TIBCO iWay® Service Manager

nHTTP Representational State Transfer Getting Started Guide

Version 7.0.x and Higher
March 2021
DN3502301.0321

Contents

Chapter 1

Introducing nHTTP Representational State Transfer

This section provides an overview of nHTTP Representational State Transfer (REST), including key features.

In this chapter:

Representational State Transfer (REST)

Representational State Transfer (REST) is a type of software architecture that is designed for distributed hypermedia systems, such as the web. The REST architectural style can be summarized as four main verbs: GET, POST, PUT, and DELETE. These verbs pertain to the HTTP 1.1 specification and the nouns which are the resources available on the network via a URI. Please note that REST is not a standard itself and implementation for its support varies. However, REST does use standards such as HTTP, URL, Resource Representations (XML/HTML/GIF/JPEG/etc.) and Mime-Types (text/xml, text/html, image/gif/etc.). REST is based on a client-server architecture using the pull-based interaction style. It is stateless and as a result, each request from a client to a server must contain all the information necessary to understand the request. It cannot take advantage of any stored context on the server. The basis for REST is the use of nouns that represent resources identified with the URI and a set of verbs, which define actions for the nouns.

nHTTP REST Support

The nHTTP listener for iWay Service Manager complies with the HTTP 1.1 specification. The listener implements all the available verbs used in REST-style communication, including the main verbs (GET, POST, PUT, and DELETE). The handling of the verbs can be configured during the nHTTP listener configuration and allows you to indicate if the verb should generate an event document, which will propagate to the process flow or if it should generate an error. The key here is that this ability to create an event document from the HTTP request allows an iSM channel to execute an underlying process flow from a URL request. As the URL request is being served by the nHTTP listener, the event document is generated, the process flow is executed, and the response document is returned via HTTP. This enables any process flow to run from any external application by issuing a URL reference to it. The REST style service support of addressing a resource with a supported set of verbs is available via the nHTTP listener implementation. For more information on a full array of nHTTP supported features, see the *iWay Service Manager User's Guide*.

The following table lists the supported HTTP requests.

Request Type	Available Actions
GET	❑ docroot - serve document from docroot. ❑ event - generate an event document. ❑ error - return HTTP error.
POST	flow - non-configurable, request propagated to flow
HEAD	❑ docroot - serve document from docroot. ❑ event - generate an event document. ❑ error - return HTTP error.
PUT	❑ event - generate an event document. ❑ error - return HTTP error.
DELETE	❑ event - generate an event document. ❑ error - return HTTP error.

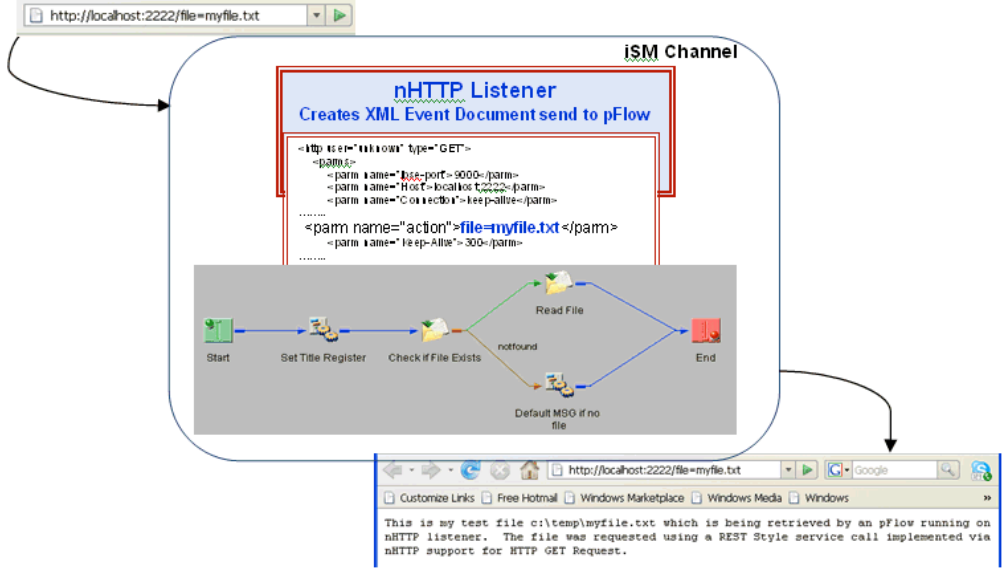
Request Type	Available Actions
TRACE	Echoes request as per RFC.
OPTION	Returns available options as per RFC.

nHTTP Configuration Overview

The example that is provided in this documentation demonstrates the configuration of an nHTTP listener and a simple process flow.

Note: You must have iWay Service Manager 6.0 or higher installed to configure and use an nHTTP listener.

The nHTTP listener is listening for HTTP requests on port 2222 and is configured to process HTTP GET requests as events. It will generate an event document with all the available information from the HTTP GET request and propagate it to the configured process flow. The process flow will parse the received event document representing the HTTP GET request and will attempt to retrieve the requested file from the directory. If the file is found, it will return the file content as an nHTTP response. If the requested file is not available, an error message will be returned. Please note that even though there is an option to simply serve the documents from the docroot, this example will call a process flow for this purpose. This is done to illustrate a technique to call a process flow as a REST style service call.



For more information on how to configure an nHTTP listener and a simple process flow, see *Configuring a Process Flow and nHTTP Channel* on page 9.

Chapter 2

Configuring a Process Flow and nHTTP Channel

This section describes how to configure a process flow and channel for nHTTP Representational State Transfer.

In this chapter:

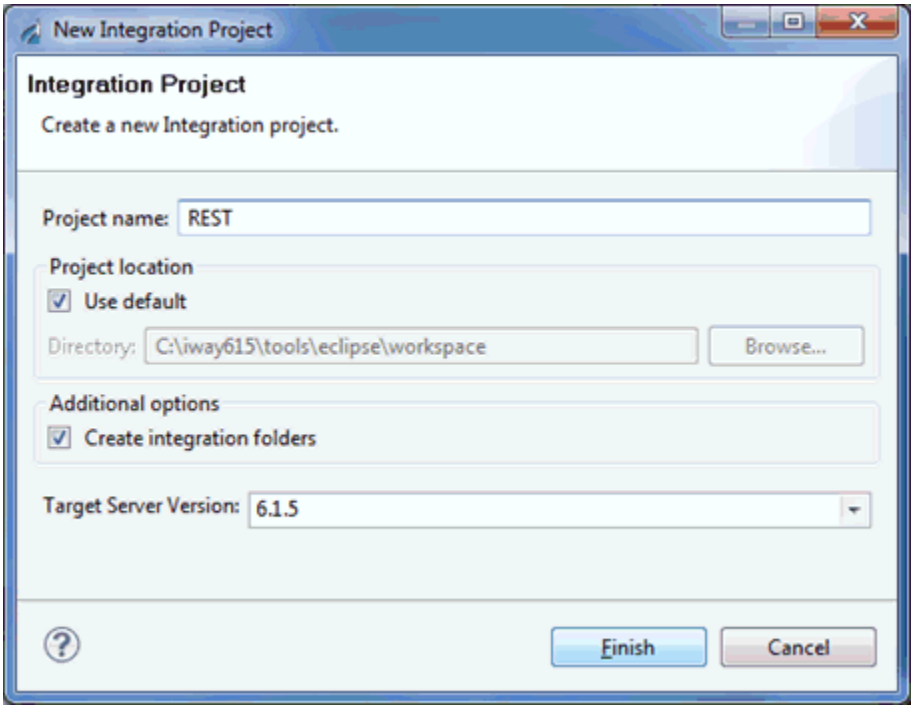
Configuring a Process Flow

A process flow contains all the business logic and performs the required processing. It parses the received event document to retrieve the parameter for the requested file name. It validates that the requested file exists. If the file exists, the process flow retrieves the content of the file. If the file is not available, the process flow returns a default error message.

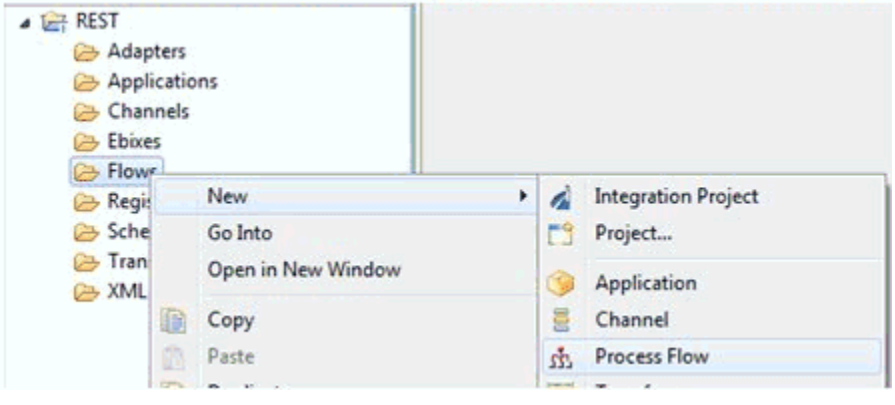
Procedure: How to Configure a Process Flow

To configure a process flow:

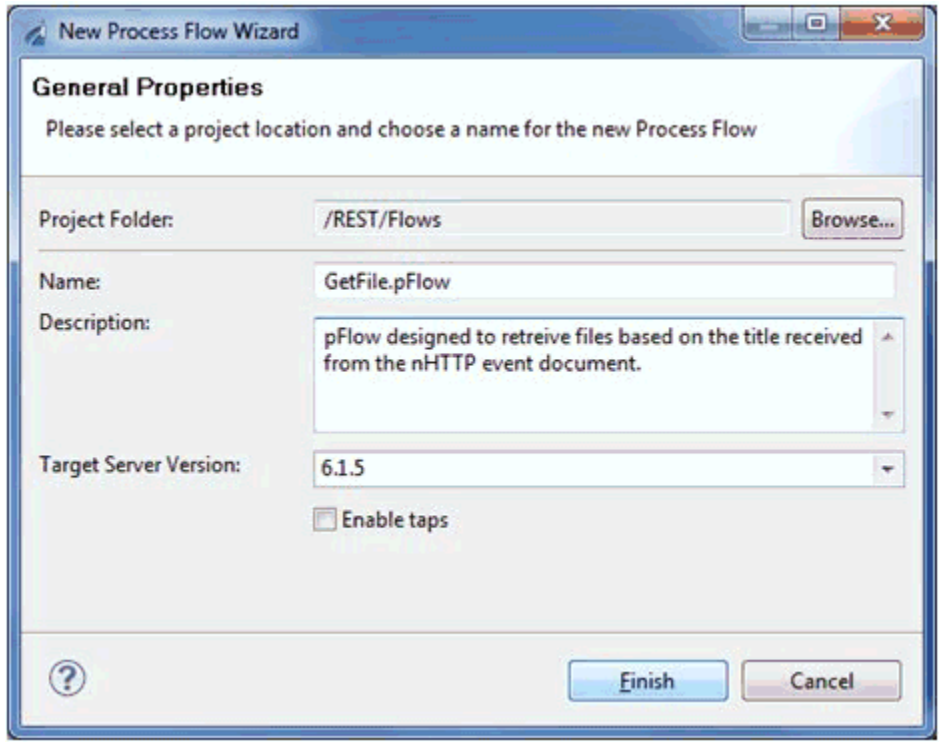
1. Open iWay Integration Tools (iIT) and create a new Integration Project called REST, as shown in the following image.



2. Expand the *REST* Integration Project node, right-click the *Flows* subfolder and select *Process Flow* from the context menu, as shown in the following image.

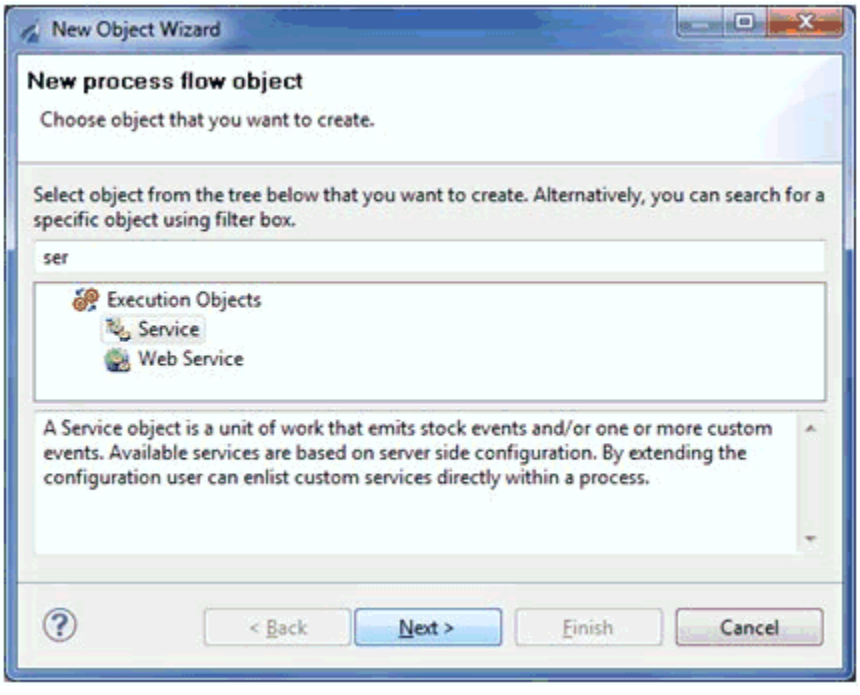


The New Process Flow Wizard opens, as shown in the following image.



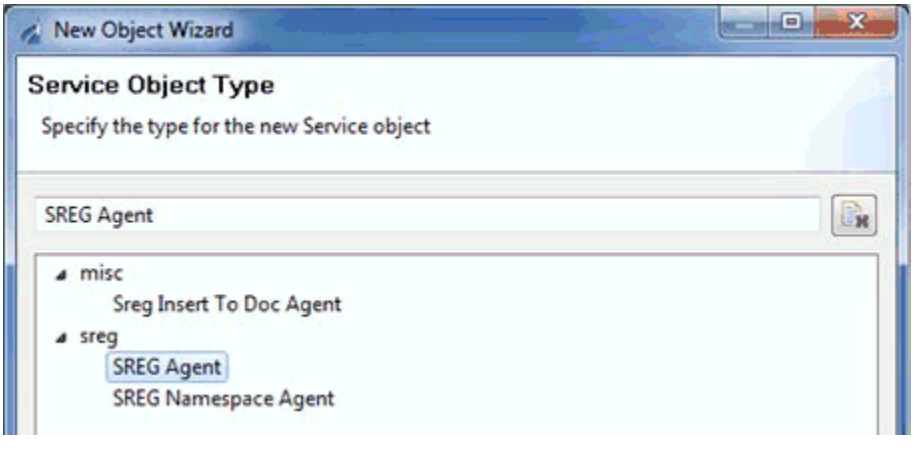
3. Create a new process flow called GetFile.pFlow under the new REST Integration Project.
4. Click *Finish*.

5. Add a Service object to the new process flow.



This Service object will set the required Special Registers (SREGs) that are required by the process flow.

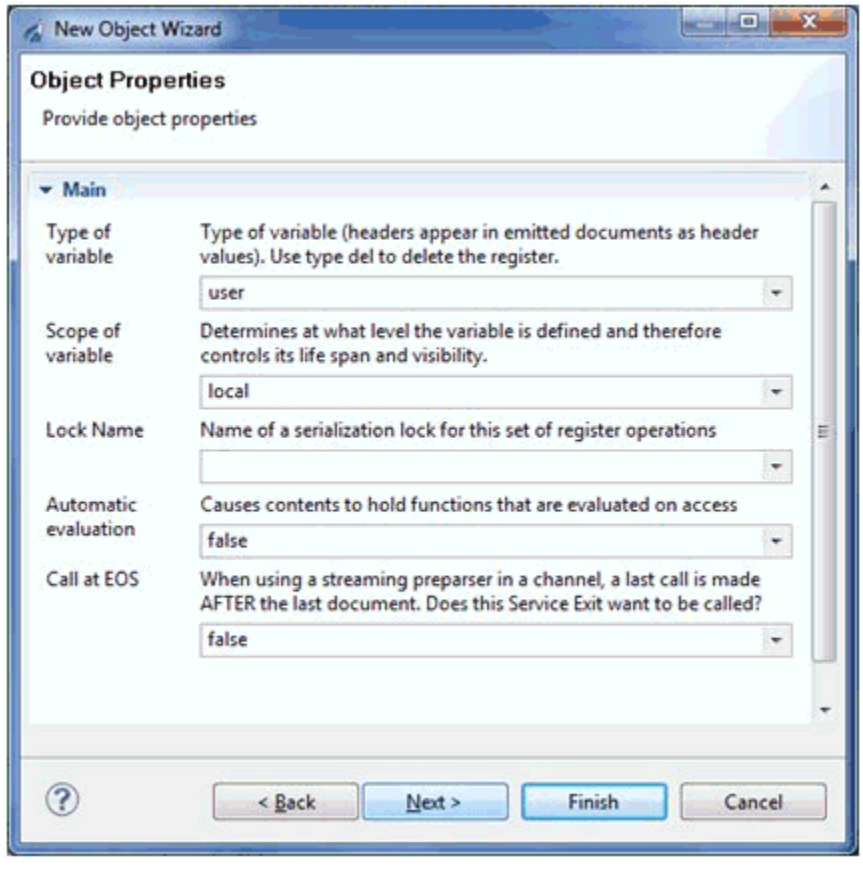
6. Name the new Service object as Set Title Register and select *SREG Agent* as the service type.



The SREG Agent service type allows SREGs to be used throughout the business logic for data manipulation and processing.

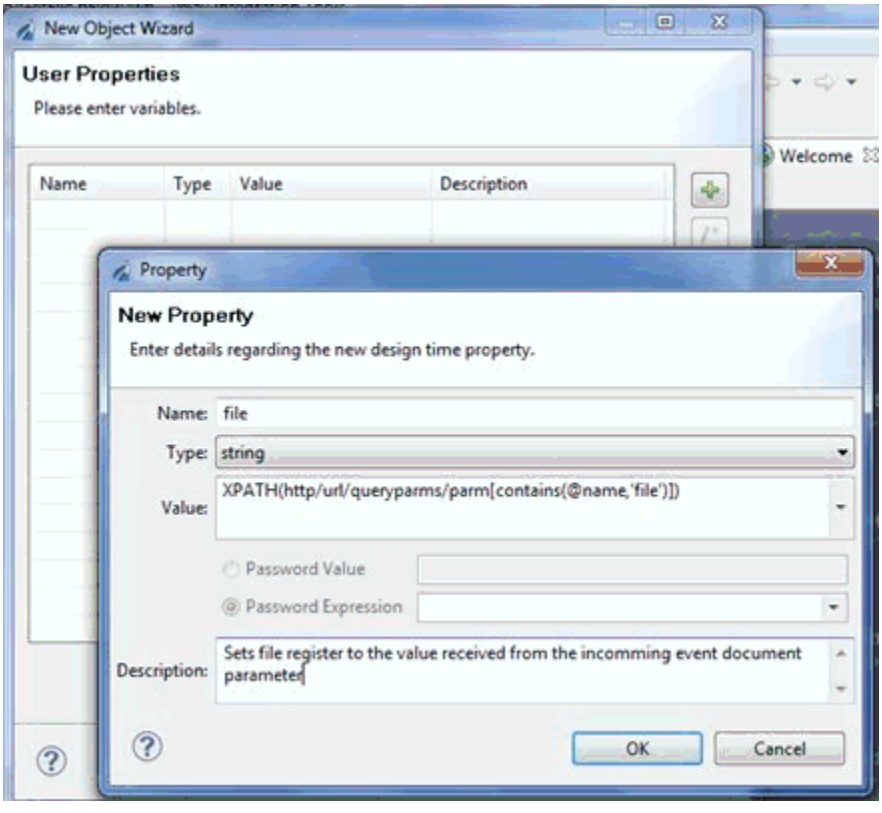
7. Click *Next*.

   The Object Properties pane opens, as shown in the following image.



8. Leave the default values that are provided and click *Next*.

The User Properties pane opens, as shown in the following image.



9. Click the green plus sign (+) icon to define a new design time property.
10. Enter the following settings for the new property:
    - **Name:** file
    - **Type:** string
    - **Value:**

      ```
      XPATH(http/url/queryparms/queryparm[contains(@name,'file')])
      ```

    - **Description:** Sets file registers to the value received from the incoming event document parameter.

Note: The iFL statement for the value will perform an XPATH into the event document that is received by the nHTTP listener. It will extract the parameter value from this document that holds the value for the parameter named *file*. This parameter from the incoming document will contain the file title that will be retrieved. The following is a sample of the event signal document returned by nHTTP with a marked *file* parameter. In this example, the file SREG will be set to myfile.txt.

The following is a sample URL provided by the user:

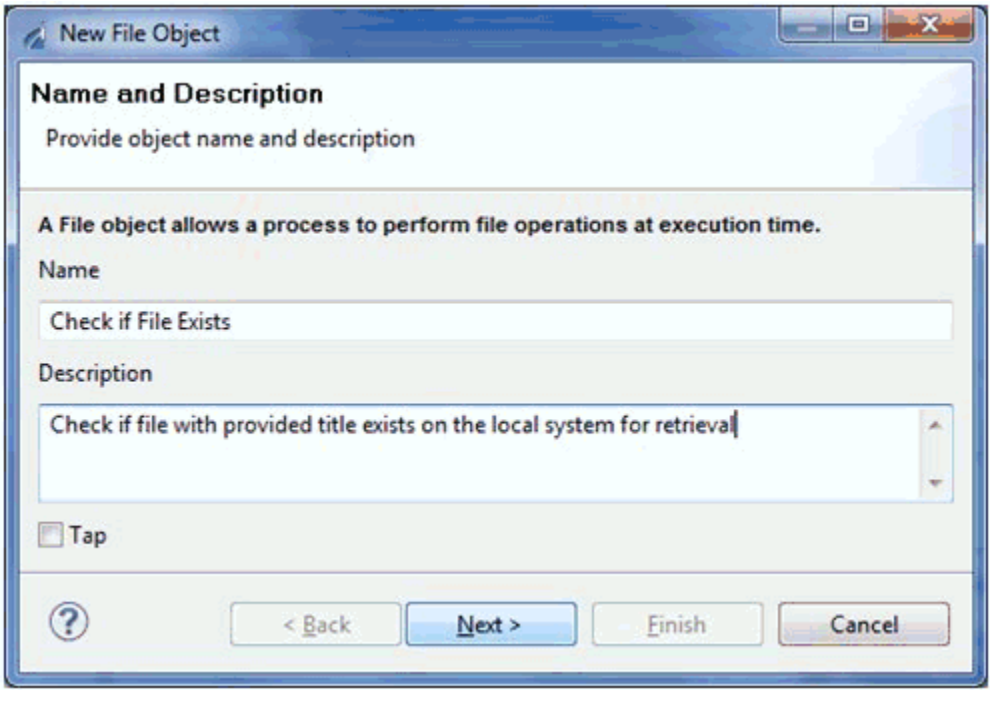
```
http://localhost:2222/get?file=myfile.txt
```

The following is a sample event document that is generated:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <http user="auto" type="GET">
  - <parms>
      <parm name="ibse-port">9000</parm>
      <parm name="Cookie">iwaybamlogin=loggedin; rolename=administrator; username=admin; password=iway;
        tntreport=false; country=; depot=; custombamauth=undefined; PSCount=25; actFields1=; actFields1Desc=;
        minmaxHeightChannel=60; minmaxHeightTrans=35; minmaxHeightPartner=60; toolbarTransCount=0</parm>
      <parm name="Host">localhost:2222</parm>
      <parm name="Connection">Keep-Alive</parm>
      <parm name="pdm">0</parm>
      <parm name="version">1.1</parm>
      <parm name="Accept-Language">en-US</parm>
      <parm name="action">get?file=myfile.txt</parm>
      <parm name="Accept">*/*</parm>
      <parm name="b">false</parm>
      <parm name="User-Agent">Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR
        2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; MDDR; MS-RTC LM 8)</parm>
      <parm name="url">/get?file=myfile.txt</parm>
      <parm name="ip">127.0.0.1</parm>
      <parm name="source">hostname unknown</parm>
      <parm name="Accept-Encoding">gzip, deflate</parm>
      <parm name="reqType">GET</parm>
    </parms>
    <body />
  - <url secure="false">
      <host>localhost</host>
      <port>2222</port>
      <path>/get</path>
      <query>file=myfile.txt</query>
    - <queryparms>
        <queryparm name="file">myfile.txt</queryparm>
      </queryparms>
      <incomingurl>http://localhost:2222/get?file=myfile.txt</incomingurl>
    </url>
    <version>1.1</version>
  </http>
```

11. Connect the Start object and the Set Title Register object using the On Completion relation.

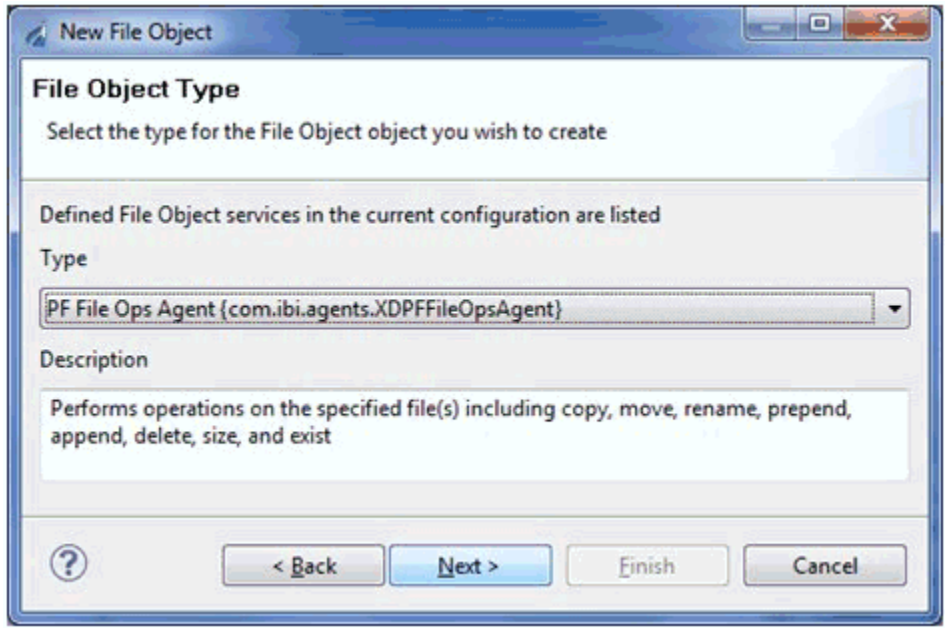
12. Add a File object named Check if File Exists to the process flow, as shown in the following image.



This File object checks to see if the file with the given title exists.

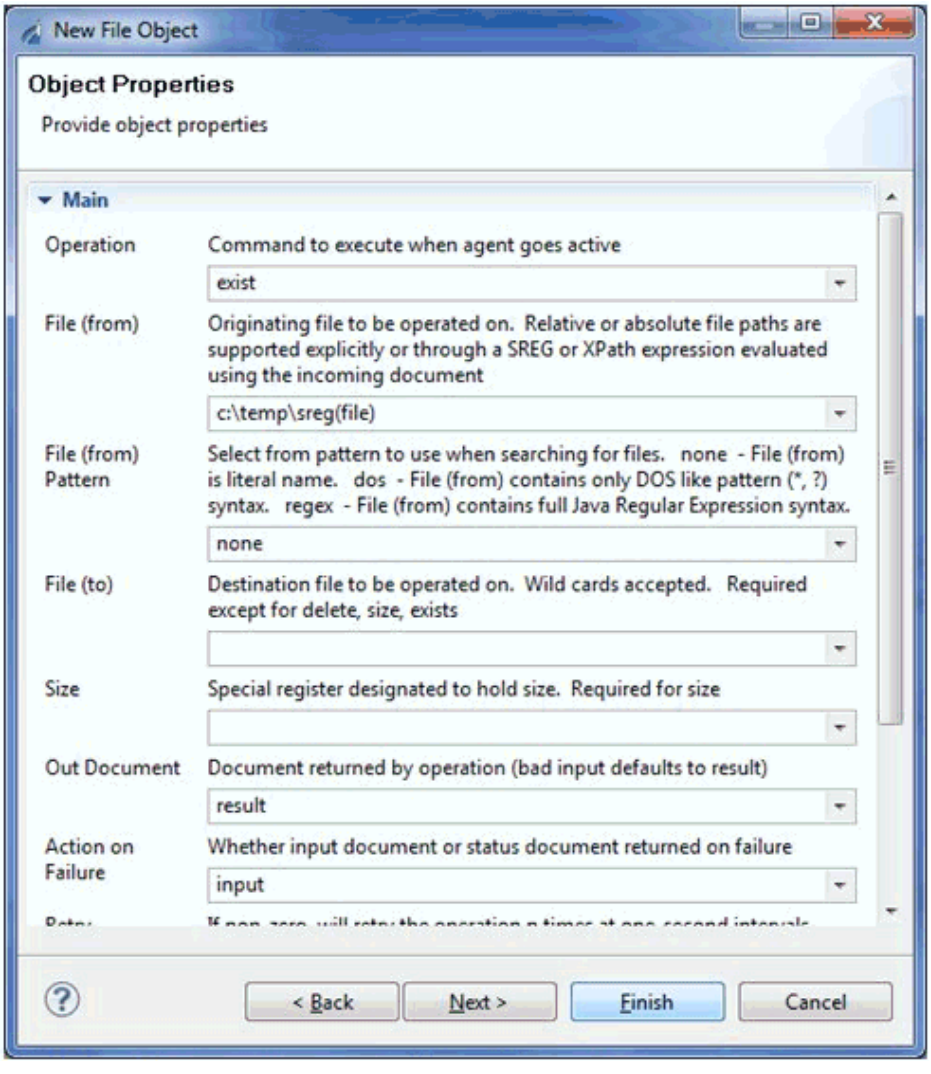
13. Click *Next*.

The File Object Type pane opens, as shown in the following image.



14. Select *PF File Ops Agent {com.ibi.agents.XDPFFileOpsAgent}* from the Type drop-down list.
15. Click *Next*.

The Object Properties pane opens, as shown in the following image.



16. Provide values for the parameters as listed in the following table.

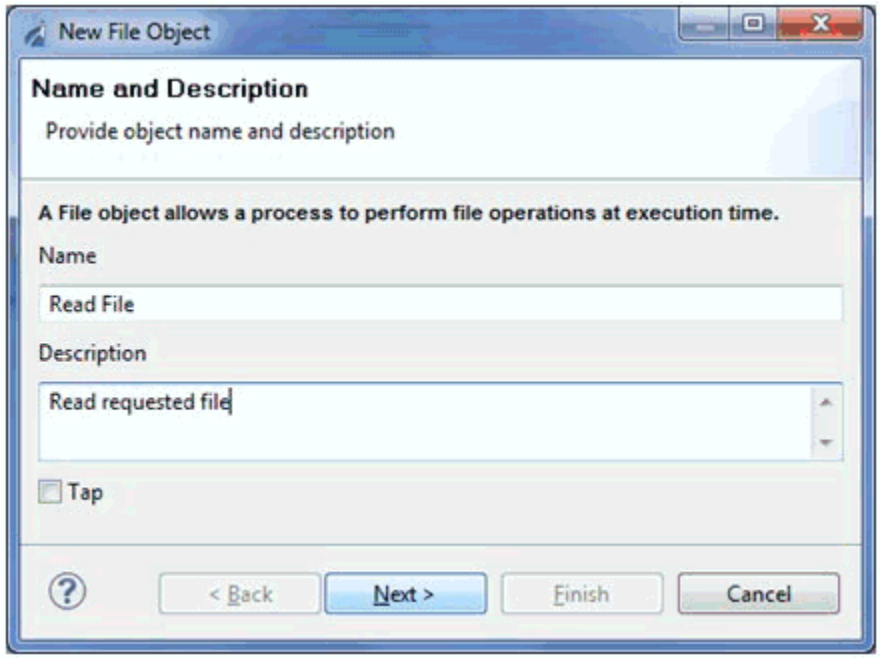
Parameter	Value
Operation	exist
File (from)	c:\temp\sreg(file)

Parameter	Value
Out Document	Result
Action on Failure	input

17. Click *Finish*.
18. Connect the Set Title Register object and the Check if File Exists object using the On Completion relation.

    Note: Error handling is not being configured in this sample configuration. However, it is recommended to add error handling in production applications.

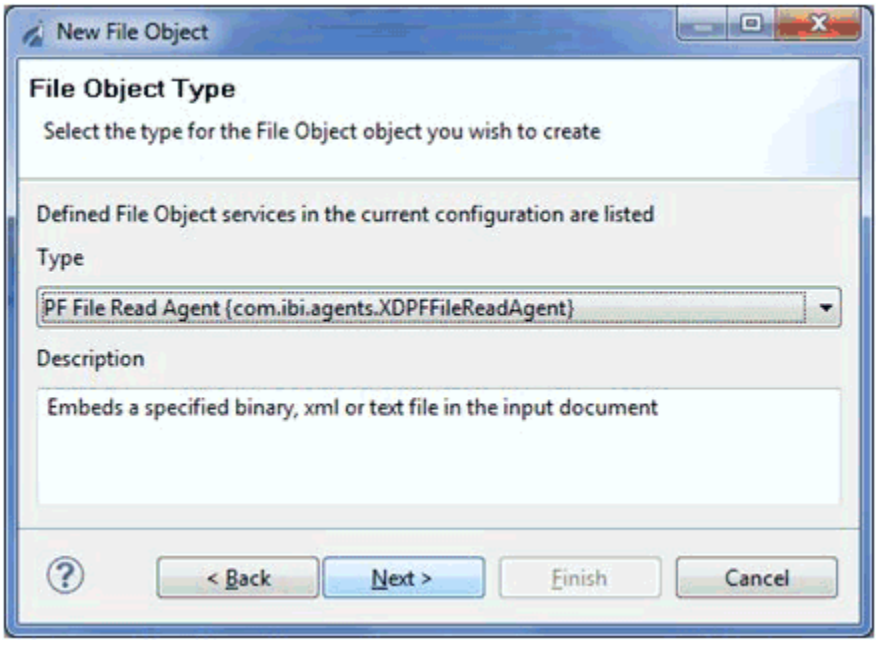
19. Add a File object named Read File to the process flow, as shown in the following image.



    This File object reads the requested file if it exists.

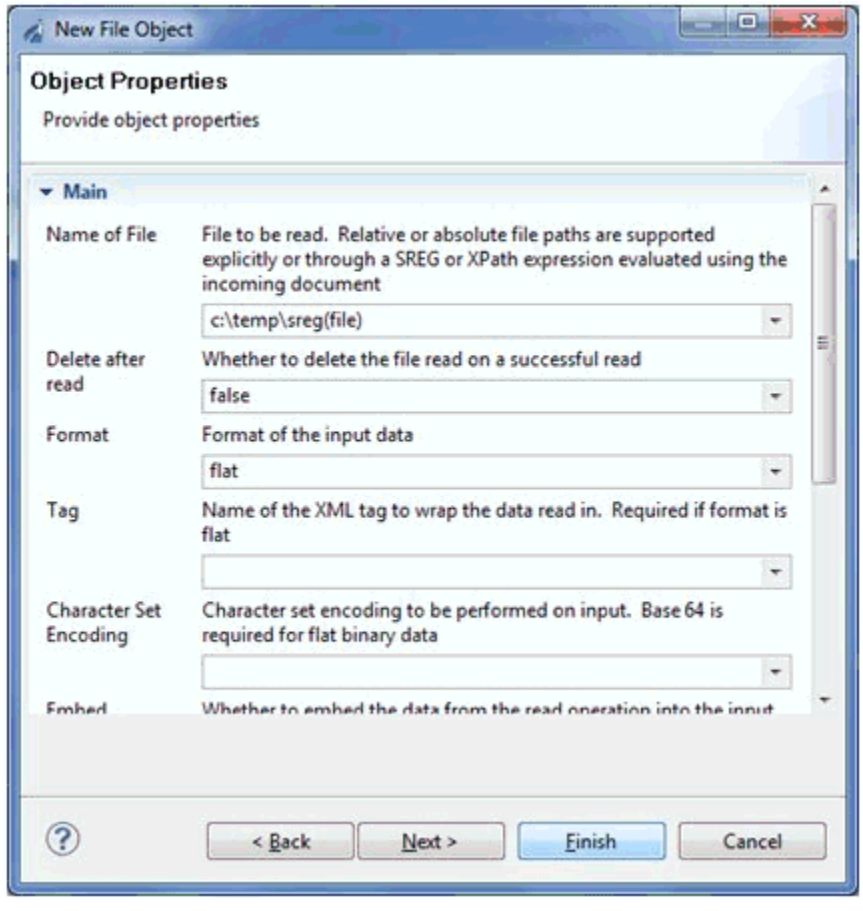
20. Click *Next*.

The File Object Type pane opens, as shown in the following image.



21. Select *PF File Read Agent {com.ibi.agents.XDPFFileReadAgent}* from the Type drop-down list.
22. Click *Next*.

The Object Properties pane opens, as shown in the following image.

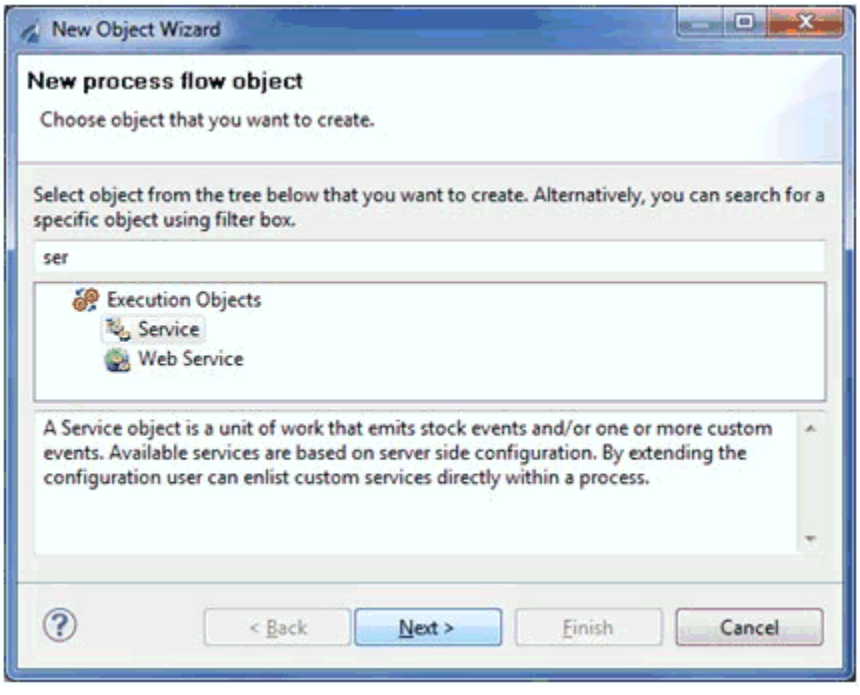


23. Provide values for the parameters as listed in the following table.

Parameter	Value
Name of File	c:\temp\sreg(file)
Response Type	flat

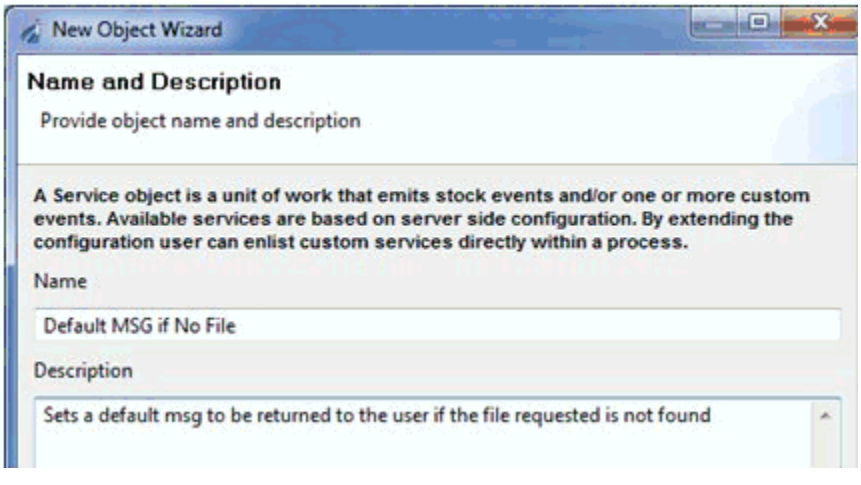
24. Click *Finish*.
25. Connect the Check if File Exists object and the Read File object using the On Success relation.

26. Add another Service object to the process flow.

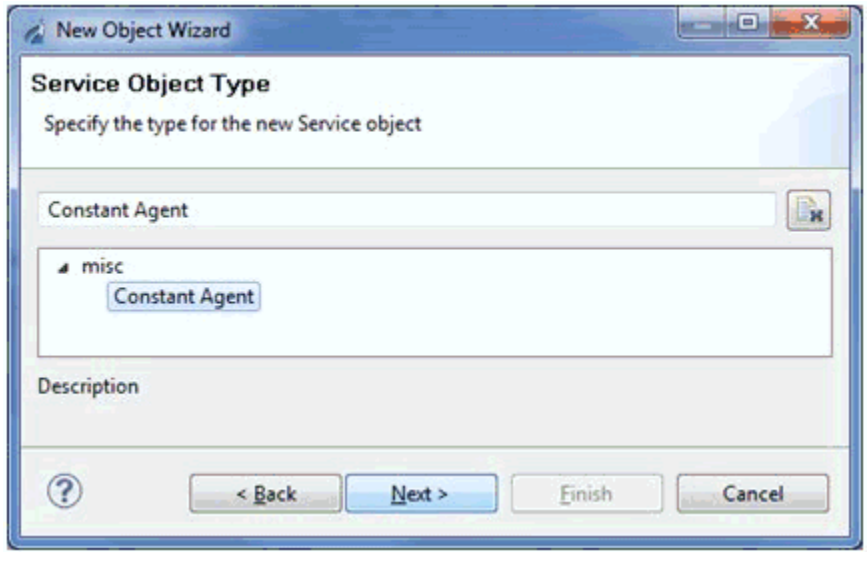


This Service object will set a default error message if the requested file is not found in the specified location.

27. Name the new Service object as Default MSG if No File.

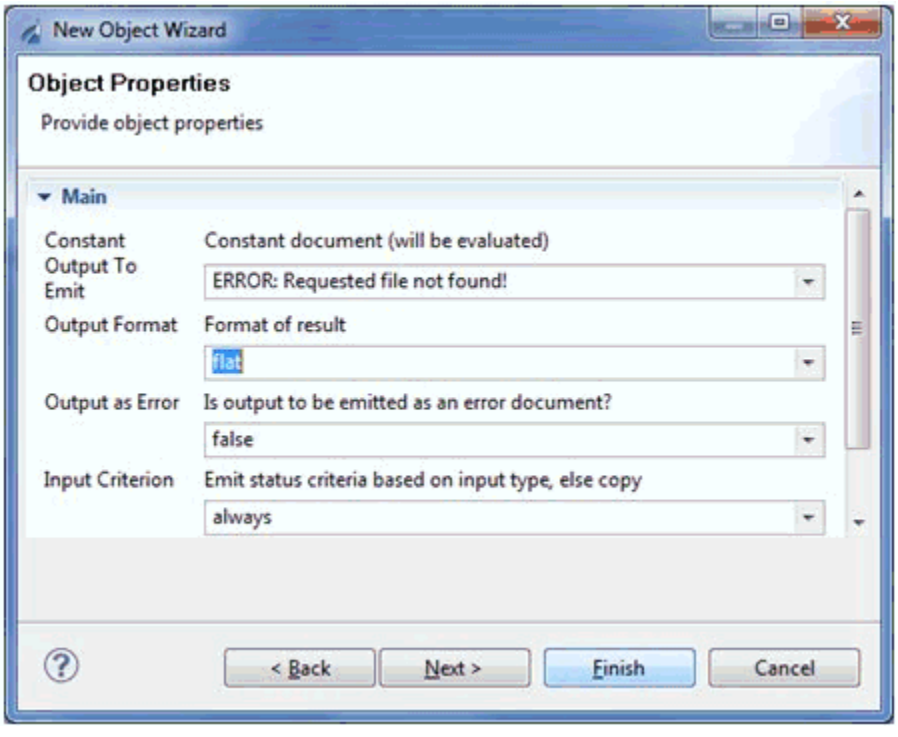


28. Select *Constant Agent* as the service type, which sets the current content of the document to the application provided value.



29. Click *Next*.

The Object Properties pane opens, as shown in the following image.

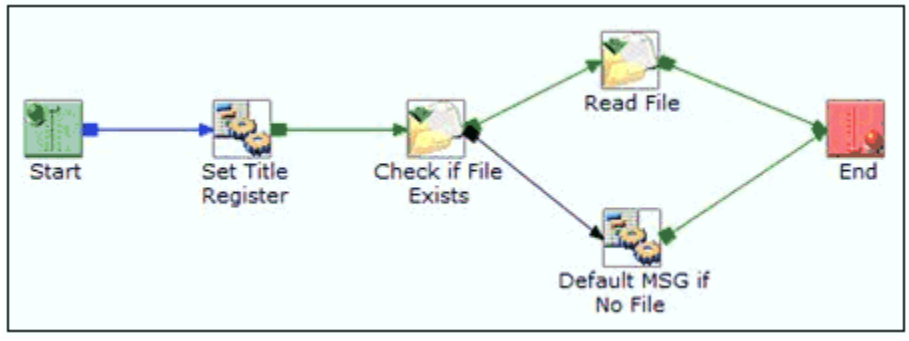


30. Provide values for the parameters as listed in the following table.

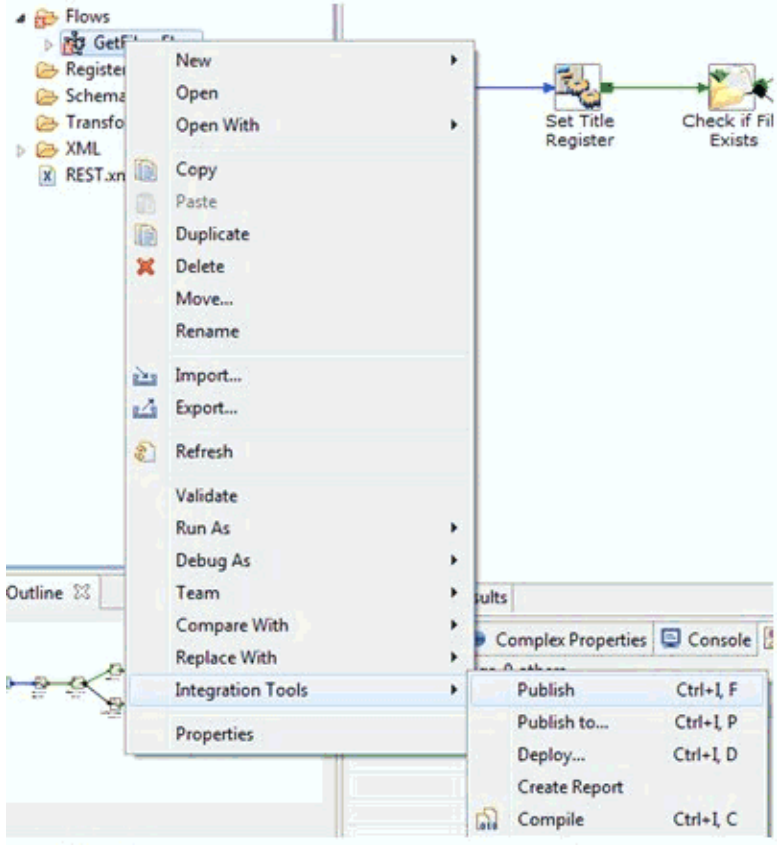
Parameter	Value
Constant Output To Emit	Error: Requested file not found! This indicates to the user that the file does not exist in the current system.
Output Format	flat

31. Click *Finish*.
32. Connect the Check if File Exists object and Default MSG if No File object using the On Custom (not found) relation.
33. Add an End object to the process flow.
34. Connect the Read File object to the End object using the On Success relation.
35. Connect the Default MSG if No File object to the End object using the On Success relation.

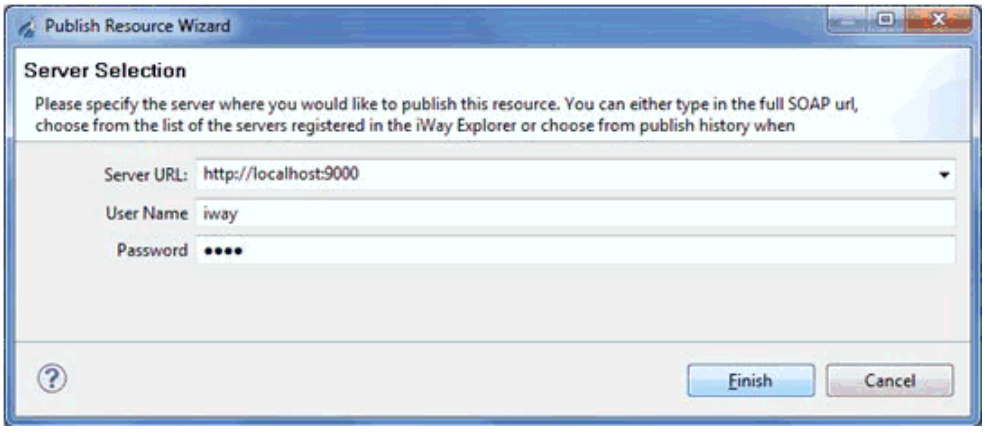
The following image shows the completed process flow.



36. Right-click the *GetFile.pFlow* process flow, select *Integration Tools*, and then click *Publish*, as shown in the following image.



The Publish Resource Wizard opens, as shown in the following image.



37. Provide the required connection parameters to iSM and click *Finish*.

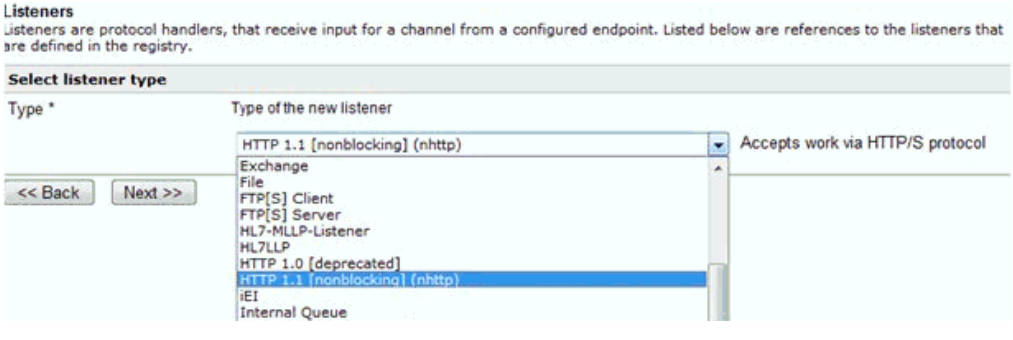
Configuring an nHTTP Channel

The nHTTP channel includes an nHTTP listener running on port 2222 and is used to generate event documents for HTTP GET requests. The channel has a route containing the process flow created in the first section. It will contain a default outlet to allow responses on the requested nHTTP listener.

Procedure: How to Configure an nHTTP Channel

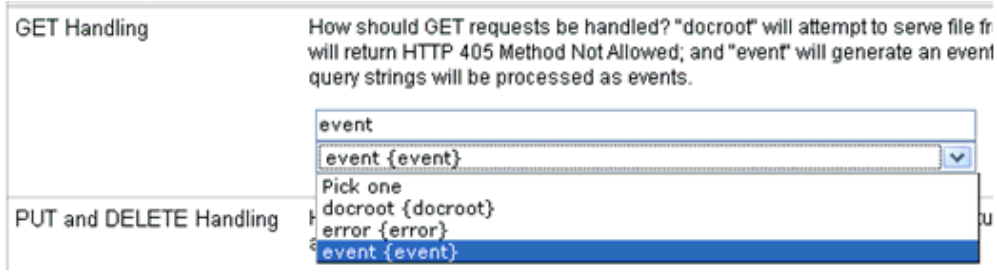
To configure an nHTTP channel:

1. Create an nHTTP Listener by going to the iWay Service Manager Administration Console and navigating to the Listeners section under the Registry. Select *HTTP 1.1 [nonblocking] (nhttp)*, as shown in the following image.

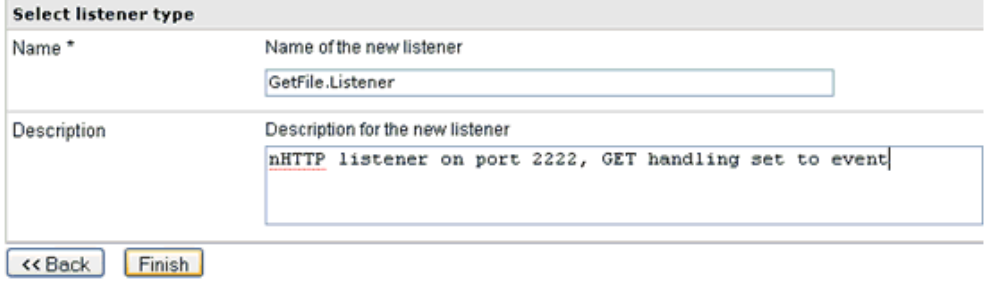


2. Click *Next* and provide listener configuration. In this example, listening for events occurs on HTTP port 2222. The configuration is port 2222. Under GET Handling, select *Event*, leaving all other parameters with the default settings.

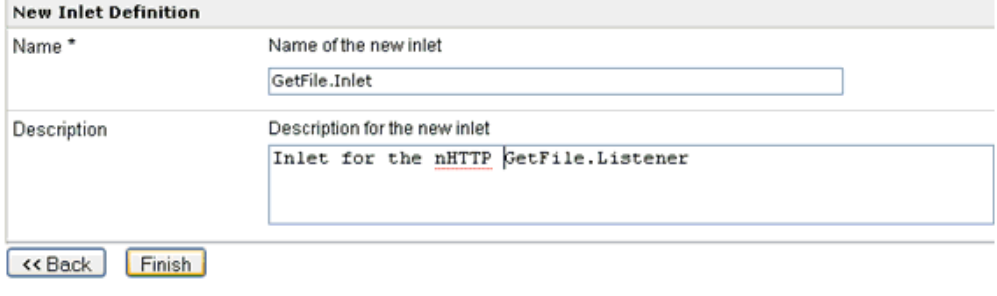
   When the listener receives an HTTP GET request, it generates an event document which will be sent to the route.



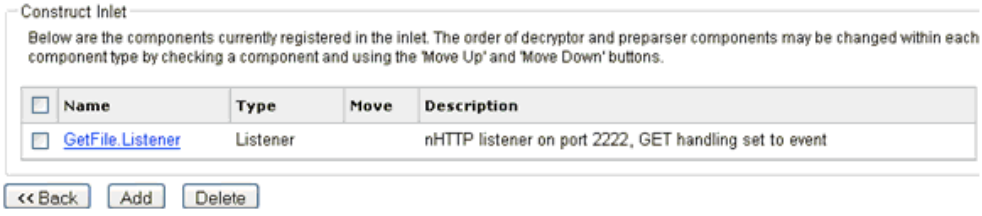
3. Click *Next* and provide a listener name as *GetFile.Listener* with an optional description. Click *Finish*.



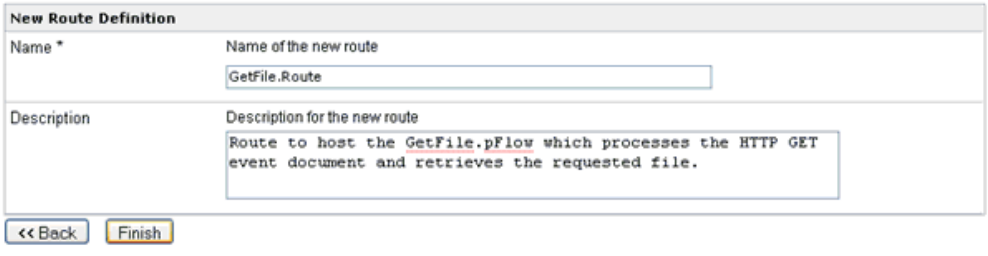
4. Create a new inlet which will contain the listener. Under the registry go to inlet section and select *New*. Provide inlet name as *GetFile.Inlet* with an optional description. Click *Finish*.



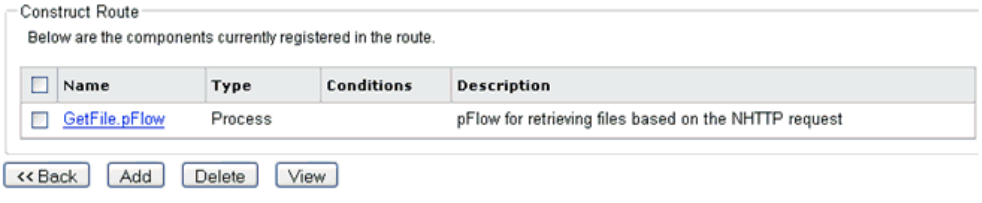
5. Add the *GetFile.Listener* to the *GetFile.Inlet*.



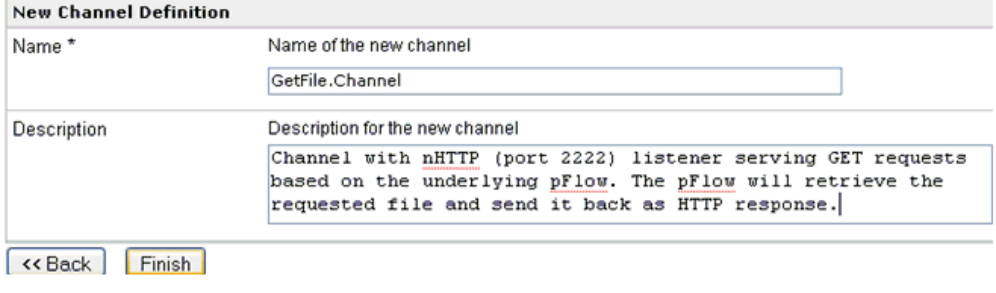
6. Create a new route to host the process flow which was created. Under the Registry section, click *routes*, and click *New* to create a new route. Provide a name *GetFile.Route* with an optional description. Click *Finish*.



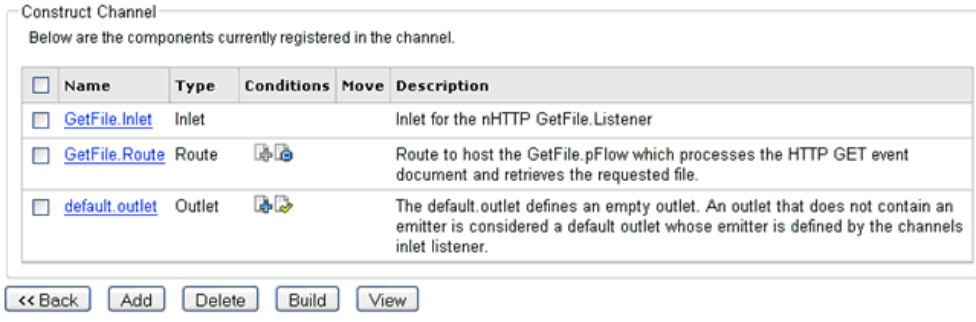
7. Add *GetFile.pFlow* to the GetFile.Route.



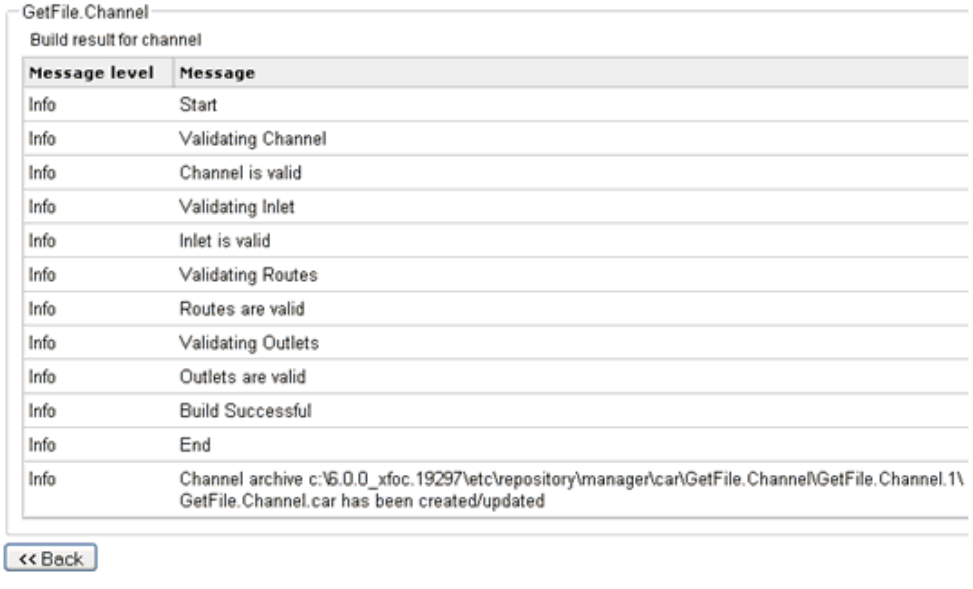
8. Create a new channel by selecting *Channels* under the Registry. Provide Channel name as *GetFile.Channel* with optional description. Click *Finish*.



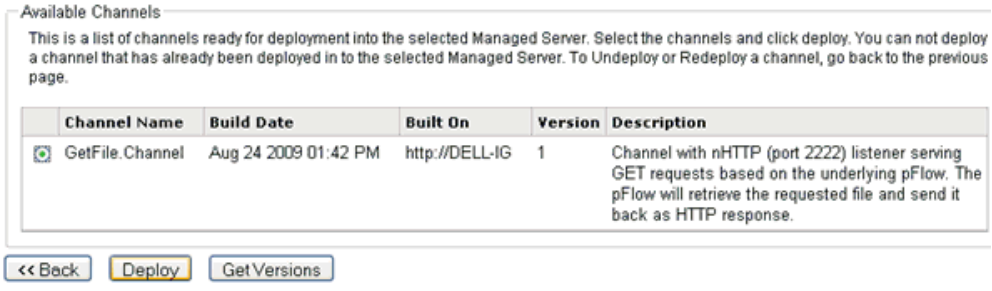
9. Add the *GetFile.Inlet*, *GetFile.Route* and a default outlet to the *GetFile.Channel*. The default outlet will indicate that the response has to be returned back on the HTTP listener's response. Set the GetFile.Route as a default route by clicking the '-' sign under the conditions column. This will indicate that the route is a default one and no condition exists to execute.



10. Once the channel is configured, click *Build* to build the channel. Ensure that the channel builds without any errors or warnings.



11. Now the channel can be deployed under the Deployments section of iSM Console. Click *Deploy* to deploy a new channel. Under the available channels, select *GetFile.Channel* and click *Deploy*.



12. Now you can start your channel under the channels page by clicking its status or by selecting a channel and clicking *Start*.

Running a REST Style Service

After you have configured a process flow and a nHTTP channel, you are ready to execute your REST style service.

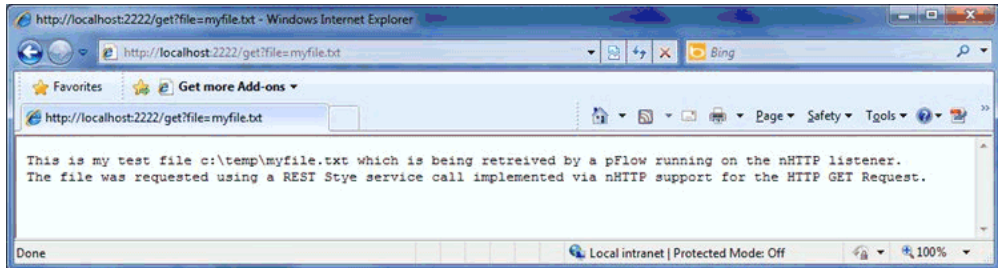
Procedure: How to Run a REST Style Service

To run a REST style service:

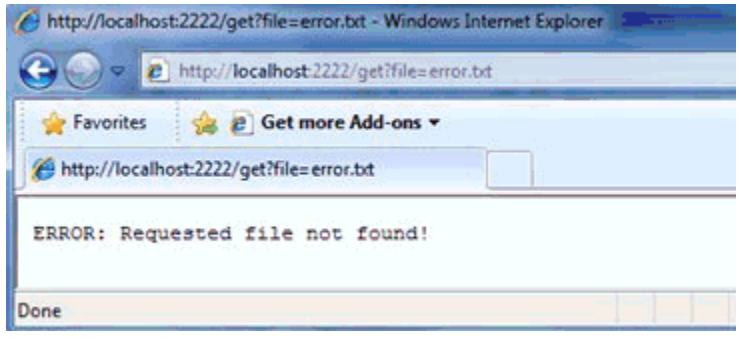
1. Create a simple text file under the c:\temp\ directory (or if you configured for another directory, use the one you configured in the process flow). Name the file *myfile.txt*. This will be a file that we'll try to retrieve using the HTTP REST-Style service call.
2. Open a web browser and issue a REST Style service call request by requesting the following URL:

   ```
   http://localhost:2222/get?file=myfile.txt
   ```

The content of the file will be returned for display, as shown in the following image.



The created process flow also contains logic to generate an error for a file that does not exist. For example, if you provide a request URL, such as `http://localhost:2222/get?file=error.txt`, then the following error message will be returned in the browser indicating that the file is not found.



SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE "LICENSE" FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

TIBCO, the TIBCO logo, the TIBCO O logo, FOCUS, iWay, Omni-Gen, Omni-HealthData, and WebFOCUS are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle Corporation and/or its affiliates.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

This software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the readme file for the availability of this software version on a specific operating system platform.

THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.