Using TCP Adapters with Ensemble

Version 2018.1
2018-12-14

InterSystems Corporation 1 Memorial Drive Cambridge MA 02142 www.intersystems.com

Using TCP Adapters with Ensemble
Ensemble Version 2018.1 2018-12-14

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Response Center (WRC)
Tel: +1-617-621-0700
Tel: +44 (0) 844 854 2917
Email: support@InterSystems.com

Table of Contents

About This Book

TCP is a transport layer protocol that supports many types of higher-level communication protocol. This book describes how to use the Ensemble TCP adapters, which support several simple types of TCP, with inbound and outbound versions of each type.

This book contains the following sections:

- Using the TCP Inbound Adapters
- Using the TCP Outbound Adapters
- Special Topics
- Reference for Settings

For a detailed outline, see the table of contents.

The following books provide related information:

- *Developing Ensemble Productions* describes how to create Ensemble productions in general.
- *Ensemble Best Practices* describes best practices for organizing and developing Ensemble productions.
- *Configuring Ensemble Productions* describes how to configure the settings for Ensemble productions, business hosts, and adapters. It provides details on settings not discussed in this book.

For general information, see the *InterSystems Documentation Guide*.

1

Using the Inbound TCP Adapters

This chapter describes how to use each of the TCP inbound adapters (EnsLib.TCP.CountedInboundAdapter, EnsLib.TCP.CountedXMLInboundAdapter, and EnsLib.TCP.TextLineInboundAdapter). It contains the following sections:

Tip: Ensemble also provides specialized business service classes that use TCP adapters, and one of those might be suitable for your needs. If so, no programming would be needed. See "Connectivity Options" in *Introducing Ensemble*.

Also, you can develop a new inbound adapter class based on the EnsLib.TCP.InboundAdapter or any of its subclasses. See the section "Creating Custom TCP Adapter Classes," later in this book.

1.1 Overview of Inbound TCP Adapter

Ensemble provides the following inbound TCP adapters, all of which are subclasses of EnsLib.TCP.InboundAdapter:

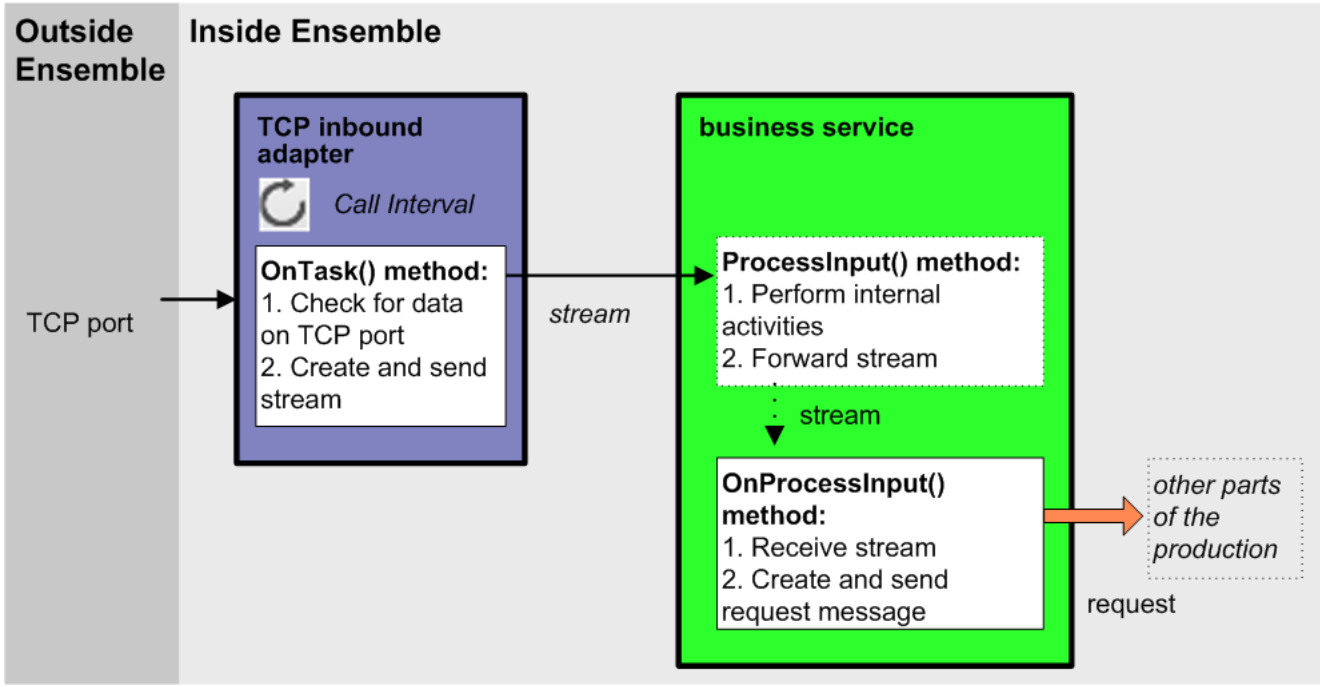
- EnsLib.TCP.CountedInboundAdapter supports incoming TCP connections over which a TCP client and TCP listener exchange blocks of data, with the block length specified in the first 4 bytes of the block. The adapter uses the block length to acquire the meaningful portion of the data from the client application.
- EnsLib.TCP.CountedXMLInboundAdapter acts as a TCP listener for an XTE server. The adapter receives XML data in counted TCP format and imports the XML data into Ensemble. The result is an instantiated Ensemble object.
- EnsLib.TCP.TextLineInboundAdapter supports incoming TCP connections over which a TCP client and TCP listener exchange data as text strings that end with a line terminator character. The default terminator is the newline character (ASCII 10).

  An important example of a business service that uses the EnsLib.TCP.TextLineInboundAdapter is the built-in business service class EnsLib.TCP.StatusService. This class provides a command-line interface that allows a console user or

command-line script to retrieve basic status information from a running Ensemble production. For more information, see "Adding a TCP Status Service," later in this book.

1.2 Overall Behavior

Each TCP inbound adapter checks for data on a specified port, reads the input, and sends the input as a stream to the associated business service. The business service, which you create and configure, uses this stream and communicates with the rest of the production. The following figure shows the overall flow:



In more detail:

1. Each time the adapter encounters input from its configured data source, it calls the internal **ProcessInput()** method of the business service class, passing the stream as an input argument.
2. The internal ProcessInput() method of the business service class executes. This method performs basic Ensemble tasks such as maintaining internal information as needed by all business services. You do not customize or override this method, which your business service class inherits.
3. The ProcessInput() method then calls your custom OnProcessInput() method, passing the input object. The requirements for this method are described later in this chapter.

The response message follows the same path, in reverse.

1.3 Creating a Business Service to Use a TCP Inbound Adapter

To use any of the TCP adapters in your production, create a new business service class as described here. Later, add it to your production and configure it.

You must also create appropriate message classes, if none yet exist. See "Defining Ensemble Messages" in *Developing Ensemble Productions*.

The following list describes the basic requirements of the business service class:

- Your business service class should extend Ens.BusinessService.
- In your class, the *ADAPTER* parameter should equal EnsLib.TCP.CountedInboundAdapter, EnsLib.TCP.CountedXMLInboundAdapter, or EnsLib.TCP.TextLineInboundAdapter.
- Your class must implement the OnProcessInput() method, as described in "Implementing the OnProcessInput Method."
- Your class can implement the **OnConnect()** callback method. This method is called each time the TCP inbound adapter establishes a new connection to or from a remote system.

  If you implement this method, it must have the following signature:

  ```
  Method OnConnect(pTimeout As %Numeric) As %Status
  ```

  Implement **OnConnect()** if you need to take some action each time a new connection is established, for example to send a logon sequence or a handshake token. The timeout argument is automatically provided by the TCP inbound adapter. It takes its value from the Read Timeout adapter setting. For details about Read Timeout, see "Reference for Settings."

  If **OnConnect()** returns an error status, the new connection fails and the adapter is disconnected. If an untrapped exception occurs within **OnConnect()**, the adapter catches it and continues as if **OnConnect()** were not implemented.
- For other options and general information, see "Defining a Business Service Class" in *Developing Ensemble Productions*.

Note: Studio provides a wizard that you can use to create a business service stub. To access this wizard, click **File —> New** and then click the **Production** tab. Then click **Business Service** and click **OK**. Note that the wizard provides a generic input argument. If you use the wizard, InterSystems recommends that you edit the method signature to use the specific arguments needed with the adapter you chose; see the next section.

1.4 Implementing the OnProcessInput() Method

This section describes the method signature for **OnProcessInput()**, which depends upon the adapter, and describes how to implement this method.

1.4.1 Signature for OnProcessInput() for EnsLib.TCP.CountedInboundAdapter

If your business service class uses EnsLib.TCP.CountedInboundAdapter, your **OnProcessInput()** method should have the following signature:

```
Method OnProcessInput(pInput As %Library.GlobalCharacterStream,
                      Output pOutput As %Library.AbstractStream) As %Status
```

Where:

- *pInput* contains the incoming data stream that the TCP client has directed to the adapter.
- *pOutput* contains any response that the business service might provide to the TCP client.

1.4.2 Signature for OnProcessInput() for EnsLib.TCP.CountedXMLInboundAdapter

If your business service class uses EnsLib.TCP.CountedXMLInboundAdapter, your **OnProcessInput()** method should have the following signature:

```
Method OnProcessInput(pInput As %RegisteredObject,
                      Output pOutput As %RegisteredObject) As %Status
```

Where:

- *pInput* can be any of the objects specified by the Accept Class Names adapter setting.

  For details about Accept Class Names, see "Reference for Settings."
- *pOutput* contains any response that you might need to return to the XTE server.

1.4.3 Signature for OnProcessInput() for EnsLib.TCP.TextLineInboundAdapter

If your business service class uses EnsLib.TCP.TextLineInboundAdapter, your **OnProcessInput()** method should have the following signature:

```
Method OnProcessInput(pInput As Ens.StringContainer,
                      Output pOutput As Ens.StringContainer) As %Status
```

Where:

- *pInput* contains the incoming line of text.
- *pOutput* contains the outgoing response string (if any).

1.4.4 Implementing OnProcessInput()

In all cases, the **OnProcessInput()** method should do some or all of the following:

1. Examine the input object (*pInput*) and decide how to use it.
2. Create an instance of the request message, which will be the message that your business service sends.

   For information on creating message classes, see "Defining Ensemble Messages" in *Developing Ensemble Productions*.
3. For the request message, set its properties as appropriate, using values in the input.
4. Call a suitable method of the business service to send the request to some destination within the production. Specifically, call **SendRequestSync()**, **SendRequestAsync()**, or (less common) **SendDeferredResponse()**. For details, see "Sending Request Messages" in *Developing Ensemble Productions*

   Each of these methods returns a status (specifically, an instance of %Status).
5. Make sure that you set the output argument (`pOutput`). Typically you set this equal to the response message that you have received. This step is required.

6. If the data source expects an acknowledgment or response to its input, **OnProcessInput()** must create this response and relay it to the data source, via the adapter.
7. Return an appropriate status. This step is required.

1.5 Example for EnsLib.TCP.CountedInboundAdapter

The following is an example of a business service class that uses EnsLib.TCP.CountedInboundAdapter.

```
Class Test.HL7v3.InterfaceEngine.Service.HL7TCPIn Extends Ens.BusinessService
{

 Parameter ADAPTER = "EnsLib.TCP.CountedInboundAdapter";

 Parameter SINGLEPOOLJOB = 1;

 Parameter SETTINGS = "TargetConfigNames";

 /// Configuration item to which to send messages
 Property TargetConfigNames As %String;

 Method OnProcessInput(pInput As %GlobalCharacterStream,
                       pOutput As %RegisteredObject) As %Status
 {
  Set $ZTrap = "OnProcessInputET"

  Set tRequest =
      ##class(Test.HL7v3.InterfaceEngine.Request.Message).%New()
  Set tRequest.Stream = pInput
  Set tRequest.DocType = $Case(pInput.Size < 10000, 1:"MCOA", :"CDAR2")

  If (..TargetConfigNames '= "") {
    For tCount = 1:1:$Length(..TargetConfigNames, ",") {
      Set tTarget =
          $ZStrip($Piece(..TargetConfigNames, ",", tCount), "<>W")
      Set tStatus = ..SendRequestAsync(tTarget, tRequest)
    }
  }

  Quit $$$OK

 OnProcessInputET
   Set $ZTrap = ""
   Quit $$$ERROR($$$GeneralError,"Error in OnProcessInput: "_$ZError)
 }

 /// Return an array of connections for
 /// drawing lines on the config diagram
 ClassMethod OnGetConnections(Output pArray As %String,
                              item As Ens.Config.Item)
 {
  Set (tValue,tIndex)=""
  For {
    Set tIndex = item.Settings.Next(tIndex) Quit:tIndex=""
    Set tSetting = item.Settings.GetAt(tIndex)
    If tSetting.Name="TargetConfigNames" Set tValue=tSetting.Value
  }
  For i=1:1:$L(tValue,",") {
   Set tOne=$P(tValue,",",i)
   Continue:""=tOne
   Set pArray(tOne)=""
  }
  Quit
 }
}
```

1.6 Example for EnsLib.TCP.TextLineInboundAdapter

The following is an example of a business service class that uses EnsLib.TCP.TextLineInboundAdapter.

```
Class TestTCPTextLine.AuthorizationTCPService Extends Ens.BusinessService
{
/// Name of the adapter class
Parameter ADAPTER = "EnsLib.TCP.TextLineInboundAdapter";

Method OnProcessInput(pInput As Ens.StringContainer,
                      pOutput As Ens.StringContainer) As %Status
{
  set $ZT = "EXCEPTION"
  set st = $$$OK

  do {
    if ('$isobject($get(pInput))) { quit }

    // Input must have the following format: 'PatientCode:ProcedureCode'
    set tSubject = pInput.StringValue
    $$$TRACE("received line "_tSubject)

    set req = ##class(TestTCPTextLine.AuthorizationRequest).%New()
    set req.PatientCode = $piece(tSubject,":",1)
    set req.ProcedureCode = $piece(tSubject,":",2)

    set st = ..SendRequestSync("AuthorizationProcess", req, .resp)
    quit:$$$ISERR(st)

    set pOutput=
      ##class(Ens.StringContainer).%New(resp.AuthorizationFlag_
        ":"_resp.AuthorizationCode)
    } while (0)

EXIT
   //do ..Adapter.Disconnect()
   quit st

EXCEPTION
   set $ZT = ""
   set st = $$$EnsSystemError
   goto EXIT
}
}
```

1.7 Adding and Configuring the Business Service

To add your business service to an Ensemble production, use the Management Portal to do the following:

1. Add an instance of your business service class to the Ensemble production.
2. Configure the business service. For information on the settings, see “Reference for Settings.”

   When you configure the business service, you specify a value for the **Allowed IP Addresses** settings, along other settings. Note that **Allowed IP Addresses** provides a way to enable the adapter to initiate the connection.
3. Enable the business service.
4. Run the production.

2

Using the Outbound TCP Adapters

This chapter describes how to use each of the TCP outbound adapters (EnsLib.TCP.CountedOutboundAdapter, EnsLib.TCP.CountedXMLOutboundAdapter, and EnsLib.TCP.TextLineOutboundAdapter). It contains the following sections:

Tip: Ensemble also provides specialized business service classes that use TCP adapters, and one of those might be suitable for your needs. If so, no programming would be needed. See "Connectivity Options" in *Introducing Ensemble*.

Also, you can develop a new outbound adapter class based on the EnsLib.TCP.OutboundAdapter or any of its subclasses. See the section "Creating Custom TCP Adapter Classes," later in this book.

2.1 Overview of Outbound TCP Adapter

Ensemble provides the following outbound TCP adapters, all of which are subclasses of EnsLib.TCP.OutboundAdapter:

- EnsLib.TCP.CountedOutboundAdapter contains methods to read or write blocks of data across the TCP connection. Ensemble acts as a TCP client exchanging blocks of data with a TCP listener. The block length specified in the first 4 bytes of the block. This convention allows an Ensemble business operation to send requests to an external TCP server for fulfillment.
- EnsLib.TCP.CountedXMLOutboundAdapter sends XML exported objects out as a counted block of bytes over a TCP connection and imports a response object.
- EnsLib.TCP.TextLineOutboundAdapter contains methods to read or write text strings across a TCP connection. Ensemble acts as a TCP client exchanging blocks of data with a TCP listener. The default terminator for the text strings is the newline character (ASCII 10).

2.2 Overall Behavior

Within a production, an outbound adapter is associated with a business operation that you create and configure. The business operation receives a message from within the production, looks up the message type, and executes the appropriate method. This method usually executes methods of the associated adapter.

2.3 Creating a Business Operation to Use a TCP Outbound Adapter

To create a business operation to use a TCP outbound adapter, you create a new business operation class. Later, add it to your production and configure it.

You must also create appropriate message classes, if none yet exist. See "Defining Ensemble Messages" in *Developing Ensemble Productions*.

The following list describes the basic requirements of the business operation class:

- Your business operation class should extend Ens.BusinessOperation.
- In your class, the *ADAPTER* parameter should equal EnsLib.TCP.CountedOutboundAdapter, EnsLib.TCP.CountedXMLOutboundAdapter, or EnsLib.TCP.TextLineOutboundAdapter.
- In your class, the *INVOCATION* parameter should specify the invocation style you want to use, which must be one of the following.
  - **Queue** means the message is created within one background job and placed on a queue, at which time the original job is released. Later, when the message is processed, a different background job is allocated for the task. This is the most common setting.
  - **InProc** means the message will be formulated, sent, and delivered in the same job in which it was created. The job will not be released to the sender's pool until the message is delivered to the target. This is only suitable for special cases.
- Your class must define a *message map* that includes at least one entry. A message map is an XData block entry that has the following structure:

```
XData MessageMap
{
<MapItems>
  <MapItem MessageType="messageclass">
    <Method>methodname</Method>
  </MapItem>
  ...
</MapItems>
}
```

- Your class should define all the methods named in the message map. These methods are known as *message handlers*. Each message handler should have the following signature:

  ```
  Method Sample(pReq As RequestClass, Output pResp As ResponseClass) As %Status
  ```

  Here *Sample* is the name of the method, *RequestClass* is the name of an Ensemble request message class, and *ResponseClass* is the name of an Ensemble response message class. In general, the method code will refer to properties and methods of the Adapter property of your business operation.

  For information on defining message classes, see "Defining Ensemble Messages" in *Developing Ensemble Productions*.

For information on defining the message handler methods, see "Creating Message Handler Methods," later in this chapter.

- Your class can implement the **OnConnect()** callback method. This method is called each time the TCP outbound adapter establishes a new connection to or from a remote system.

  If you implement **OnConnect()** method, it must have the following signature:

  ```
  Method OnConnect(pTimeout As %Numeric) As %Status
  ```

  Implement this method if you need to take some action each time a new connection is established, for example to send a logon sequence or a handshake token. The timeout argument is automatically provided by the TCP outbound adapter. It takes its value from the ConnectTimeout adapter setting. For details, see "Settings for the TCP Outbound Adapters."

  If **OnConnect()** returns an error status, the new connection fails and the adapter is disconnected. If an untrapped exception occurs within **OnConnect()**, the adapter catches it and continues as if **OnConnect()** were not implemented.

- For other options and general information, see "Defining a Business Operation Class" in *Developing Ensemble Productions*.

The following example shows the general structure that you need:

```
Class ETCP.NewOperation1 Extends Ens.BusinessOperation
{
Parameter ADAPTER = "EnsLib.TCP.CountedOutboundAdapter";

Parameter INVOCATION = "Queue";

Method Sample(pReq As RequestClass, Output pResp As ResponseClass) As %Status
{
  Quit $$$ERROR($$$NotImplemented)
}

XData MessageMap
{
<MapItems>
  <MapItem MessageType="RequestClass">
    <Method>Sample</Method>
  </MapItem>
</MapItems>
}
}
```

Note: Studio provides a wizard that you can use to create a business operation stub similar to the preceding. To access this wizard, click **File —> New** and then click the **Production** tab. Then click **Business Operation** and click **OK**.

2.4 Creating Message Handler Methods

When you create a business operation class for use with a TCP outbound adapter, typically your biggest task is writing message handlers for use with this adapter, that is, methods that receive Ensemble messages and then write files.

Each message handler method should have the following signature:

```
Method Sample(pReq As RequestClass, Output pResp As ResponseClass) As %Status
```

Here *Sample* is the name of the method, *RequestClass* is the name of an Ensemble request message class, and *ResponseClass* is the name of an Ensemble response message class.

In general, the method should do the following:

1. Examine the inbound request message.

2. Using the information from the inbound request, call a method of the Adapter property of your business operation. For example, the following line invokes the **Connect()** method:

   ```
   set sc=..Adapter.Connect()
   ```

   See "Calling Adapter Methods from the Business Operation."

3. Make sure that you set the output argument (`pOutput`). Typically you set this equal to the response message. This step is required.

4. Return an appropriate status. This step is required.

2.4.1 Calling Adapter Methods from the Business Operation

Connect()

```
Method Connect(pTimeout As %Numeric) As %Status
```

Applies to all TCP outbound adapters.

Connect() opens a TCP socket to a port on the TCP listener machine. **Connect()** has one input argument, a %Numeric value that identifies the number of seconds to wait for the connection attempt to succeed. A typical call to the **Connect()** method gives this input argument the value of the configurable ConnectTimeout property.

If the Stay Connected adapter setting is set to –1, Ensemble creates the connection automatically at startup time, from within the **OnInit()** method. See "Reference for Settings."

Disconnect()

```
Method Disconnect()
```

Applies to all TCP outbound adapters.

Disconnect() takes down the connection to the TCP listener.

SendMessageStream()

```
Method SendMessageStream(pRequestStream As %Stream.Object,
  ByRef pResponseStream As %CharacterStream = "%GlobalCharacterStream")
  As %Status
```

Applies to EnsLib.TCP.CountedOutboundAdapter

SendMessageStream() sends the stream contents as a counted block on the TCP socket.

SendMessageString()

```
Method SendMessageString(pRequestString As %String,
                Output pResponseString As %String) As %Status
```

Applies to EnsLib.TCP.CountedOutboundAdapter and EnsLib.TCP.TextLineOutboundAdapter

SendMessageString() sends the string contents as a counted block on the TCP socket.

SendMessageXMLObj()

```
Method SendMessageXMLObj(pRequest As %RegisteredObject,
                Output pResponse As %RegisteredObject) As %Status
```

Applies to EnsLib.TCP.CountedXMLOutboundAdapter

SendMessageXMLObj() sends the request object to the TCP listener as a counted XML block. If the Get Reply adapter setting is True, **SendMessageXMLObj()** also gets a reply from the TCP listener as a counted XML block, which it imports as an Ensemble object. For details on Get Reply, see "Reference for Settings."

TestConnection()

```
Method TestConnection()
```

Applies to all TCP outbound adapters.

TestConnection() corrects the properties reflecting the connection state. If the adapter is connected, **TestConnection()** returns True. If not, **TestConnection()** calls **Disconnect()** and returns False.

2.5 Example for EnsLib.TCP.CountedOutboundAdapter

The following is an example of a business operation class that references the EnsLib.TCP.CountedOutboundAdapter.

```
Class Test.GeneralTCPOperation Extends Ens.BusinessOperation
{
 Parameter ADAPTER = "EnsLib.TCP.CountedOutboundAdapter";
 Parameter MAXREAD=30000;

 Method OnMessage(pReq As Test.GeneralSendRequest,
                  Output pResponse As Test.GeneralSendResponse) As %Status
 {
  Set str=
    pReq.Hr_pReq.DataLth_pReq.BID_pReq.Hr2_pReq.HrVacant_pReq.Cat_pReq.Hr3_pReq.Hr4
  Set tTextStream= ##class(%GlobalCharacterStream).%New()
  Set tSC=tTextStream.Write(str)
  Do pReq.Body2.Read(32) // Throw it away
  Set len=..#MAXREAD
  Set token= pReq.Body2.Read(.len)
  Set i=0
  While(len>0) {
     Set i=i+1
     Do tTextStream.Write(token)
     Set len=..#MAXREAD
     Set token= pReq.Body2.Read(.len)
  }
  Set tSC=
  ..Adapter.SendMessageStream(tTextStream,.tStreamReply) Quit:$$$ISERR(tSC) tSC
  Set pResponse=##class(Test.GeneralSendResponse).%New()
  Do tStreamReply.Rewind()
  Set pResponse.Body = tStreamReply.Read(,.tSC)
  Quit tSC
 }
}
```

2.6 Adding and Configuring the Business Operation

To add your business operation to an Ensemble production, use the Management Portal to do the following:

1. Add an instance of your business operation class to the Ensemble production.
2. Configure the business operation. For information on the settings, see "Reference for Settings."
3. Enable the business operation.
4. Run the production.

3

Special Topics

This chapter discusses the following additional topics:

- Adding a TCP Status Service
- Creating Custom TCP Adapter Classes
- Common Customizations in TCP Adapter Subclasses

For general information on customizing the callback methods of adapters, see "Less Common Tasks" in *Developing Ensemble Productions*.

Tip: Ensemble also provides specialized business service classes that use TCP adapters, and one of those might be suitable for your needs. If so, no programming would be needed. See "Connectivity Options" in *Introducing Ensemble*.

3.1 Adding a TCP Status Service

The EnsLib.TCP.StatusService class parses an incoming text string to determine what type of production status information is being requested, and then produces a reply string suitable for writing out to the console screen. A user can interact with the status service directly, or write a command script that contacts the status service, issues commands, and writes the replies to a text file for later analysis.

A developer enables interactions via the status service as follows:

1. Add an instance of EnsLib.TCP.StatusService to the production.
2. Configure the instance of EnsLib.TCP.StatusService to accept communications via a particular TCP port. Set a Port number and a list of Allowed IP Addresses from which to accept connections. These are only two of the several settings that are available to configure the TCP text line inbound adapter associated with the status service. For details, see "Reference for Settings."

Once you have completed these steps, any time the status service is enabled and running, a user or command-line script can initiate a Telnet connection to the configured Port and send commands to the status service. Each command must be a text string that ends with the newline character (ASCII 10). The reply strings will also terminate with a newline.

The following table describes the supported command lines and the contents of the text string returned in each case.

Command Line	Text String Returned
`build`	The full name of the Ensemble software version, for example `Cache for Windows (x86-32) 2007.1.2 (Build 514U) Mon Aug 13 2007 11:30:37 EDT`
`configitemstatus` *name*	When you enter this command with the configured *name* of any business host in the currently running production, the status service returns a string that expresses the current state of that business host. A host that has currently running jobs, or active connections, records those activities in the returned string. If the identified host is not a member of the currently running production, the returned string indicates this.
`exit`	No string is returned. This command disconnects from the status service. You may enter `x` instead of `exit`
`localstarttime`	If Ensemble is running, this returns the start time of the currently running production, expressed in local time coordinates. Otherwise, it returns a string with the message `<UNDEFINED>`
`localtime`	The current time, in local time coordinates.
`namespace`	The Ensemble namespace where the currently running production resides.
`production`	The configured name of the currently running production.
`quit`	No string is returned. This command disconnects from the status service. You may enter `q` instead of `quit`
`utcstarttime`	If Ensemble is running, this returns the start time of the currently running production, is expressed in universal time coordinates (UTC). Otherwise, it returns a string with the message `<UNDEFINED>`
`utctime`	The current time, in universal time coordinates (UTC).
`version`	The abbreviated name of the Ensemble software version, for example `2007.1.0.514/2171`
`x`	Same as `exit`

3.2 Creating Custom TCP Adapter Classes

To create a custom subclass of a TCP adapter class, do the following in Studio:

1. On the **File** menu, click **New**, and then click the **General** tab.
2. Start the New Class Wizard by clicking **Caché Class Definition** and clicking **OK**.
   a. Enter a package and class name and click **Next**.

      Important: Be sure to avoid reserved package names; see "Reserved Package Names" in *Developing Ensemble Productions*.

   b. Click **Extends** for the **Class type**.
   c. Click **Browse** next to **Name of super class** and navigate to the class that you want to subclass.

d. Click **OK**.

e. Click **Finish**.

The result is a template class like the following, depending on the class you chose:

```
Class MyTest.NewClass1 Extends EnsLib.TCP.InboundAdapter
{

}
```

3. You can override any property, method, class parameter, or other class member inherited from the class or its parent classes, or you can add new class members. The only requirements are as follows:

   - An *inbound* adapter class must implement the **OnConnected()** method.

     In your implementation, use helper methods inherited by your class.

   - An *outbound* adapter class define methods that create a TCP connection, read or write over the connection, and disconnect.

     To define these methods, use helper methods inherited by your class.

4. Compile the class, which also saves it.

3.3 Common Customizations in TCP Adapter Subclasses

The following list suggests some common customizations in TCP adapter subclasses, in addition to implementing the methods that the adapter needs:

- You can define a different value for InitialExpressions for the Terminators property.

  To specify an ASCII character, use the ObjectScript function $CHAR (abbreviated to $C). The following example from EnsLib.TCP.TextLineInboundAdapter sets Terminators to the newline character (ASCII 10):

  ```
  Property Terminators As %String [ InitialExpression = {$C(10)} ];
  ```

  For more information about functions like $CHAR, see "ObjectScript Functions" in the *Caché ObjectScript Reference*.

  You can set Terminators to a single character or to a multi-character string. If you supply a string for the Terminators value, Ensemble uses the string as follows:

  – Any one of the characters in the string serves to terminate the input.

  – The complete string of characters is appended to the output.

- You can add properties and methods.
- You can add and remove settings. See "Adding and Removing Settings" in the chapter "Programming in Ensemble" in *Developing Ensemble Productions*.
- If you need to require login credentials for the connection, simply add the property name `Credentials` to the `SETTINGS` list. The `Credentials` property is already defined as a %String in the base class Ens.Adapter.
- (For inbound adapters) You can also enforce a pool size of 1 by setting the parameter *SINGLEPOOLJOB* = 1:

  ```
  /// Force a single listener job regardless of PoolSize setting
  Parameter SINGLEPOOLJOB = 1;
  ```

Any subclass of this adapter class or a business service class that uses such an adapter class can use this parameter in its class code to better control the pool size.

- You can implement the **OnInit()** callback method to perform any special setup actions.

  For inbound adapters, one of these actions might be to examine settings and initiate a connection between the adapter and the TCP client it will listen to, or to create any object properties that the adapter will use.

  For outbound adapters, one of these actions might be to examine settings and open a socket to a port on the TCP listener.

- For inbound adapters, you can implement the **OnConnected()** callback method.

  Whenever a connection exists to the configured TCP client, the adapter calls **OnConnected()** to read a stream from the TCP data source. This read operation is controlled by adapter settings.

  OnConnected() uses helper methods to parse the data.

  Upon successfully reading data, **OnConnected()** calls the Ensemble framework's **ProcessInput()** method to pass the received data stream to the associated business service. If this call returns an outbound stream, **OnConnected()** writes that data as a reply to the TCP client.

  If no data is available to read from the TCP connection during the CallInterval time, **OnConnected()** returns without doing anything.

Reference for Settings

This section provides the following reference information:

- Settings for the TCP Inbound Adapters
- Settings for the TCP Outbound Adapters

Also see "Settings in All Productions" in *Managing Ensemble Productions*.

Settings for the TCP Inbound Adapters

Provides reference information for settings of the TCP inbound adapters.

Summary

The inbound TCP adapters have the following settings:

Group	Settings
Basic Settings	Call Interval, Port
Connection Settings	Job Per Connection, Allowed IP Addresses, OS Accept Connection Queue Size, Stay Connected, Read Timeout, SSL Configuration, Local Interface
Additional Settings	Accept Classnames, Charset

The remaining settings are common to all business services. For information, see "Settings for All Business Services" in *Configuring Ensemble Productions*.

Accept Classnames

Applies to EnsLib.TCP.CountedXMLInboundAdapter.

Comma-separated list, giving the names of classes that this adapter will instantiate from XML blocks received.

Allowed IP Addresses

Applies to all inbound TCP adapters.

A comma-separated list of remote IP addresses from which to accept connections. The adapter accepts named IP addresses, IPV4 addresses, or IPV6 addresses. An optional :*port* designation is supported; so, for example, both of the following address formats are acceptable: 192.168.1.22 or 192.168.1.22:3298.

Note: IP address filtering is a means to control access on private networks, rather than for publicly accessible systems. InterSystems does not recommend relying on IP address filtering as a sole security mechanism, as it is possible for attackers to spoof IP addresses.

If a port number is specified, connections from other ports will be refused.

If the string starts with an exclamation point (!) character, the inbound adapter initiates the connection rather than waiting for an incoming connection request. The inbound adapter initiates the connection to the specified address and then waits for a message. In this case, only one address may be given, and if a port is specified, it supersedes the value of the Port setting; otherwise, the Port setting is used.

Call Interval

Applies to all inbound TCP adapters.

Number of seconds that the adapter will listen for incoming data from its configured source, before checking for a shutdown signal from the Ensemble framework.

If the adapter finds input, it acquires the data and passes it to the business service. The business service processes the data, and then the adapter immediately begins waiting for new input. This cycle continues whenever the production is running and the business service is enabled and scheduled to be active.

The default CallInterval is 5 seconds. The minimum is 0.1 seconds.

Charset

Applies to EnsLib.TCP.CountedInboundAdapter, EnsLib.TCP.CountedXMLInboundAdapter, and EnsLib.TCP.TextLineInboundAdapter.

Specifies the character set of the inbound data. Ensemble automatically translates the characters from this character encoding. The setting value is not case-sensitive. Use `Binary` for binary files, or for any data in which newline and line feed characters are distinct or must remain unchanged, for example in HL7 Version 2 or EDI messages. Other settings may be useful when transferring text documents. Choices include:

- `Binary` — Binary transfer (the default)
- `Ascii` — Ascii mode FTP transfer but no character encoding translation
- `Default` — The default character encoding of the local Ensemble server
- `Latin1` — The ISO Latin1 8-bit encoding
- `ISO-8859-1` — The ISO Latin1 8-bit encoding
- `UTF-8` — The Unicode 8-bit encoding
- `UCS2` — The Unicode 16-bit encoding
- `UCS2-BE` — The Unicode 16-bit encoding (Big-Endian)
- Any other alias from an international character encoding standard for which NLS (National Language Support) is installed in Ensemble

For background information on character translation in Caché, see "Localization Support" in the *Caché Programming Orientation Guide*.

Endian

Applies to EnsLib.TCP.CountedInboundAdapter and EnsLib.TCP.CountedXMLInboundAdapter.

A choice of `Big` or `Little` indicates the byte order of the 4-byte block count prefix. `Big` endian means the most significant byte (MSB) goes over the wire first; `Little` endian means the least significant byte (LSB) goes over the wire first. The default value for this string is `Big`.

Job Per Connection

Applies to all inbound TCP adapters.

When True, the adapter spawns a new job to handle each incoming TCP connection and allows simultaneous handling of multiple connections. When False, it does not spawn a new job for each connection. False is usually the appropriate choice for HL7 or X12 connections. The default is True.

For TCP services, **JobPerConnection** causes each new incoming socket connection to be handled by a freshly spawned job rather than by the listener job itself. Only one job at a time can be the listener, and one job must be the listener, so TCP services configured with **PoolSize** greater than 1 still only start one listener job. However, this listener can spawn an unlimited number of connection jobs if **JobPerConnection** is set to True. If the **PoolSize** setting is configured to a value greater than 1, it serves as a limit on the number of simultaneous connection jobs that can exist. When this limit is reached, the listener does not accept any more connections until one or more of the existing connection jobs quits or dies. An Event Log warning appears when it first reaches the limit.

Local Interface

Applies to all inbound TCP adapters.

Specifies the network interface through which the TCP connection should go. Select a value from the list or type a value. An empty value means use any interface.

OS Accept Connection Queue Size

Applies to all inbound TCP adapters.

Specifies the number of incoming connections the operating system should hold in reserve on behalf of this business service. Set to 0 if only 1 connection at a time is expected. Set to a large number if you expect many clients to connect rapidly. The default is 100. The range is 0–1000.

Port

Applies to all inbound TCP adapters.

Identifies the TCP port on the local machine where the adapter is listening for TCP requests. Avoid specifying a port number that is in the range used by the operating system for ephemeral outbound connections. See "Inbound Ports May Conflict with Operating System Ephemeral Ports" in the *Ensemble Release Notes* for more information.

QSize

Applies to EnsLib.TCP.CountedInboundAdapter, EnsLib.TCP.CountedXMLInboundAdapter, and EnsLib.TCP.TextLineInboundAdapter.

Specifies the number of incoming connections the operating system should hold in reserve on behalf of this business service. Set to 0 if only 1 connection at a time is expected. Set to a large number if many clients will connecting rapidly. The default is 100. The range is 0–1000.

Read Timeout

Applies to all TCP adapters.

Number of seconds to wait for each successive incoming TCP read operation, following receipt of initial data from the remote TCP port. The default is 5 seconds. The range is 0–600 seconds (a maximum of 10 minutes).

SSLConfig

Applies to all TCP adapters.

The name of an existing SSL/TLS configuration to use to authenticate this connection. Choose a client SSL/TLS configuration, because the adapter initiates the communication.

To create and manage SSL/TLS configurations, use the Management Portal. See the chapter "Using SSL/TLS with Caché" in the *Caché Security Administration Guide*. The first field on the **Edit SSL/TLS Configuration** form is **Configuration Name**. Use this string as the value for the **SSLConfig** setting.

Stay Connected

Applies to all TCP adapters.

If StayConnected is a positive value, the adapter stays connected to the remote system for this number of seconds between input events. A zero value means to disconnect immediately after every input event. The default of –1 means to stay permanently connected, even during idle times. Adapters are assumed idle at startup and therefore only auto-connect if they are configured with a StayConnected value of –1.

The value of StayConnected controls how the TCP adapter treats disconnections. If StayConnected has a value of –1, the TCP adapter treats a disconnection as an error. If it has a value of 0 or a positive integer, the TCP adapter does not consider a disconnection an error.

Settings for the TCP Outbound Adapters

Provides reference information for settings of the TCP outbound adapters.

Summary

The outbound TCP adapters have the following settings:

Group	Settings
Basic Settings	IP Address, Port
Connection Settings	Stay Connected, Connect Timeout, Reconnect Retry, Get Reply, Response Timeout, Read Timeout, SSL Configuration, Local Interface, Charset, Endian
Additional Settings	Charset

The remaining settings are common to all business operations. For information, see "Settings for All Business Operations" in *Configuring Ensemble Productions*.

Charset

Applies to EnsLib.TCP.CountedOutboundAdapter, EnsLib.TCP.CountedXMLOutboundAdapter, and EnsLib.TCP.TextLineOutboundAdapter.

Specifies the desired character set of the outbound data. Ensemble automatically translates the characters to this character encoding. See "Charset" in "Settings for the TCP Inbound Adapters."

Connect Timeout

Applies to all outbound TCP adapters.

Number of seconds to wait on each attempt to connect to the remote TCP listener. The default is 5 seconds.

Endian

Applies to EnsLib.TCP.CountedOutboundAdapter and EnsLib.TCP.CountedXMLOutboundAdapter.

A choice of `Big` or `Little` indicates the byte order of the 4-byte block count prefix. `Big` endian means the most significant byte (MSB) goes over the wire first; `Little` endian means the least significant byte (LSB) goes over the wire first. The default value for this string is `Big`.

Get Reply

Applies to all outbound TCP adapters.

If 1 (true) wait to read a reply message back from the socket before returning. If 0 (false) do not wait. The default value is 1.

IP Address

Applies to all outbound TCP adapters.

IP address to make a TCP connection to. The adapter accepts a named IP address, an IPV4 address, or an IPV6 address.

If the string starts with an exclamation point (!) character, the outbound adapter does not initiate the connection but rather waits for an incoming connection request. Once it accepts the connection request, it can send outbound messages. The exclamation point can be specified by itself or followed by a comma-separated list of IP addresses. If the exclamation point is not followed by an IP address, the outbound TCP adapter accepts a connection from any IP address, otherwise the adapter

accepts connections only from the specified addresses. The adapter accepts named IP addresses, IPV4 addresses, or IPV6 addresses. You can optionally specify a port for each IP address by appending a colon (:) followed by the port number. If a port is specified, the adapter accepts connections from only the specified port.

Note: IP address filtering is a means to control access on private networks, rather than for publicly accessible systems. InterSystems does not recommend relying on IP address filtering as a sole security mechanism, as it is possible for attackers to spoof IP addresses.

LocalInterface

Applies to all outbound TCP adapters.

Specifies the network interface through which the TCP connection should go. Select a value from the list or type a value. An empty value means use any interface. For details, see "Local Interface" in "Settings for the TCP Inbound Adapters."

Port

Applies to all outbound TCP adapters.

Identifies the TCP port to connect to.

QSize

Applies to EnsLib.TCP.CountedOutboundAdapter, EnsLib.TCP.CountedXMLOutboundAdapter, and EnsLib.TCP.TextLineOutboundAdapter.

Specifies the number of incoming connections the operating system should hold in reserve on behalf of this business operation. There is generally no reason to change the default value of 0. Either the connection is outbound, or the business operation is using '!' inbound mode, in which case only 1 client is allowed at a time, per business operation pool job.

Read Timeout

Applies to all TCP adapters.

Number of seconds to wait for each successive incoming TCP read operation, following receipt of initial data from the remote TCP port. The default is 5 seconds. The range is 0–600 seconds (a maximum of 10 minutes).

Reconnect Retry

Applies to all outbound TCP adapters.

Number of retries at which to drop the connection and try reconnecting again. A value of 0 (zero) means never disconnect. The default value is 5.

Response Timeout

Applies to all outbound TCP adapters.

Number of seconds to wait for a response to begin arriving back from the remote system after sending a request. The default is 15 seconds.

SSL Configuration

Applies to all TCP adapters.

The name of an existing SSL/TLS configuration to use to authenticate this connection. Choose a client SSL/TLS configuration, because the adapter initiates the communication. For details, see "SSL Config" in "Settings for the TCP Inbound Adapters."

Stay Connected

Applies to all TCP adapters.

If StayConnected is a positive value, the adapter stays connected to the remote system for this number of seconds after completing an operation. A zero value means to disconnect immediately after every operation. The default of –1 means to stay permanently connected, even during idle times. Adapters are assumed idle at startup and therefore only auto-connect if they are configured with a StayConnected value of –1.

The value of StayConnected controls how the TCP adapter treats disconnections. If StayConnected has a value of –1, the TCP adapter treats a disconnection as an error. If it has a value of 0 or a positive integer, the TCP adapter does not consider a disconnection an error.