRES : # Pour afficher le détail des variables lues ou écrites
46  Call FERME_TRACE From LECFIC
47  Call LEC_TRACE From LECFIC
48  End
49
```

Enable/uncomment #Gosub DETAILRES function

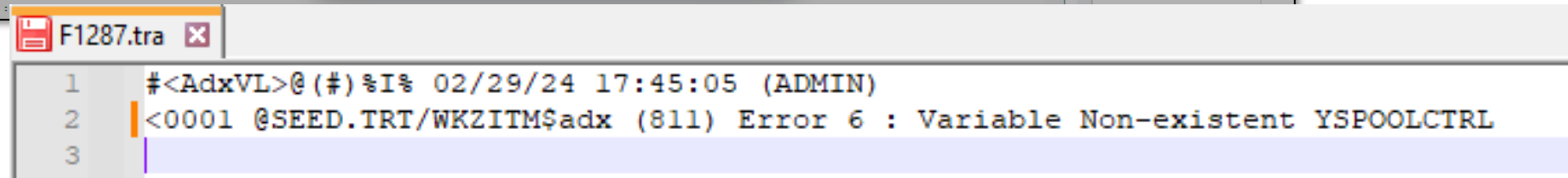


# When there is no SOAP Response

In Sage X3 Script editor, RUN the web service program, and an error will be displayed on the screen



The full error is logged in a trace file in the TRA director of the folder, On this case a variable was missing



# Control Batch Server using Postman

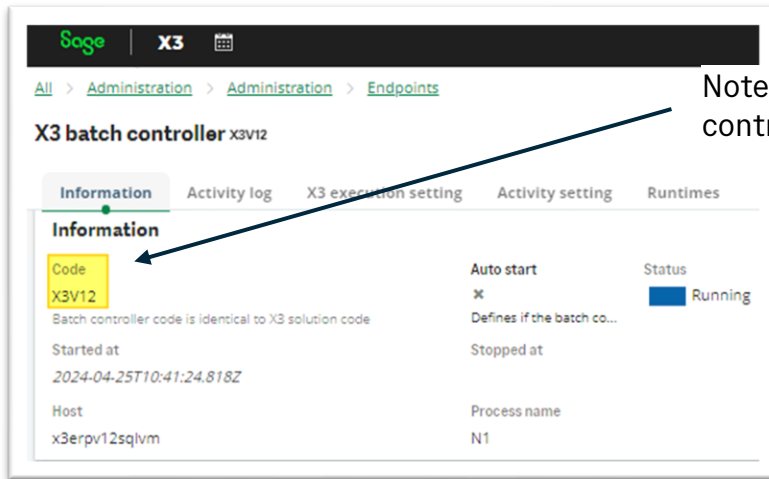
Postman can also be used for REST requests , which can be used to control the Batch Server

## To stop all Post

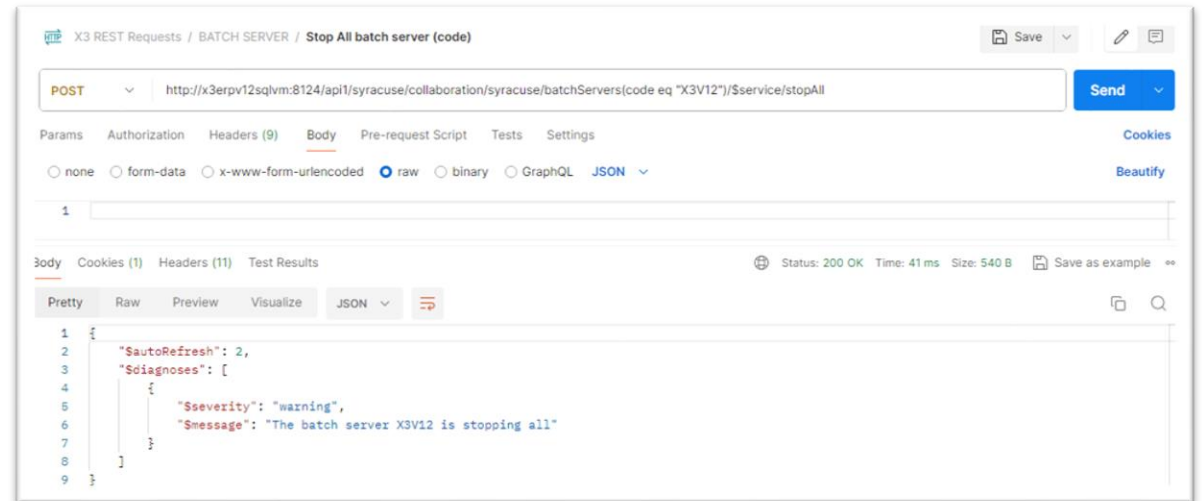
`http://localhost:8124/api1/syracuse/collaboration/syracuse/batchServers(code eq "X3V12")/$service/stopAll`

## To start Post

`http://localhost:8124/api1/syracuse/collaboration/syracuse/batchServers(code eq "X3V12")/$service/start`



Note that X3V12 is the Batch controller code



Other tasks can also be performed, like getting the status of the batch server and the configuration information.

More info can be found in the online help <https://online-help.sageerpx3.com/erp/12/staticpost/how-to-control-the-batcher-server/>

# GraphQL Network Inspector

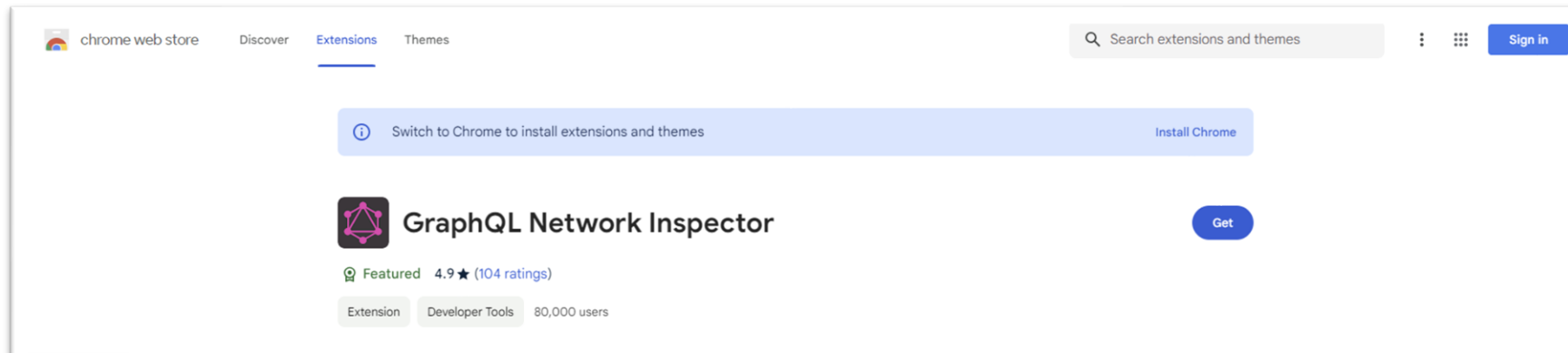
The GraphQL Network Inspector for Chrome is a tool designed to help debug and analyse GraphQL requests and responses in web applications.

With this extension, you can

View GraphQL Requests and Responses: It allows you to see the GraphQL requests made by the application and the corresponding responses returned by the server.

Extension can be used to debug GraphQL queries, identify errors, and optimize query performance.

Provides visibility and insights into the communication between their web applications and GraphQL servers.



[GraphQL Network Inspector \(google.com\)](https://chrome.google.com/webstore/detail/graphql-network-inspector)

# GraphQL Network Inspector

The screenshot displays the GraphQL Network Inspector interface. On the left, a sidebar shows a list of purchase orders under the heading "Purchase orders" and "Purchase order lines:4". The main panel is divided into two sections: a table of queries and mutations, and a detailed view of the selected query.

Query / Mutation	Status	Size	Time	URL
Q pages	200	75.1 kB	152ms	http://x3erpv12sqlvm:81...
Q __type	200	23.6 kB	526ms	http://x3erpv12sqlvm:81...
Q xtremX3Purchasing	200	400 B	1.3s	http://x3erpv12sqlvm:81...
Q xtremX3Purchasing	200	697 B	953ms	http://x3erpv12sqlvm:81...
Q xtremX3Purchasing	200	697 B	843ms	http://x3erpv12sqlvm:81...
Q pages	200	88.7 kB	81ms	http://x3erpv12sqlvm:81...
Q xtremX3Purchasing	200	573 B	691ms	http://x3erpv12sqlvm:81...

The detailed view shows the following query:

```
query {
  xtremX3Purchasing {
    purchaseOrderLine {
      query(
        first: 200
        filter: "{\"purchaseOrder\":\"POGB0120022\",\"receiptSite\":\"GB012\",\"workOrderInProgressStatus\":\"\"}"
        orderBy: "{\"lineNumber\":\"1\"}"
      ) {
        edges {
          node {
            purchaseOrder {
              id
            }
            lineNumber
            receiptSite {
              code
            }
            quantityInPurchaseUnitOrdered
            quantityInStockUnitOrdered
            quantityInStockUnitReceived
            quantityInPurchaseUnitReceived
            purchaseUnit {
              code
            }
            stockUnit {
              code
            }
            product {
              code
              localizedDescription1
            }
            lineStatus
            isClosed
            lineType
            workOrderInProgressStatus
            sourceRequest
            expectedReceiptDate
          }
          cursor
        }
        pageInfo {
```

Query/ Mutations clearly visible

GraphQL Network tab is available in developer tools after the extension has been installed

Option to see the response to this query request

Each request is formatted and displayed

# Appendices



# Useful Links

Resource	Link
Postman help	<a href="#">Postman   Sage Developer</a>
SoapUI® Official Help	<a href="#">An Introduction to API Testing with SoapUI®</a>
REST Webservices in detail	<a href="#">Sage X3 Technical Support Tips and Tricks (May 2023)</a>
Installing X3 Services	<a href="#">Sage X3 Services installation</a>
Web Services in detail	<a href="#">Sage X3 Technical Support Tips and Tricks (March 2022)</a>
How to use REST to GET and Post using Postman	<a href="#">How to use REST Web services to Get and Post a record</a>
Graph QL Network Inspector	<a href="#">GraphQL Network Inspector (google.com)</a>
Sage X3 Developer Portal	<a href="#">Develop for X3   Sage Developer</a>

# Thank you!

