QUEST PROXY WEB SERVICE

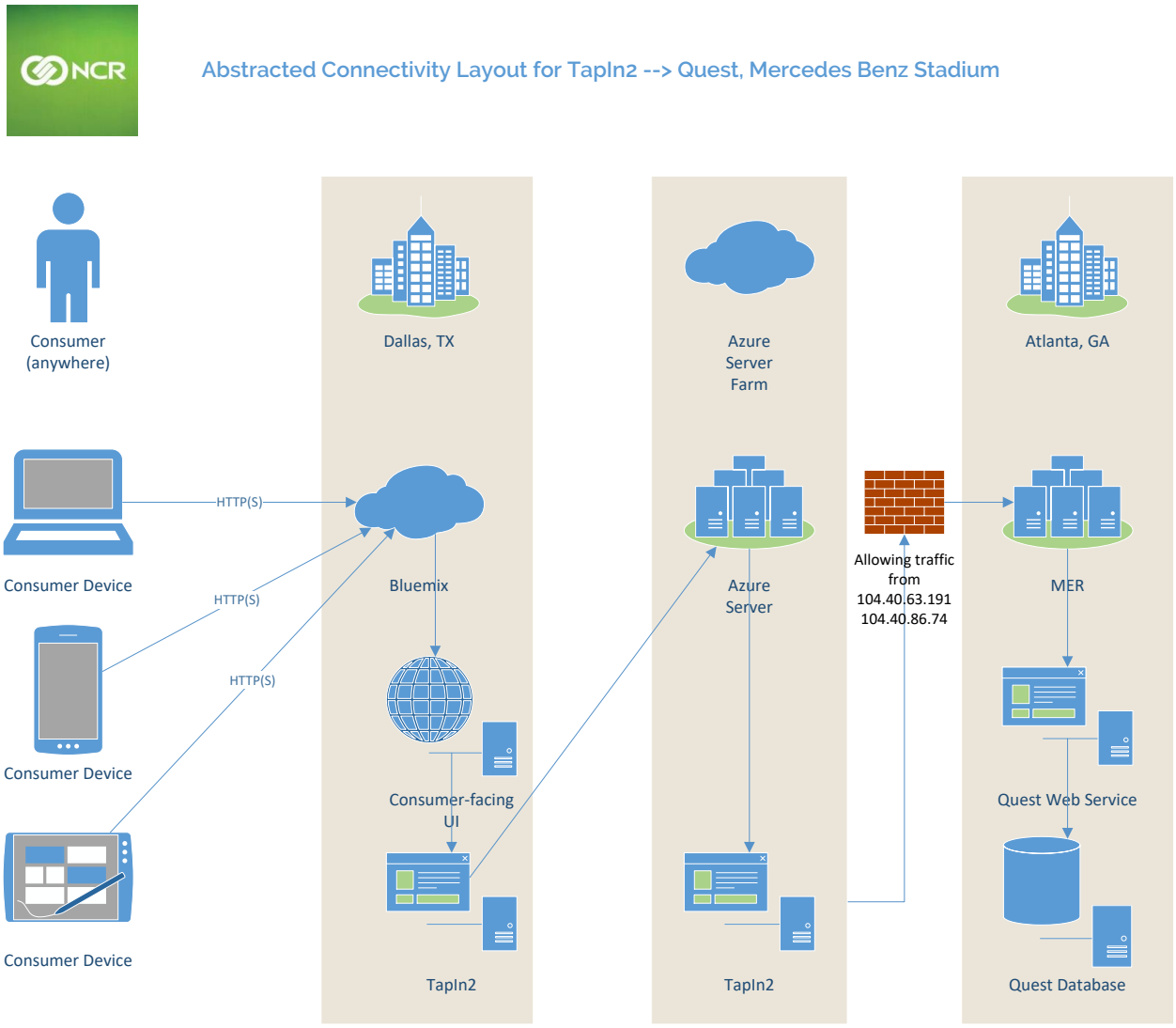
Thomas, David G

NCR CORP

Table of Contents

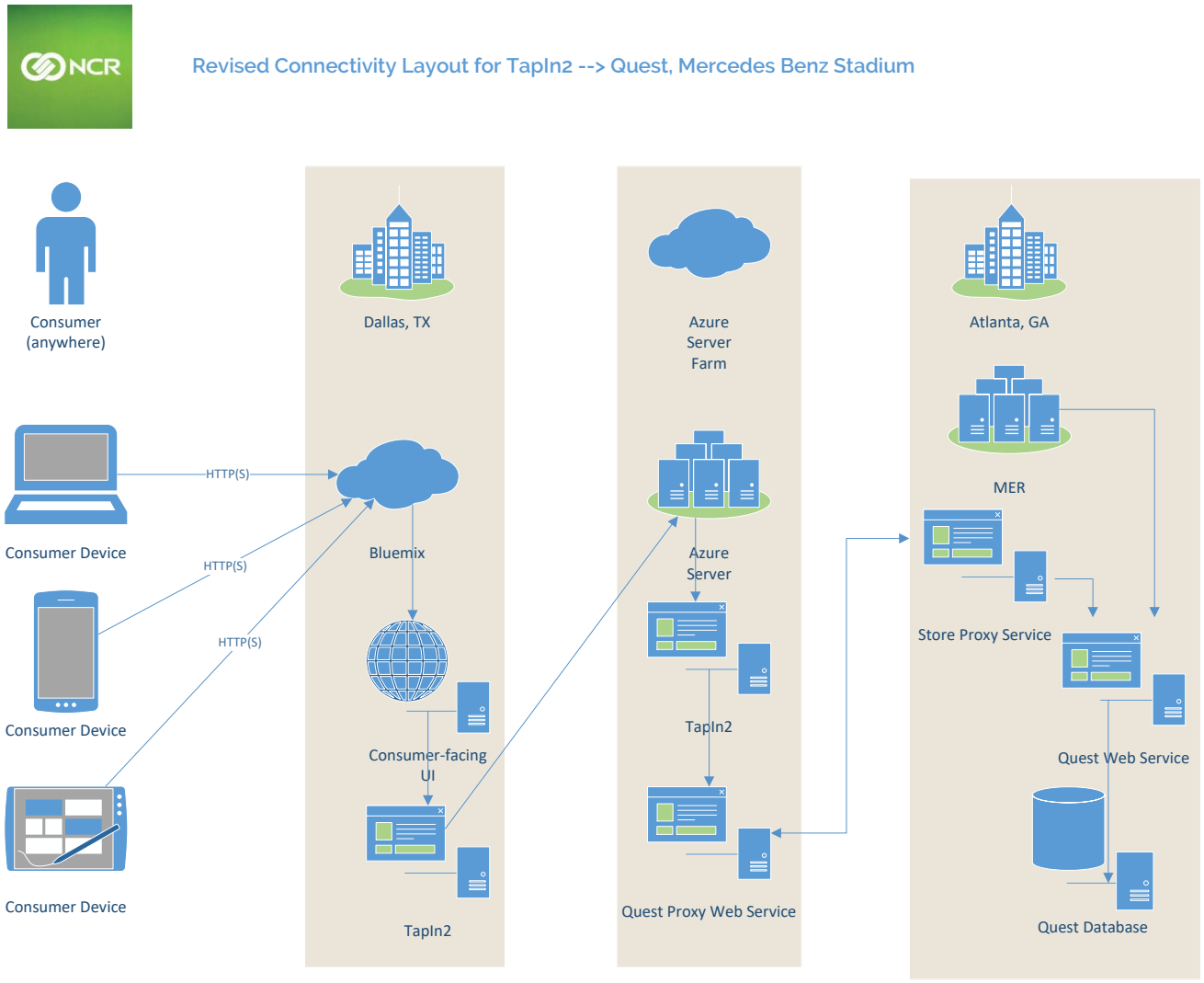
Overview

The following diagram illustrates the existing system. Note that Tapin2 communicates with the Mercedes servers directly, and firewall rules are enforced to restrict access so that only the Tapin2 server has access to the Quest Web Services.



The system was updated so that the network connection is initiated within MER (store server). On the store side, a Windows service initiates a connection with the Tapin2 server. On the Azure server farm side, a new web service is implemented that provides indirect access to the QuestMenu Web Service, without the need to connect into the Mercedes server. Tapin2 should be configured to access the local web service in place of the web service at the store. The local web service acts as a proxy so that each call to the proxy service will execute on the Mercedes server. This is done transparently, so no changes to Tapin2 are necessary other than configuring it to use the local web service.

A diagram illustrating the updated system is shown below. Note that it is the same as the prior diagram with the addition of two services. A more detailed diagram of the changes is detailed on the next page.

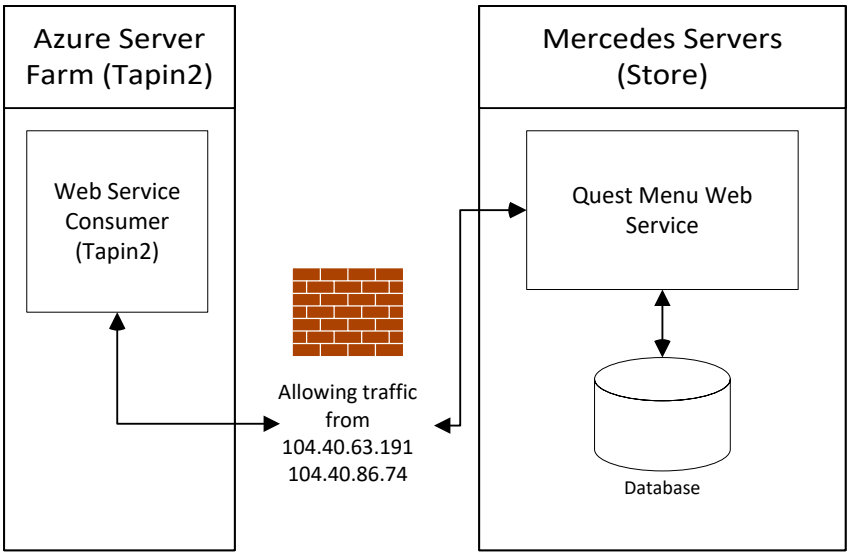


Quest Web Service Access - Detailed Functional View

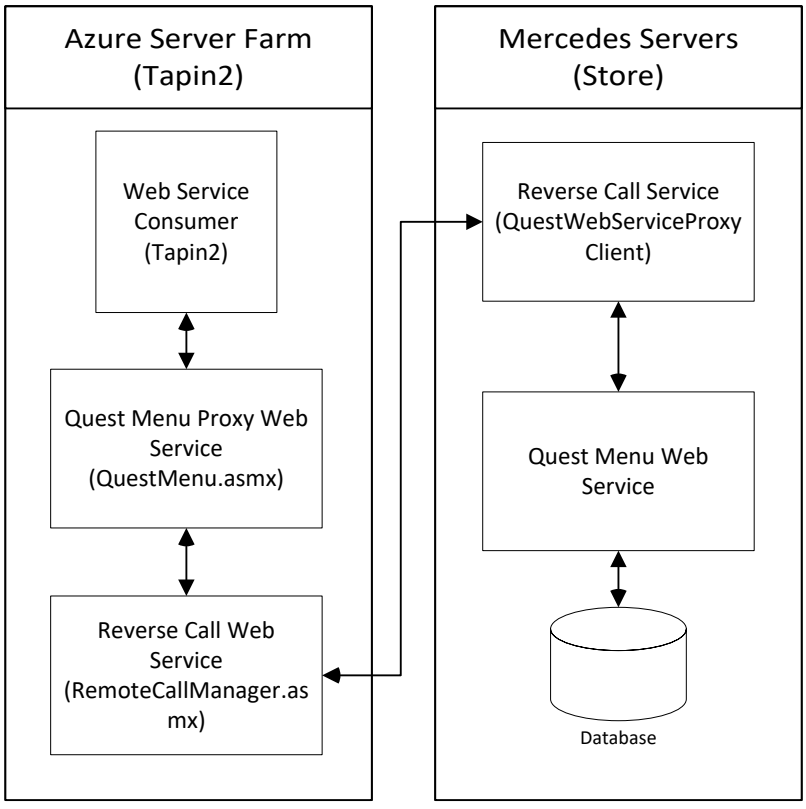
The following diagram zooms in on the functional differences between the prior system and the new one.

Functional Diagram – Quest Web Service Access

Original System



With Quest Proxy Web Service



Proxy System Components

Quest Menu Proxy Web Service (QuestMenu.asmx)

The QuestMenu web services are accessed using a proxy that imitates the actual QuestMenu web service. Calls to the QuestMenu proxy are executed using the QuestMenu web service on a remote server, and results are returned as if the actual QuestMenu web service was called directly.

Reverse Call Web Service (RemoteCallManager.asmx)

The RemoteCallManager web service is not called directly by the application. It is installed along-side the Quest Menu proxy web service, and accessed remotely by the client side services that are installed in the store. It provides communication to implement a reverse-call – which is a call using a connection initiated by the store server.

Reverse Call Client Service (QuestWebServiceProxyClient.exe)

This service runs on the store server where the QuestMenu web service is installed. It initiates a connection that can be used to retrieve the next call, along with call parameters, from the proxy service. The call will be processed by invoking the QuestMenu service, and the return parameters will be passed back to the proxy with a subsequent connection.

There is also an application, QuestProxyClientApp.exe, that provides the same function without needing to start a service. It can be run using the command line, or it can be launched by the service to provide additional listener threads.

Proxy Web Service

Proxy Web Service Installation

This component must run on an IIS server.

Copy all files and subfolders into a folder where the application is to reside. It is recommended to create a folder for proxy services and place the store into its own folder within the proxy services folder. The name of the store folder could be any identifier that is convenient. For example, it could be the name of the stadium (e.g. Mercedes), the customer (e.g. AtlFalcons), or any other name that identifies the web service. The folder name will be the default service name, so it is best to use a folder name that helps identify the purpose of the folder.

To install, the following commands may be used (assuming z: is the installation drive)

```
mkdir C:\QuestProxyService
mkdir C:\QuestProxyService\AtlFalcons
xcopy /s z:\Install\WebSvc C:\QuestProxyService\AtlFalcons
```

Using the IIS administration tool, add an IIS application, and set the root folder of the application as the folder copy. The name of the application could be any identifier that is convenient, but using the same name as the folder name can simplify maintenance. In the above example, the application will be called AtlFalcons.

Note: After IIS installation, there should be a Services subfolder within the application folder – do not point the application into the Services folder. Instead, use the folder that contains the Services folder.

If necessary, the settings for the web service can be adjusted. However, the defaults should be suitable, so configuration changes at this stage should not be necessary. See Web Services Proxy Configuration for more details regarding available parameters.

The next step is to install the store service on the computer where the Quest Web Service resides, and configure that service to use the proxy URL. After the store service has been installed and configured on the store server, the system can be tested. See Testing the System for tools and tips for system testing.

Multiple Store Support

Multiple instances of the proxy web service may be run on the same web server, and each instance can supportsa different store/customer. To implement multiple stores, create a subfolder for each store, and copy installation files into each folder.

For example, the following commands may be used (assuming z: is the installation drive)

```
mkdir C:\QuestProxyService
mkdir C:\QuestProxyService\AtlFalcons
mkdir C:\QuestProxyService\AtlBraves
xcopy /s z:\Install\WebSvc  C:\QuestProxyService\AtlFalcons
xcopy /s z:\Install\WebSvc  C:\QuestProxyService\AtlBraves
```

After the files are copied use the IIS administrative tool and add each folder as a separate IIS application.

If necessary, view and/or edit the Web.config and/or QuestProxy.config file in each application folder.

Proxy Web Service Configuration

The Proxy Web Service IIS application uses the standard web configuration file (web.config) in the folder where the web service is installed. Settings specific to the proxy web service are in a separate file named QuestProxy.config, which is also a standard .NET xml configuration format. The QuestProxy.config file is installed with placeholders and/or defaults for all available settings. Note that this file is separated from the web.config file so that it may be edited/updated while the web service is running, without causing the web service to restart.

Most values in the WebProxy.config can be updated while the web service is running and will file take effect soon after changed (See the ConfigInterval parameter). Log file names and folders may be changed while running, but will not take effect until next time the web service restarts. See Restarting the Web Service for more information.

Settings Value Name	Purpose
ServiceName	Specify a name for the service. It is recommended to use the same name as the IIS Web Service name. This name must be a valid file name, and **must also be unique** if multiple proxy web services are installed because it is used internally to reference the web service across sessions. This name will also appear in the web store log to identify which web service is being accessed. If not specified, the folder name where the web service is installed will be used.
ConfigInterval	Most parameters may be adjusted while the service is running. This parameter defines how often to check for an updated configuration. Setting this value to 0 will disable config reloads. Parameters that are not updated after the service starts are Log file names and Timeouts. All other parameters, including log options, can change while the service is running.
LogFolder	Specify the folder to use for all log files. Note this is for convenience – it can be over-ridden if a log file specification includes the full path name.
LogEnable	Set to "false" to disable all event log, trace log and file log output. This does not affect XML log output.
LogFile	Specify a file to log basic activity such as call processing stages.
LogFileLevel	Specify the message level for filtering output to the log file. Specify one of the following: None, Error, Warning, Information or Verbose. Specify “None” to disable the file log. Other values may be used to filter what is included in the log file.
LogXmlFile	Specify a file name where SOAP XML is to be logged. This file will be in XML format. See the LogXmlOption value for additional information.
LogXmlOption	Specify one or more of the following options separated by comma. **IN** Log the SOAP input XML for every call **OUT** Log the SOAP output XML for every call **ERR** Log error messages and SOAP input XML for calls that fail to execute **FMT** format XML so it is more readable in text view The default config value is “err, fmt” so that SOAP requests that fail are logged.
CallQueueTimeout	Specify timeout (in seconds) for call to begin execution. If there are too many calls waiting to be run, and a call sits in the queue longer than this timeout then the call will be cancelled and an error will be returned to the caller.
CallTimeout	Specify timeout (in seconds) for call execution. This includes time spent in the waiting queue and time spent running. If a call takes longer than the timeout to run, an error will be returned to the caller. Note the call will continue to run after the timeout, but its return value will not be available, and the call will effectively be abandoned.
ListenerTimeout	Listener connections normally wait for an incoming call before returning. This parameter specifies how long to wait (in seconds) before returning when no calls are available. This parameter should be significantly less than the web service timeout (normally 100 seconds) for the application to run smoothly. If a web service timeout is detected, this parameter will automatically be adjusted to a value that is between ½ and ¾ of the actual web service elapsed time.
MaxWaitingCalls	This limits the number of calls that can be in process or queued. It appears there is an OS limit on the number of concurrent calls into a web service, and if the limit is exceeded, operation of the proxy web service will be interrupted. To keep this from happening, the call limit should be enforced. On a workstation, it should be no more than 6. On a server machine the limit is much higher, but there can be performance consequences is it is higher than 16. The default value depends on whether it is a workstation or server machine.

Settings Value Name	Purpose
FixNewLine	If a SOAP argument string contains new-line characters (CR/LF) the Web Services system removes the CR character and keeps only LF. Any response string that includes new lines can also be affected by this. One example is the SOAP exception response, which contains a multi-line error message. Setting the FixNewLine option to "true" will compensate for this and cause the proxy to convert each LF character in the SOAP exception response to CR/LF. Set to "false" to disable this behavior. This value should be true if the SOAP exception needs to be returned exactly as it is with the non-proxy web service.
PostToWebServiceUrl	This parameter allows temporary bypass of the reverse calling mechanism by directly forwarding each call to a specific web service URL. This may be used for diagnostic purpose, or to access an alternate web service in special circumstances.

The web site configuration file can include other standard .NET configuration parameters. The following are notable to mention.

Event Logging

Standard .NET event logging and trace logging parameters may be specified in the `<system.diagnostics>` section of the web.config file. Note that caution should be exercised wen editing the web.config file. See Editing the web.config file for details.

.NET event logging uses the "ProxyServer" event source. Standard trace listeners may also be added if desired. However, it appears that Windows event Log listener does not operate correctly on the web server, probably due to details in how the Windows event log extracts resourced from running EXE, and IIS does not contain the required resources. An entry for the Windows event log is included in the default config file, but it is commented-out. It can be enabled if the problem with using the Windows event log is corrected,

.NET trace output is also available. This allows monitoring trace messages using a debug viewer such as Dbgview.exe. See Monitoring and Logging for more details on using log files and real-time monitoring of log output.

Editing the web.config file

The web service will be immediately restarted if the web.config file is updated. This is not recommended while the service is running, or as a minimum, should be done with extreme caution. See Restarting the Web Service for details.

Note that the QuestProxy.config file contains the proxy parameters that control proxy operation, and is separate from the web.config, so it may be freely edited. Most parameters in the QuestProxy.config file are reloaded at regular intervals, so they will take effect soon after they are made. See Proxy Web Service Configuration for details.

Restarting the Web Service

A web service will normally timeout after it has been idle for a while. However, with the proxy service, listeners are continuously being created to process the next incoming call. The listener is considered by IIS to be "activity", so the proxy service is never idle, and will not time out while it has listener processes running.

Updating the web.config file will cause the web service to restart immediately. This should be done with extreme caution – preferably at a time where the web service is idle (no calls in progress, and none expected soon). If calls are in progress when the web service is restarted, those calls will be terminated. The log will normally include entries to verify call termination, or in some cases the log may indicate the call timed out. In either case, the call fails and the caller will receive a SOAP exception error.

Store Service

Store Service Installation

This component should normally be run on the same IIS server where the QuestMenu web service runs. However this is not a requirement. It can be installed and run on any machine that can access both the proxy RemoteCallManager.asmx web service and the store QuestMenu.asmx web service.

To install, create a main folder for the client and two subfolders; One subfolders for the service, and one for the non-service application.

For example, the following commands may be used (assuming z: is the installation drive)

```
mkdir C:\QuestProxyClient
mkdir C:\QuestProxyClient\App
mkdir C:\QuestProxyClient\Service
xcopy /s z:\Install\StoreSvc\App  C:\QuestProxyClient\App
xcopy /s z:\Install\StoreSvc\Service C:\QuestProxyClient\Service
```

After copying the files, install the .NET service so that it can be started as a Windows service. To do this the .NET InstallUtil.exe utility is needed. It may be found in the .NET installation folder. Run that command from the service folder, and specify the service executable name as its command line parameter. For example, the following command line will install the service with .NET version 4.0.30319.

```
chdir /d C:\QuestProxyClient\Service
c:\Windows\Microsoft.NET\Framework\v4.0.30319\InstallUtil.exe QuestWebServiceProxyClient.exe
```

If the service needs to start automatically, run the system admin “Manage” utility, and change the service start mode.

The final step is to edit the config files in both the Service and the App folders. As a minimum, set the ServiceName, ProxyService and PostToService parameters.

The ServiceName should reflect the name of the store, and its value will be displayed in logs.

The ProxyService parameter should be the URL of the proxy service that will access the local web service.

The PostToService should define the local web service URL, or just the web service application name if it is installed on the local computer.

See Store Service Configuration for additional details and configuration options.

After the store service and proxy service are installed on their respective servers, the system can be tested. See Testing the System for tools when the system is ready to test.

Store Service Configuration

The Store Service uses configuration in file QuestWebServiceProxyClient.exe.config which is standard .NET xml configuration format. The file is installed with placeholders and/or defaults for all available settings. The non-service app (QuestProxyClientApp.exe), config file Is named QuestProxyClientApp.exe.config.

The configuration file has an application section `<userSettings>` dedicated to setting up the proxy client. The following configuration is available in that configuration section.

Settings Value Name	Purpose
ServiceName	Specify a name or description of the service. This name will appear in the proxy web service log to identify which store is being accessed. It may also be used a part of the log file name.
PostToService	Specify the local web service that is to execute requests. The whole URL is not necessary, just the IIS application name. The *Questmenu.asmx* portion of the URL should **not** be specified because this service can handle web services other than QuestMenu – it will always append the specific Quest Web Service name (e.g. QuestMenu.asmx) to the URL based on the original call.
ProxyService	Specify the URL where the Quest Proxy Web Service Call manager (RemoteCallManager.asmx) is installed. It should include the full URL. It may optionally omit the RemoteCallManager.asmx file name, which will be appended if missing.
MaxThreads	Set the total number of processing threads. 2 threads are required for communications. Additional threads may be used to allow multiple calls to be executed simultaneously. The default is 8 total threads. This value may be set to 0 to disable service listeners if ListenerAppCount is set to launch additional listener processes.
MaxActiveConnections	This value sets the maximum number of Web Service connections. It cannot exceed the "maxconnection" parameter in the <system.net><connectionManagement> section of the config file. If that section/parameter is not defined this parameter must be 2 (see Limits for details).
MaxListenerConnections	This value defines the number of listener connections that are used. If multiple calls arrive at the same time and there is only one listener, the second call cannot start until another listener is ready. Setting this value above 1 increases the odds that another listener is already waiting and ready when the second call arrives. This value must be less than the MaxActiveConnections value so that some connections are available to return call responses.
LogFile	Set the path and file name for the log file. Note that the file name will not change after the program has started. The file name can contain environment values to be expanded. Environment values for ServiceName , Date (YYYYMMDD format), Time (HHMM format) and ProcessId are automatically added so that they may be included as part of the file name.
LogFileEnable	This may be used to enable or disable the log file.
ListenerAppCount	Specify the number of additional listener processes that are to be used. This is not normally necessary, but it could be used if it is required to handle more simultaneous requests than a the maxconnection configuration setting would normally allow.
ListenerAppPath	Specify the full path name where the listener application, QuestProxyClientApp.exe, is installed. Note that when the listener app is launched by the service, Proxy, Post URL and ServiceName parameters are taken from the service configuration. All other parameters are taken from the application configuration file (QuestProxyClientApp.exe.config).

The application configuration file can include other standard .NET configuration parameters. The following are notable to mention.

system.net

the <system.net> configuration file section Is required if the proxy service is to access more than 2 simultaneous network connections. The proxy client can work with 2 connections – one for the listener and one to return call responses – but it can provide higher throughput and lower call delays if additional connections are used. The default <system.net> section enables only two connections, but the configuration installed with the proxy client supports 10 connections. This can be further increased if desired, but the increase will not have effect unless the MaxActiveConnections application parameter is also increased. Similarly, if the number of connections are decreased, MaxActiveConnections parameter must be decreased. If the maxconnection parameter is removed, or that section is removed from the config file, the default is 2, so the MaxActiveConnections must be changed to 2 for the program to operate correctly.

The <system.net> configuration section that is installed with the proxy client is as follows:

```
<system.net>
    <connectionManagement>
      <add address="*" maxconnection="10" />
    </connectionManagement>
  </system.net>
```

Event Logging

Standard .NET event logging and trace parameters may be specified in the <system.diagnostics> section of the application config file

Event logging uses the “ProxyClient” event source. Standard trace listeners may be added if desired. The default config file adds a trace listener for the windows event log, so that errors will be included in the Windows “Application” log.

Standard .NET trace output is also available. This allows monitoring trace messages using a debug viewer such as Dbgview.exe. See Monitoring and Logging for more details on using log files and real-time monitoring of log output.

Stopping and Restarting the service

The service can be stopped and restarted without causing SOAP errors or lost calls. While the service is stopping, new calls will be queued by the proxy web service, and calls that are in progress will complete before the service is stopped. Queued calls will be processed when the service restarts, so the service should be restarted soon after it is stopped to provide effectively continuous operation. If the service is not restarted soon after it is stopped, calls can fail due to time out.

Monitoring and Logging

All components of this system support standard .NET event log and trace logs. The default config files include placeholders for these logs which are mostly empty, to indicate where additions may be placed. The only trace item added by default is to write error messages to the Windows Event log when possible (only on store based services).

In addition to trace logs, an internal file log may be enabled. This log contains the same entries that are written to the event log and trace logs – but additional information is included with each log record.

One final log is the XML log file, which is available on the proxy server. That log file is in XML format, and can include SOAP XML (input and/or output), and error messages. It can also be configured to only write error messages along with SOAP XML for calls that failed to execute due to error.

Real-Time monitoring

Real time monitoring is possible using the Dbgview.exe program available from SysInternals.com or another, similar debug viewer. To use, run the DbgView.exe program. Options in that program allow color coding messages, filtering, and saving to a file.

This capability is included by default with .NET applications. To disable this capability, edit the config file and uncomment the line that contains “<clear/>”. If the <clear/> line is included, it will remove default listeners, thereby removing the debug viewer listener.

Troubleshooting

When troubleshooting, it is best to see all messages (verbose as well as Information level). It is also best to know what is expected. See the Reverse Call Log Details section for a description of a typical reverse call log. That section illustrates information messages that are expected. Verbose level messages are also included in a normal log to fill in details between the items listed at Information level.

Reverse Call Log Details

Call processing steps may be understood and verified by examining Information level messages that are present in the log files. A typical call log is illustrated below.

The Proxy Web Service log contains:

```
10:21:28.391 (4108,6216) I Call #1 Received: [QuestMenu.RequestVersionInformation()] 329 bytes.
10:21:28.392 (4108,8028) I Call #1 Begin processing on connection #1
10:21:28.473 (4108,4336) I Call #1 Results received.  511 bytes on connection #1
10:21:28.476 (4108,6216) I Call #1 Processing complete.  Returning results.
```

The Store log contains:

```
10:21:28.402 6080 4596 I [1] Call #1 Received: QuestMenu.RequestVersionInformation() 329 bytes.
10:21:28.413 6080 8044 I [2] Begin Reverse Call - Connect to AtlBraves
10:21:28.463 6080 4596 I [1] Call #1 Post complete.  Return 511 bytes.
```

Merging the two logs in time-order gives a more complete picture. The steps are numbered to facilitate explanation below the log entries.

Proxy Web Svc

```
(1)10:21:28.391 (4108,6216) I Call #1 Received: [QuestMenu.RequestVersionInformation()] 329 bytes.
(2)10:21:28.392 (4108,8028) I Call #1 Begin processing on connection #1
```

Store

```
(3)10:21:28.402 6080 4596 I [1] Call #1 Received: QuestMenu.RequestVersionInformation() 329 bytes.
(4)10:21:28.413 6080 8044 I [2] Begin Reverse Call - Connect to AtlBraves
(5)10:21:28.463 6080 4596 I [1] Call #1 Post complete.  Return 511 bytes.
```

Proxy Web Svc

```
(6)10:21:28.473 (4108,4336) I Call #1 Results received. 511 bytes on connection #1
(7)10:21:28.476 (4108,6216) I Call #1 Processing complete.  Returning results.
```

(1) The call is received by the proxy, and if a listener is waiting, the call is immediately passed to the listener. The calling thread is suspended until the listener is done processing the call.
(2) The listener is activated and returns call parameters to the store
(3) The Store receives the call parameters and posts to its local web service
(4) Another listener is started while the current call is being processed - to quickly handle another call when it arrives
(5) The local web service call is completed. Results are retrieved and passed back to the proxy web service
(6) The proxy web service receives the results and activates the suspended calling thread
(7) The calling thread receives results and returns them to the original caller

Testing the System

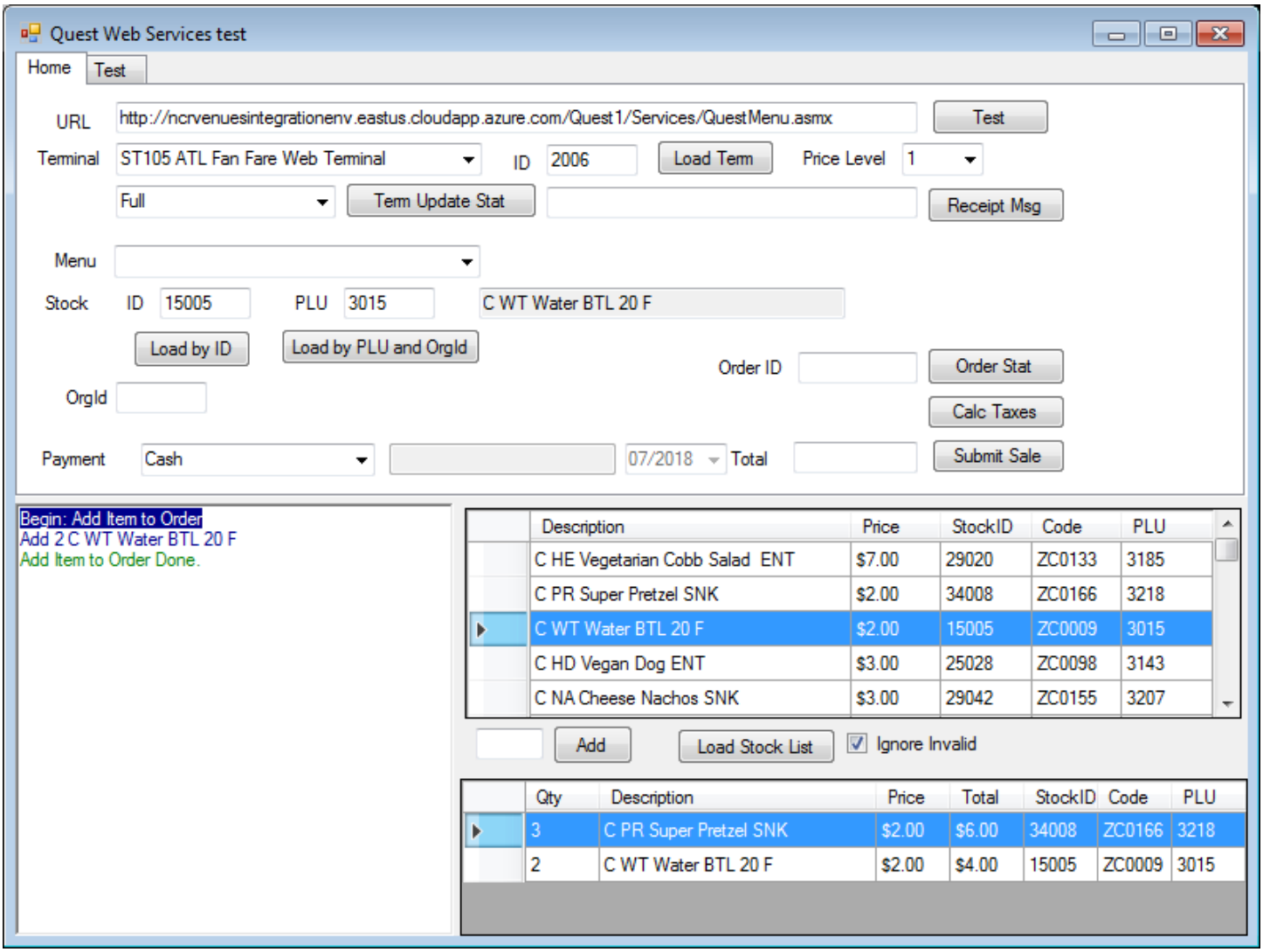
The system is delivered with two test programs.

Note: Do not use the web browser to test the proxy service. The web browser tool is not compatible with the proxy. To test use one of the included tools, or any other tool that uses the QuestMenu API.

Interactive Ordering App

The QwsTest.exe program is a generic/simple ordering system that uses the QuestMenu API. It can be used to exercise the API and verify results for both the QuestMenu API, and for the Proxy web service (which also implements the QuestMenu API). It Is not intended as a complete end-user ordering system. It is more of a sample for those who are familiar with the API or want to be familiar with the API

A sample screen follows.



To use the app, first set the URL to the QuestMenu web service. The URL can be that of a QuestMenu web service, or of a QuestMenu proxy web service. The two web services are indistinguishable as far as the calling application is concerned.

After entering the URL, it can be tested with the **Test** button. That button will simply retrieve the QuestMenu version and display it in the lower-left results area.

After testing the URL, select a terminal from the list (or enter its ID) and press **Load term**. When selecting a terminal from the list, the program will use the RequestTerminalList API function to retrieve a list of terminals and populate the

list box. Or just enter a terminal number. Either way, the **Load Term** button needs to be pressed prior to continuing so that the complete terminal data is retrieved.

After loading the terminal, other functions on the screen become available. List menu items, load stock list, or load stock items using data as provided with the QuestMenu web service. Also, select stock items, calculate taxes for those items and place the order.

Note that some input fields on the screen are needed to complete the API call (e.g. **Load by PLU and OrgId** requires the OrgId be entered). Other API calls use data from a prior call to simplify usage.

SOAP Post App

PostSoap.exe is a command line utility to post SOAP xml to a web service. The following features are available:

1. Display input and/or results XML on the console.
2. Output is color coded for clarity. Error results are displayed red for easy identification.
3. Save results of a single SOAP request to a file.
4. Displayed/stored SOAP XML is normally formatted for improved readability, but raw XML results are also available.
5. Send multiple files in a single command line
6. Send the same file to multiple web services in a single command line. This allows posting to the actual web service and the proxy web service to make sure the results match. To aid this, the comparison is done, and result is displayed automatically.
7. Run the request (or list of requests) in multiple threads simultaneously to simulate load
8. Repeat multiple times to maintain load
9. When running multiple threads, the keyboard may be used to pause, stop or change some settings while it is running.
10. Basic help/usage text may be displayed – run the command with no parameters to see help text.

To test the web services using PostSoap, change into the test folder and run the following command line (*WebServiceuRL* is the URL of the QuestMenu web service to be tested, and the XML file can be any SOAP compatible request. See the xml sub-folder for more sample SOAP XML files.

```
PostSoap getVersion.call.xml WebServiceuRL
```

For example:

```
PostSoap getVersion.call.xml http://localhost/AtlBraves/Services/QuestMenu.asmx
```

Or, the shortcut if it is QuestMenu on the local machine:

```
PostSoap getVersion.call.xml AtlBraves
```

To make sure the proxy works the same as the actual web service, specify both on the command line. Both web services will be called and the responses will be compared. Note: The following uses a shortcut for both URLs – the full URL can also be used.

```
PostSoap getVersion.call.xml AtlBraves WebServer/QWS-2.30.5.0
```

To execute multiple files add them to the command line.

Note that the command line can identify URLs verses file names, but the distinction is not perfect. If the URL contains / or the path contains \ identification will be correct. Otherwise, a name without extension is considered a URL, one with an extension is a file name. To access web service named QWS-2.30.5.0 on the local machine, the simple shortcut QWS-2.30.5.0 will not work. Use “localhost/QWS-2.30.5.0” instead. Also to send a file named MySoapInput, specify “.\ MySoapInput” as the file name (or give the file name an extension).

For a more complete and updated list of options the program offers, run it with the /? Option.

```
PostSoap /?
```

Note: When testing multiple threads, make sure the PostSoap.exe.config file is present, and that its maxconnection parameter is at least as high as the number of threads being used. Otherwise, PostSoap will not have sufficient connections, and the timing numbers will be deceptively high.