Version 25 | Trapeze Enterprise Asset Management

MAXQueue Interface Data Exchange Methods

Reference Guide

Trade Secret | February 2025

Technical Support

Trapeze provides several ways to connect with the Customer Care team. Please provide detailed information to the representative. If you are reporting an issue by e-mail, include any error messages, screen shots of your problem, and steps to replicate the issue.

Customer Care is available Monday through Friday, 8:00 a.m. to 8:00 p.m., Eastern Standard Time.

Toll-Free Telephone:	1-877-411-8727
E-mail:	cc@trapezegroup.com
FTP Site:	https://ftps.trapezegroup.com/
Web Site:	https://collaborate.trapezegroup.com

The website provides access to Trapeze Collaborate which offers resources to implement and operate the software, search knowledge, participate in communities, log/track service requests, sign-up for release alerts, view product health recommendations, and download updates or documentation. For secure access to Collaborate, contact Customer Care through the website or by calling the number above.

Product Compatibility

Refer to the latest Product Compatibility information on Trapeze Collaborate for requirements including operating system information, databases, Crystal reports, network protocol, HTTP servers and browsers that are supported.

Interface Data Exchange Methods - Reference Guide

Version 25.x

February 2025

Contents

1. Overview 2

2. Staging Tables 2

MAXQueue Staging 2

Notes 2

Import and Export Staging Tables 3

Custom Table Outline 4

ODBC/DNS Settings 4

Notes 4

3. EAM REST API 6

Import and Export Staging Tables 6

4. Web Services Calls 7

XML Passed via HTTP/SOAP 7

Web Service Methods 7

Register 8

GetRegistrations 8

SendMessage 8

GetResponse 8

ProcessEvent 8

Making Web Service Calls 8

SOAP 9

Example of Web Service Use with Visual Studio and C# 12

Generating Code from WSDL File 12

Using Generated SOAP Objects 18

ProcessEvent Responses 22

SendMessage, GetResponse Details 22

5. Flat Files 24

File Transfer 24

File Types 25

Specification Details 25

Notes 25

6. Other 26

1. Overview

The purpose of this document is to provide information about the options for exchange data between FleetFocus and their other systems using MAXQueue.

There are three main ways to exchange data using MAXQueue: staging tables, web services, and flat files.

2. Staging Tables

Staging tables are database tables either in the MAXQueue staging schema, in the FleetFocus database, or in a client database. These are used to exchange data between FleetFocus and client systems.

- MAXQueue has a built-in maxqueue.MAXQ_Staging table to store structure data, and once processed, this data may be deleted depending on the workflow you have chosen for the staging tables. In other cases, it will be marked as processed and kept as log of the activity.
- An import staging table is one in which the client sends data to the FleetFocus system, by writing a record in the staging table.
- An export staging table is one in which FleetFocus writes a record to the staging table.

MAXQueue Staging

MAXQueue has a built-in table that is intended to store structure data (with a preference to XML) and once processed, this data is deleted. A user can write data into to this table (maxqueue.MAXQ_Staging), with the intent that a MAXQueue service later will pick up the record out of this table, process it, and delete it.

Alternatively, if preferred, a MAXQueue custom service can write to this table with the expectation that a customer will later read, process, and delete the records from this table instead.

To use these tables as part of the MAXQueue custom service, you and the software provider must both agree on the contents of the columns: EventDirection, EventType, and EventKey. The XML data (or other structure data) will be placed in EventDetail column.

Notes

- This is an easy-to-use method as no custom SQL scripts needed to define the table.
- This also means that no extra security is necessary.
- Changing the data schema is easier due to the structure.

- Complexities come in that it may be more difficult for some clients to read or write data to an XML structure.

Import and Export Staging Tables

Import staging table: This table is one in which the client sends data to the FleetFocus system by writing a record in the staging table. In the case of an import table, a custom MAXQueue service will read records from the staging table and process them in the system. It will then mark the record as processed.

Export staging table: This table is one in which FleetFocus writes a record to the staging table. In the case of an export table, the client is responsible for processing the records in the staging table.

In both cases, it is up to your organization to define when and how records are purged from these tables. If you do not define this, then the staging table will continue to grow, and no records will be removed. You can handle the purging of records from the staging table by hand, or you may define the parameters in the specification so that the MAXQueue service is responsible for purging records. For example, if using the MAXQueue parameters, you can set purging so that every three days after processing, a record is automatically removed.

If the staging tables are hosted in the MAXQueue schema (the FleetFocus database), then security is handled automatically. However, if the staging tables are hosted externally (i.e. in a separate user database) then you are responsible for ensuring the appropriate security is applied to the staging tables. In particular, during configuration of your MAXQueue services, you will need to provide a database user and password that have authority to read and update the staging tables, and if MAXQueue is responsible for purging the staging tables, authority to delete from the staging table.

In general, the system that will be receiving the data in the staging table should be the system that hosts the staging table. It is recommended that the import staging table be internally (FleetFocus) hosted and the export table should be externally (client) hosted.

⚠ If staging tables are hosted externally, SQL joins between the staging tables would involve security risks, and therefore is not a practice we use. No joins in MAXQueue services are allowed between an external staging table and FleetFocus.

Custom Table Outline

You and the software provider must both agree to the definition of the table. All tables will include the following items.

Column	Data Type	Description
ROW_ID	Integer	Auto sequence number
X_DATETIME_INSERT	Date	Defaults to system date
PROCESSED_FLAG	CHAR(1)	N=Not processed; I=In process; Y=Processed
PROCESSED_DATETIME	Date	System date when PROCESS_FLAG set
Remaining columns of data to be exchanged		

ODBC/DNS Settings

As part of the MAXQueue base installation, an ODBC/DNS was created which points to the FA/EAM database. If the staging tables are hosted in the MAXQueue schema, then that ODBC/DNS will have access to the staging tables.

If the staging tables are hosted externally, then MAXQueue does not have access to these tables and the client must provide access by:

- Creating a database user in the hosting database and grant that user select, update, and delete permissions to the staging tables.
- Creating a new ODBC/DNS on the MAXQueue server, that uses the user and password created previously, which points to the server and database containing the staging tables.

This ODBC/DNS will be using during the configuration of the MAXQueue custom services.

Notes

- The SQL script defining the table is an authoritative definition of the data structure, which can be easily understood.
- Direct ODBC connections with staging tables can be particularly good when the interface collects data for a period of time and then processes the whole batch in/out (for example, a monthly billing interface), and when calculations need to be run against a set of data prior to processing.

- If the staging records are not immediately purged after being processed, having the processed records in the staging table can make it easier to troubleshoot; particularly in reference to a web service call.
- There are several potential complexities regarding use of this method:
    - Security complexities could be present if the staging table is not in the same database schema. You must consider if there is a firewall between the servers, or tables are managed by different business units.
    - An ODBC connection must be created on the MAXQueue server for the server and database that hosts the staging tables. Potential difficulties can arise with establishing or setting up this ODBC connection across servers.
    - Create a plan for removing old data or it may build up indefinitely. This typically is managed by the user organization.
    - Adding or removing columns is more difficult than the other formats and requires use of an alter script, and at times, a change in the interface.
    - If the staging table is not in the same database, SQL joins between the staging tables and internal tables cannot be performed. This may make the creation of the MAXQueue service more difficult.

3. EAM REST API

Import and Export Staging Tables

MAXQueue Staging endpoint

The EAM REST API has a built-in endpoint that is intended to write to the MAXQueue Staging table. Any data written to the MAXQueue Staging API will be processed like data written to the table either directly by a client or via a custom MAXQueue service.

The MAXQueue Staging API can also be used to get data from the MAXQueue Staging table that has been written to the table by a custom MAXQueue service.

To use these tables as part of the MAXQueue custom service, you and the software provider must both agree on the contents of the columns: EventDirection, EventType, and EventKey. The data will be placed in the EventDetail column. This data can be text, xml, json, or a delineated list. The ProcessingTag field can also be used to store information about the state of data in the table.

This API endpoint supports POST (create), PUT (update), and GET (read) operations. Deletion of records will be handled by the custom MAXQueue service.

Any service making a GET request must supply EventType, EventDirection, and EventKey values. Depending on the intended use of the data one or more MAXQueue staging table records can share with these values.

Notes

- No network configuration required to allow access to database server or a file location.
- This also means that no extra security is necessary.
- Changing the data schema is easier due to the structure.
- Complexities come in that it may be more difficult for some clients to read or write data to processable data structure (either XML or JSON).

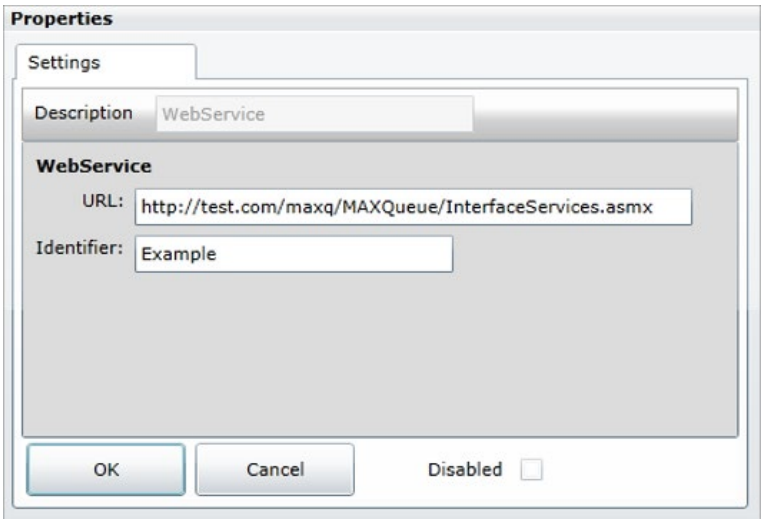
4. Web Services Calls

XML Passed via HTTP/SOAP

MAXQueue web services can be used to trigger MAXQueue services, which can in turn retrieve data, save data, and interact with FleetFocus.

MAXQueue services that are web accessible begin with a Web Service Entry Point adapter. That adapter will contain the URL of the MAXQueue server's Interface Services page, and a unique identifier that will be used to distinguish it from other web services.

The Properties window for that adapter (accessed by clicking the button in the top right of the adapter) will look something like the following image:



Note: This is only an example, and you should not change these properties unless directed to do so by customer support or development.

The Interface Services page can be accessed by using the same URL used to access the MAXQueue Designer, but change the last section "/maxqueuedesigner.aspx" to "/InterfaceServices.asmx".

- For example: "<IPOrDomainName>/<NameOfISSWebInstance/MAXQueue/InterfaceServices.asmx"

Using the Interface Services page, you can access the methods listed below to test that they work without needing to access them through code.

Web Service Methods

The MAXQueue Web Services have the following available methods.

Register

This method is used to register a service using an interface identifier and a destination. This should generally not be used by the client because MAXQueue will automatically register any web services when it runs, provided that the adapter URL/identifier is appropriate, MAXQueue Server is running, and the IIS instance is set up properly. Only use this method if directed to do so by customer support.

GetRegistrations

This method is used to get a list of all of the currently registered services. This is a good way to check that the MAXQueue services are running as expected because if they are running correctly, they will be listed here.

SendMessage

This sends an event XML to a service identified by the ID attribute on the root node (<event id='Example'>). This method does not wait to receive the response from the server, instead it returns an event ID, which can be passed into GetResponse to retrieve the response later.

Note: The service must show as registered (see GetRegistrations) to receive an event.

GetResponse

Gets the web service's response for the service name and the event ID returned by SendMessage.

ProcessEvent

This is a combination of SendMessage and GetResponse so it will send the event XML to the service and then receive the response.

Making Web Service Calls

There are three ways to call MAXQueue Web Services: SOAP, GET, and POST. Any of the three methods work equally well, and it is entirely up to your organization to decide how they interact with the web services.

To send information to any of these web services, you will need to construct XML as defined in the specification documents for the interfaces in question, or as defined in an agreed upon format (e.g. an XSD file). That XML is then passed to MAXQueue, typically through the ProcessEvent method. The root node of that XML should be an event tag, and should have an ID attribute that contains the identifier for the service that should receive the XML.

Below is an example of what that XML may look like:

```
<event id="Example">
    <exampleData>
        .
        .
        .
    </exampleData>
</event>
```

SOAP

If using SOAP calls, a WSDL file can be generated from your MAXQueue installation by visiting the following URL:

<IPOrDomainName>/<NameOfIISWebInstanceForInterfaceServices>/MAXQueue/InterfaceServices.asmx?wsdl

- <NameOfIISWebInstanceForInterfaceServices> can be different from what is set in the Configurator that sets the Integration Designer application name.
- This WSDL will be the same regardless of the custom interfaces that have been built for the client because the methods for interacting with the web server are generic (refer to the Web Service Methods section for details).

For a full example of a web service call from importing the WSDL to using the SOAP objects to call the Web Service method, refer to the HTTP POST section.

SOAP 1.1

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

```
POST /MyDeployment/MAXQueue/InterfaceServices.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/ProcessEvent"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlso
ap.org/soap/envelope/">
  <soap:Body>
    <ProcessEvent xmlns="http://tempuri.org/">
      <event id="Example">...</event>
    </ProcessEvent>
  </soap:Body>
</soap:Envelope>
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
```

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ProcessEventResponse xmlns="http://tempuri.org/">
      <ProcessEventResult>string</ProcessEventResult>
    </ProcessEventResponse>
  </soap:Body>
</soap:Envelope>
```

SOAP 1.2

The following is a sample SOAP 1.2 request and response. The placeholders shown need to be replaced with actual values.

```
POST /MyDeployment/MAXQueue/InterfaceServices.asmx HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <ProcessEvent xmlns="http://tempuri.org/">
      <event id="Example">...</event>
    </ProcessEvent>
  </soap12:Body>
</soap12:Envelope>
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <ProcessEventResponse xmlns="http://tempuri.org/">
      <ProcessEventResult>string</ProcessEventResult>
    </ProcessEventResponse>
  </soap12:Body>
</soap12:Envelope>
```

HTTP GET

The following is a sample HTTP GET request and response. The placeholders shown need to be replaced with actual values.

```
GET /MyDeployment/MAXQueue/InterfaceServices.asmx/ProcessEvent?event=<exampleData>...</exampleData HTTP/1.1
Host: localhost
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
```

```
8.  <string xmlns="http://tempuri.org/">string</string>
```

HTTP POST

The following is a sample HTTP POST request and response. The placeholders shown need to be replaced with actual values.

```
POST /MyDeployment/MAXQueue/InterfaceServices.asmx/ProcessEvent HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: length

event=<exampleData>...</exampleData>
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
```

Example of Web Service Use with Visual Studio and C#

Generating Code from WSDL File

To generate C# code from the WSDL file:

1. Find the WSDL file on the MAXQueue Server and copy the URL. Refer to the SOAP section for details on where the WSDL file is located.

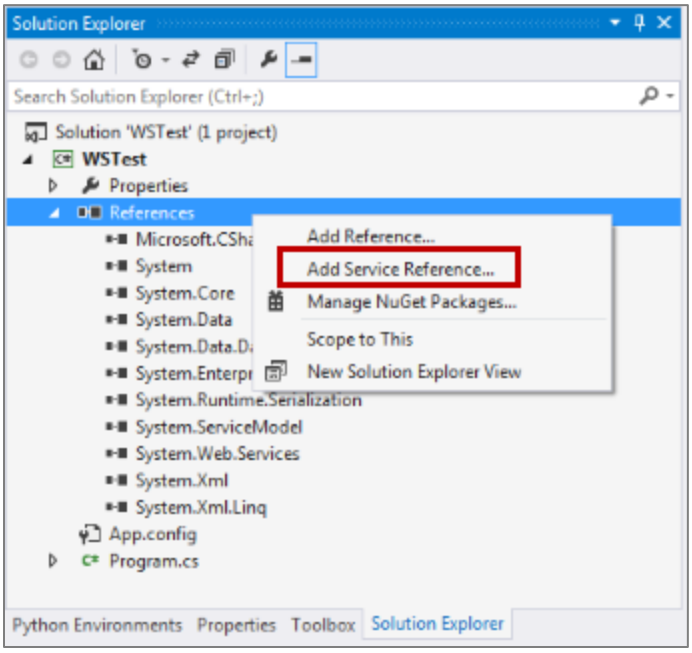
   Note: To confirm that the URL is correct, navigate to it in a web browser. The webpage should show the content of the WSDL file if it is correct.

2. Open Visual Studio (2010 or later).
3. Open the project that the code will be added to.
4. Follow the steps below for individual setup processes.

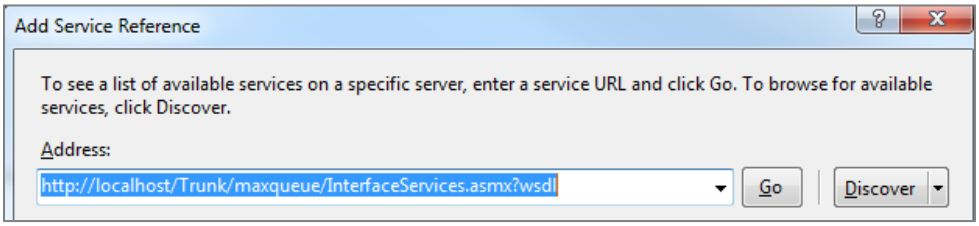
Service Reference Example (Recommended)

1. Right-click on **Service References**, and select **Add Service Reference…**

   If the project does not have a Service References folder, right-click on **References** instead.

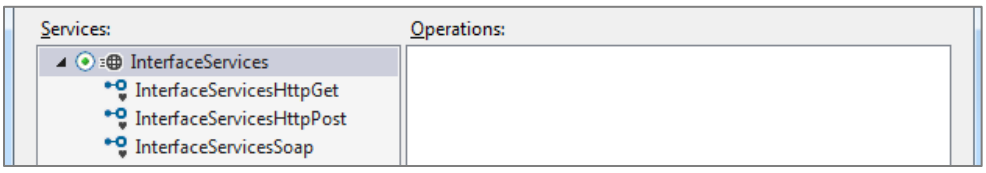


2. Paste the WSDL URL into the **Address:** text field.

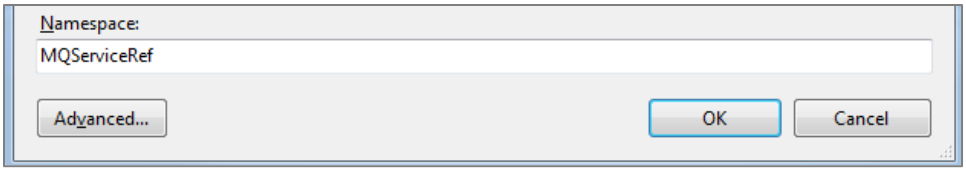


3. Click **Go**.

4. In the **Services:** section, select the **InterfaceServices** line.



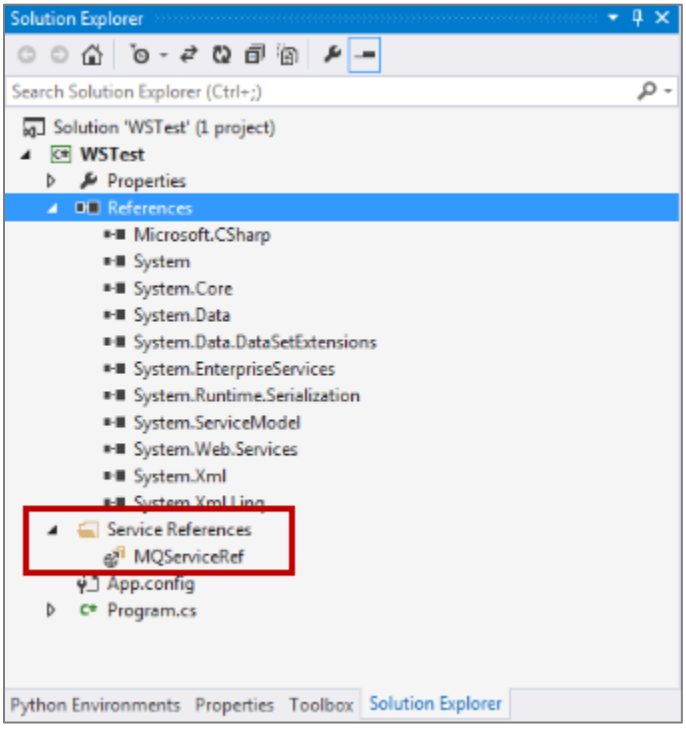
5. Enter the name to be used in the **Namespace:** field.



- The name can be anything that will not conflict with an existing namespace such as MAXQueueSOAP, MAXQueueWebServices, MQWS, etc..

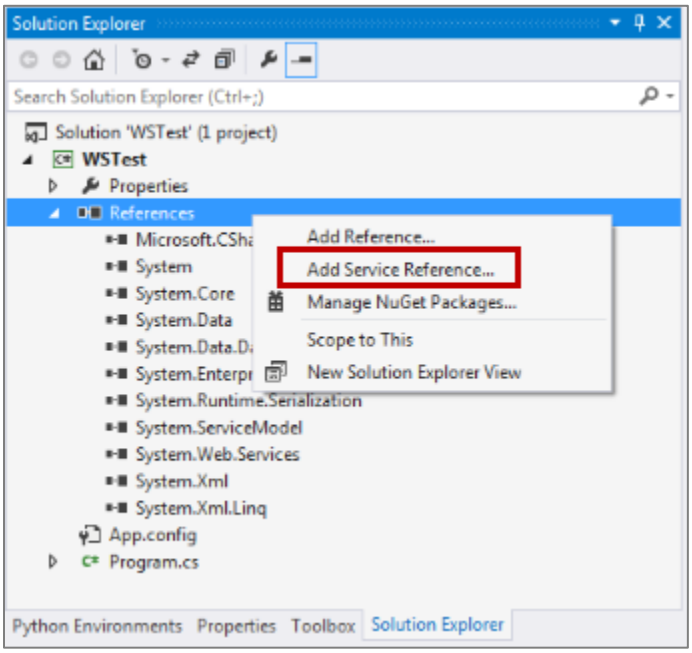
6. Click **OK**.

Services References now shows and can be accessed using this name.



Web Reference Example

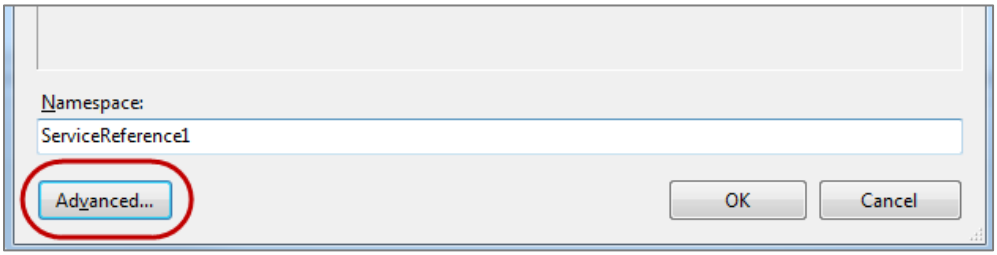
1. Right-click on **Service References**, and select **Add Service Reference…**



   - If the project does not have a Service References folder, right-click on **References** instead.

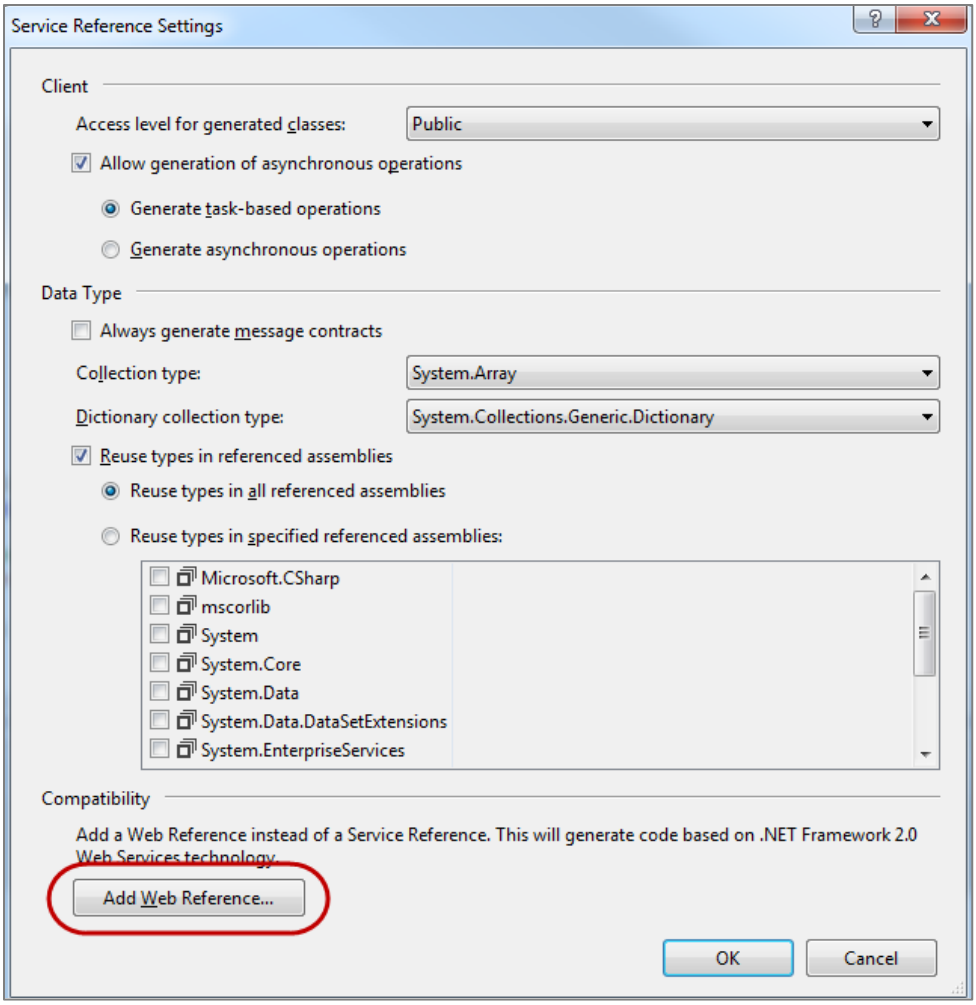
   The Add Service Reference page displays.

2. At the bottom of the page, click **Advanced...**.



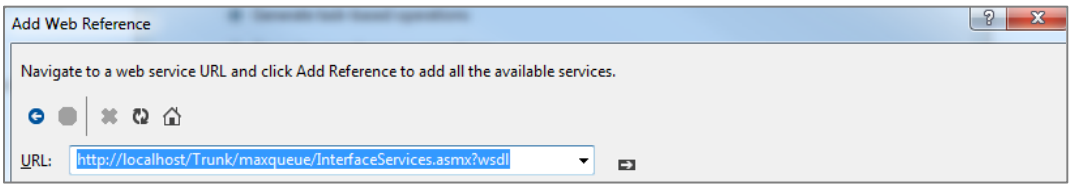
   The Service Reference Settings page displays.

3. Click **Add Web References**.



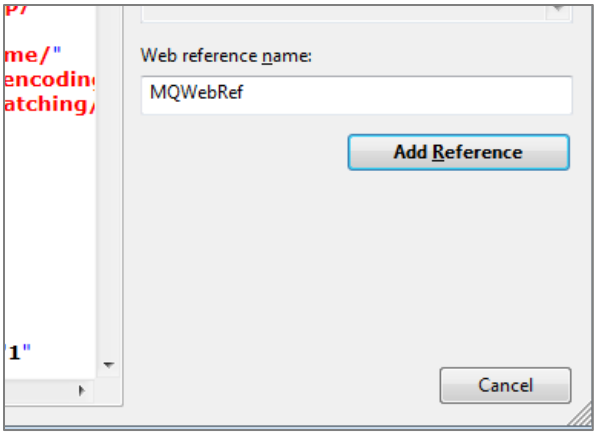
The Add Web Reference page displays.

4. Enter the MAXQueue Interface Services WSDL URL in the **URL** field.



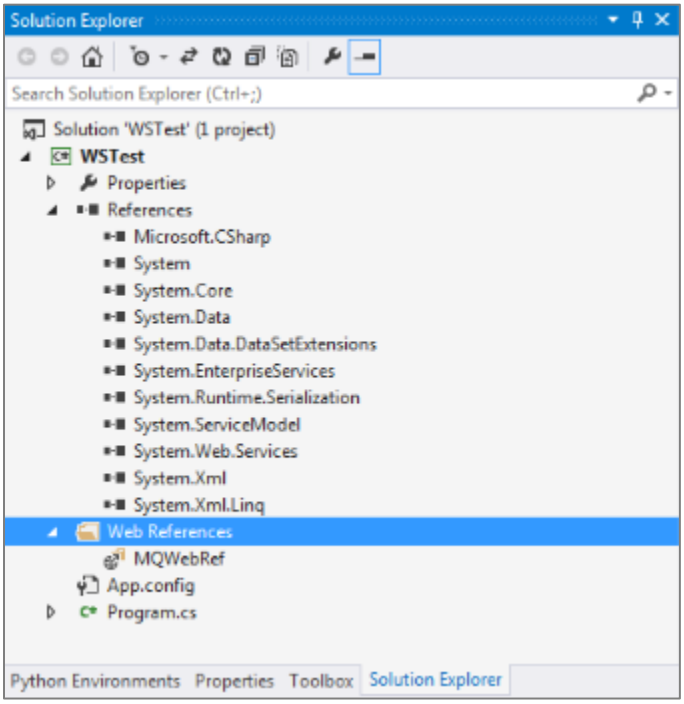
5. Click the arrow to the right of that field.

6. Provide a namespace for the service in the **Web reference name** field..



7. Click **Add Reference**.

   Services References now shows and can be accessed using this name.

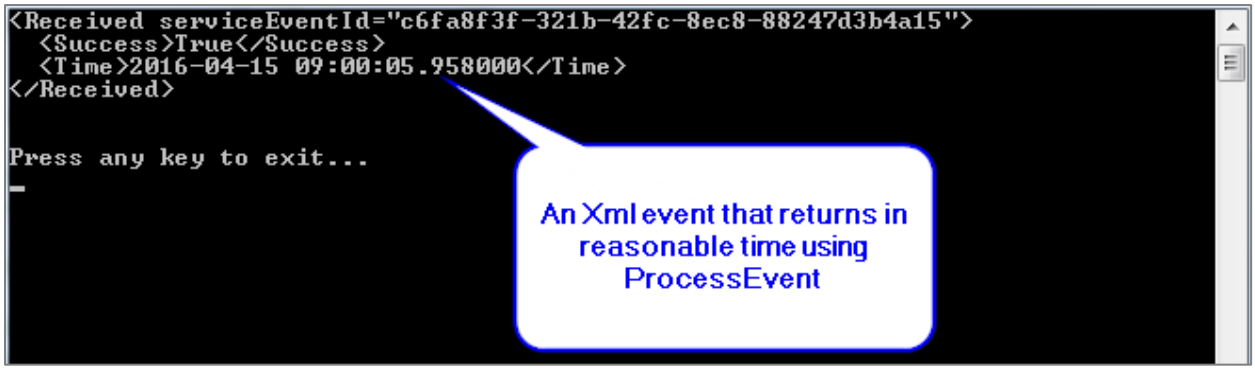


Using Generated SOAP Objects

Once the service or web reference has been added, you can access the generated code.

Service Reference – ProcessEvent Example

```
static void Main(string[] args)
{
    string response = String.Empty;
    response = ServiceReferenceUsingProcessEvent();

    if(response != String.Empty)
        Console.WriteLine(System.Xml.Linq.XDocument.Parse(response).ToString());

    Console.WriteLine("\n\nPress any key to exit...");
    Console.ReadKey();
}

private static string ServiceReferenceUsingProcessEvent()
{
    System.ServiceModel.BasicHttpBinding binding = new System.ServiceModel.BasicHttpBinding();
    System.ServiceModel.EndpointAddress remoteAddress = new System.ServiceModel.EndpointAddress("http://localhost/MAXQueueDesigner/maxqueue/InterfaceServices.asmx");

    //Creates client object that contains the Web Service Methods
    MQServiceRef.InterfaceServicesSoapClient client = new MQServiceRef.InterfaceServicesSoapClient(binding, remoteAddress);

    //In reality this XML would be built from data in the client's system
    string xmlString = "<event id=\"WebServiceExample\"><eventData /></event>";

    string response = client.ProcessEvent(xmlString);
    return response;
}
```



Note: In this example, the generated namespace is called MQServiceRef¬¬

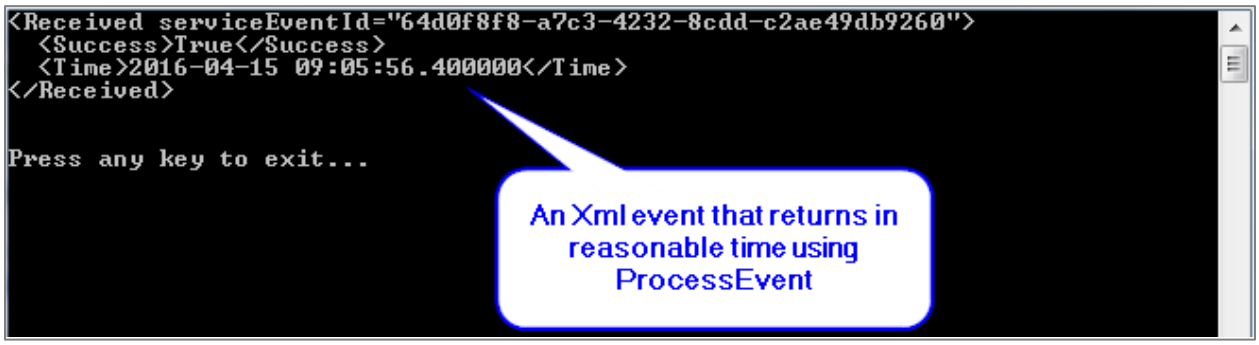
Service Reference – SendMessage, GetResponse Example

```
static void Main(string[] args)
{
    string response = String.Empty;
    response = ServiceReferenceUsingSendMessageGetResponse();

    if(response != String.Empty)
        Console.WriteLine(System.Xml.Linq.XDocument.Parse(response).ToString());

    Console.WriteLine("\n\nPress any key to exit...");
    Console.ReadKey();
}


private static string ServiceReferenceUsingSendMessageGetResponse()
{
    System.ServiceModel.BasicHttpBinding binding = new System.ServiceModel.BasicHttpBinding();
    System.ServiceModel.EndpointAddress remoteAddress = new System.ServiceModel.EndpointAddress("http://localhost/MAXQueueDesigner/maxqueue/InterfaceServices.asmx");

    //Creates client object that contains the Web Service Methods
    MQServiceRef.InterfaceServicesSoapClient client = new MQServiceRef.InterfaceServicesSoapClient(binding, remoteAddress);

    //In reality this XML would be built from data in the client's system
    string xmlString = "<event id=\"WebServiceExample\"><eventData /></event>";

    string eventID = client.SendMessage("WebServiceExample", xmlString);
    string response = String.Empty;
    DateTime start = DateTime.Now;

    //Try and get a response for up to 5 minutes
    while (response == String.Empty && (DateTime.Now - start).TotalMinutes < 5)
    {
        Console.WriteLine(String.Format("Waiting for response (minutes elapsed: {0:0.00})...", (DateTime.Now - start).TotalMinutes));
        response = client.GetResponse(eventID, "A1140785333^5"); //"MAXQ Instance ID"^"MAXQ Service ID"
        System.Threading.Thread.Sleep(5000);
    }

    if (response == String.Empty)
        response = "<Error>Timeout occurred. No response received.</Error>";

    return response;
}
```

Note: In this example, the generated namespace is called MQServiceRef¬¬

Web Reference – ProcessEvent Example

```
static void Main(string[] args)
{
    string response = String.Empty;
    response = WebReferenceUsingProcessEvent();

    if(response != String.Empty)
        Console.WriteLine(System.Xml.Linq.XDocument.Parse(response).ToString());

    Console.WriteLine("\n\nPress any key to exit...");
    Console.ReadKey();
}

private static string WebReferenceUsingProcessEvent()
{
    MQWebRef.InterfaceServices client = new MQWebRef.InterfaceServices();
    client.Url = "http://localhost/MAXQueueDesigner/maxqueue/InterfaceServices.asmx";

    //In reality this XML would be built from data in the client's system
    string xmlString = "<event id=\"WebServiceExample\"><eventData /></event>";

    string response = client.ProcessEvent(xmlString);
    return response;
}
```



Note: In this example, the generated namespace is called MQWebRef¬¬

Web Reference – SendMessage, GetResponse Example

```
static void Main(string[] args)
{
    string response = String.Empty;
    response = WebReferenceUsingSendMessageGetResponse();

    if(response != String.Empty)
        Console.WriteLine(System.Xml.Linq.XDocument.Parse(response).ToString());

    Console.WriteLine("\n\nPress any key to exit...");
    Console.ReadKey();
}

private static string WebReferenceUsingSendMessageGetResponse()
{
    MQWebRef.InterfaceServices client = new MQWebRef.InterfaceServices();
    client.Url = "http://localhost/MAXQueueDesigner/maxqueue/InterfaceServices.asmx";

    //In reality this XML would be built from data in the client's system
    string xmlString = "<event id=\"WebServiceExample\"><eventData /></event>";

    string eventID = client.SendMessage("WebServiceExample", xmlString);
    string response = String.Empty;
    DateTime start = DateTime.Now;

    while (response == String.Empty && (DateTime.Now - start).TotalMinutes < 5)
    {
        Console.WriteLine(String.Format("Waiting for response (minutes elapsed: {0:0.00})...", (DateTime.Now - start).TotalMinutes));
        response = client.GetResponse(eventID, "A1140785333^5"); //"MAXQ Instance ID"^"MAXQ Service ID"
        System.Threading.Thread.Sleep(5000);
    }

    if (response == String.Empty)
        response = "<Error>Timeout occurred. No response received.</Error>";

    return response;
}
```

```
Waiting for response (minutes elapsed: 0.00)...
Waiting for response (minutes elapsed: 0.08)...
Waiting for response (minutes elapsed: 0.17)...
<Received serviceEventId="b2b1401c-554f-4177-aedd-57317313ebc0">
  <Success>True</Success>
  <Time>2016-04-15 09:06:36.491000</Time>
</Received>

Press any key to exit...
```

An Xml event that takes more time, retrieved using SendMessage and GetResponse

Note: In this example, the generated namespace is called MQWebRef¬¬

ProcessEvent Responses

There are three common responses to web service calls to ProcessEvent listed below:

- **No Event Registration:** This means that the service with the given ID is not running properly or that the given ID does not match the service's ID. To troubleshoot this issue, ensure that the following is true:
  - MAXQueue is running
  - The service is enabled
  - The service's **Identifier** field matches the ID attribute in the XML
- **Invalid XML:** This is shown when the XML string passed to the Web Server cannot be parsed into valid XML. To troubleshoot this issue, check for the following in the XML:
  - Open tags without a matching closing tag
  - Extraneous characters between tags
- **Success:** This can be any return that is agreed upon in the specification for the interface, anything from a simple **Success** message to a return of structured data.

SendMessage, GetResponse Details

SendMessage

Input

- Interface Identifier: The Identifier found in the MAXQueue Web Service Entry Point adapter of the interface. (Ex. "WebServiceExample")
- Message: The Xml to be sent to the MAXQueue interface.

```
1.  <Event id="The Interface Identifier">
2.      <The Message Body />
3.  </Event>
```

Output

A unique service event identifier: A GUID. (Ex. 983b5479-9667-4577-a6bc-348456142e85)

GetResponse

Input

- Event ID: The unique service event identifier. (Ex. 983b5479-9667-4577-a6bc-348456142e85)

- Service Name: The MAXQueue Instance ID ^ The MAXQueue Service ID (Ex. A1140785333^1)

Output

- The Xml response: The Xml created and returned by the MAXQueue interface.

```
<?xml version="1.0" encoding="UTF-8"?>
<string xmlns="http://tempuri.org/">
    <Received serviceEventId="983b5479-9667-4577-a6bc-348456142e85">
        <Success>True</Success>
    </Received>
</string>
```

5. Flat Files

Flat files, sometimes also referred to as text, TXT, CSV, or XML files, are another method of exchanging data between the software and client's systems.

Other than line end, file ending characters, and the tab character (when used as a field separating character), each flat file contains only user-readable characters.

From the software perspective, an import file is one in which the client sends data to the software system by creating a flat file; and an export file is one in which software system creates a flat file. Each MAXQueue custom service that uses a flat file must read and write to a file single directory, and that directory must be made available to the MAXQueue service. Multiple MAXQueue custom services can use the same directory, but in that case, the file naming conversion must be agreed to which allows the identification of which file is associated with which service.

It is the client's responsibility to configure the security on the directory to allow the MAXQueue service user the ability to read and delete from that directory. If the file directory is not on the MAXQueue server, then this security becomes more complicated.

Once a flat file is processed, it is moved to an archive. It is the client's responsibility to maintain the contents archive directory. If the files are never deleted, the disk space consumed by these archive files will continue to grow. The location of the archive directory is configurable in the MAXQueue custom service. If the default location is used, then no further security configuration is required. If the archive directory is changed, then it is the client's responsibility to configure the security on the archive directory to allow the MAXQueue server user the ability to create and write for that directory.

File Transfer

The following is supported for file transfer:

1. Direct connection to a local or shared folder with proper access rights.

   Example:

   - Relative to MAXQueue Server directory: .\Input\MyEvent.csv
   - Full path specified: C:\InputDirectory\MyEvent.csv
   - Shared Network Drive Path: \\<hostname>\directory

2. FTP and SFTP (v.15 forward) configured with a Username and Password

⚠ For any other methods or files, please provide full details to analyze and discuss integrating options based on your system's functionality.

File Types

Type	Summary
CSV	Comma separated value. This is actually any file where fields are separated by a specific character. Common separator values are comma, tab, caret (^), and pipe (|). In order to allow string data to contain the separator character, all string data should be surrounded by double quotes.
TXT	Values are positional based. For instance, positions 1-4 contain field1; positions 5-9 contain field2.
XML	String data conforming to minimum standard XML 1.0.
XLS, XLSX	Clients can export Excel data into CSV or TXT files.
Others	Other formats may be able to be processed, but only after client provides a sample for analysis.

Specification Details

- CSV and TXT file may optionally contain a header row, but this row must conform to the format (separator character or position) of the underlying data rows. XML files cannot contain header rows.
- The specification for the file should also include for numeric values if negative signs are leading or following. Decimal places can be explicit (125.25) or implied (12525); if implied, the implied number of decimal places needs to be specified (two in this example).

Notes

- Flat files are usually simple integrations to create and modify which works well for simple input and output of data to external systems.
- Data can be easily analyzed for troubleshooting and other investigation.
- Data can be shared within other groups in an organization.
- Output is usable as-created (a flat file can serve as its own final output, and be analyzed in common external programs such as Microsoft Excel).
- Many systems are set up to handle input and output of flat files out-of-the-box and do not require additional programming on the client's side of the integration. This is useful if a client has minimal programming access to or from their other systems, or output coming directly from an external source.
- Potential complexities regarding use of this method:

- MAXQueue services cannot easily pull a subset of the file. The whole file must be pulled each time. This makes the use of this method best suited to use with batch-processing style integrations which happen on a set timer.

  ⚠ SFTP is not supported prior to version 15.0.

6. Other

- The most common standard data exchange methods are outlined in this document. We have also created standard methods with other technologies and formats, such as MSMQ and Oracle Advanced Queuing.
- Generally speaking, a custom solution for integration can be created with any technology or format.
- If you are using some other technology, please provide customer support with the details to analyze, to discuss integrating options based on your system's functionality.