TIBCO ActiveMatrix®
Implementation Type for WCF Host

User's Guide

Software Release 1.0
June 2009

TIBCO®
The Power of Now®

Important Information

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 LICENSE.PDF) 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.

This document contains confidential information that is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIB, TIBCO, TIBCO ActiveMatrix, TIBCO Adapter, TIBCO Administrator, TIBCO AutoMeditate, TIBCO Enterprise Message Service, ActiveMatrix, AutoMediate, Predictive Business, Information Bus, The Power of Now, and TIBCO Rendezvous are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

EJB, Java EE, J2EE, and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

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. PLEASE SEE THE README.TXT 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.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

Copyright © 2005-2009 TIBCO Software Inc. ALL RIGHTS RESERVED.

TIBCO Software Inc. Confidential Information

Contents

Tables

Preface

TIBCO ActiveMatrix® Implementation Type for WCF Host is a scalable and extensible platform for developing, deploying, and managing applications that conform to a service-oriented architecture.

Topics

Related Documentation

This section lists documentation resources you may find useful.

TIBCO ActiveMatrix Implementation Type for WCF Host Documentation

The following documents form the TIBCO ActiveMatrix Implementation Type for WCF Host documentation set:

- *TIBCO ActiveMatrix Implementation Type for WCF Host User's Guide:* Read the user's guide to learn how to develop WCF Host components.
- *TIBCO ActiveMatrix Implementation Type for WCF Host Release Notes:* Read the release notes for a list of new features. This manual also contains lists of known issues for this release.

Other TIBCO Product Documentation

You may find it useful to read the documentation for the following TIBCO products:

- TIBCO ActiveMatrix® Service Bus
- TIBCO ActiveMatrix® Service Grid
- TIBCO ActiveMatrix® Registry
- TIBCO ActiveMatrix® Policy Manager

Third Party Documentation

TIBCO ActiveMatrix software supports the following standards:

- Service Component Architecture

  http://www.osoa.org/display/Main/Service+Component+Architecture+Specifications

- World Web Web Consortium web services activity

  http://www.w3.org/2002/ws/

  — Simple Object Access Protocol (SOAP) 1.1 W3C Note

  http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

  — WSDL 1.1 W3C Note

  http://www.w3.org/TR/wsdl

- OASIS

  http://www.oasis-open.org/committees/tc_cat.php?cat=ws

  — UDDI Version 3 OASIS Standard

    http://www.oasis-open.org/specs/index.php#uddiv2

Typographical Conventions

The following typographical conventions are used in this manual.

Table 1 General Typographical Conventions

Convention	Use
TIBCO_HOME *ENV_NAME* *ENV_HOME* *AMX_HOME* *AMX_ADMIN_HOME*	Many TIBCO products are installed within the same home directory. This directory is referenced in documentation as *TIBCO_HOME*. The value of *TIBCO_HOME* depends on the operating system. For example, on Windows systems the default value is `C:\tibco`. Other TIBCO products are installed into an installation environment. A product installed into an installation environment does not access components in other installation environments. Incompatible products and multiple instances of the same product must be installed into different installation environments. An installation environment consists of the following properties: • **Name** Identifies the installation environment. The name is appended to the name of Windows services created by the installer and is used in the path to the product in the Windows Start > All Programs menu. This directory is referenced in documentation as *ENV_NAME*. • **Description** Provides information about what the environment contains or is used for. • **Path** The directory into which the product is installed. This directory is referenced in documentation as *ENV_HOME*. The value of *ENV_HOME* depends on the operating system. For example, on Windows systems the default value is `C:\tibco`. TIBCO ActiveMatrix installs into a directory inside *ENV_HOME*. This directory is referenced in documentation as *AMX_HOME*. The value of *AMX_HOME* depends on the operating system. For example, on Windows systems the default value is `C:\tibco\amx\`. TIBCO ActiveMatrix Administrator installs into a directory inside *ENV_HOME*. This directory is referenced in documentation as *AMX_ADMIN_HOME*. The value of *AMX_ADMIN_HOME* depends on the operating system. For example, on Windows systems the default value is `C:\tibco\amxadministrator\`.
`code font`	Code font identifies commands, code examples, filenames, pathnames, and output displayed in a command window. For example: Use `MyCommand` to start the foo process.

Table 1 General Typographical Conventions (Cont'd)

Convention	Use
`bold code font`	Bold code font is used in the following ways: • In procedures, to indicate what a user types. For example: Type **`admin`**. • In large code samples, to indicate the parts of the sample that are of particular interest. • In command syntax, to indicate the default parameter for a command. For example, if no parameter is specified, `MyCommand` is enabled: MyCommand [**`enable`** | disable]
italic font	Italic font is used in the following ways: • To indicate a document title. For example: See *TIBCO BusinessWorks Concepts*. • To introduce new terms For example: A portal page may contain several portlets. *Portlets* are mini-applications that run in a portal. • To indicate a variable in a command or code syntax that you must replace. For example: `MyCommand` *pathname*
Key combinations	Key name separated by a plus sign indicate keys pressed simultaneously. For example: Ctrl+C. Key names separated by a comma and space indicate keys pressed one after the other. For example: Esc, Ctrl+Q.
	The note icon indicates information that is of special interest or importance, for example, an additional action required only in certain circumstances.
	The tip icon indicates an idea that could be useful, for example, a way to apply the information provided in the current section to achieve a specific result.
	The warning icon indicates the potential for a damaging situation, for example, data loss or corruption if certain steps are taken or not taken.

Table 2 Syntax Typographical Conventions

Convention	Use
`[ ]`	An optional item in a command or code syntax. For example: `MyCommand [optional_parameter] required_parameter`
`|`	A logical 'OR' that separates multiple items of which only one may be chosen. For example, you can select only one of the following parameters: `MyCommand para1 | param2 | param3`

`{ }`	A logical group of items in a command. Other syntax notations may appear within each logical group.
	For example, the following command requires two parameters, which can be either the pair `param1` and `param2`, or the pair `param3` and `param4`.
	`MyCommand {param1 param2} | {param3 param4}`
	In the next example, the command requires two parameters. The first parameter can be either `param1` or `param2` and the second can be either `param3` or `param4`:
	`MyCommand {param1 | param2} {param3 | param4}`
	In the next example, the command can accept either two or three parameters. The first parameter must be `param1`. You can optionally include `param2` as the second parameter. And the last parameter is either `param3` or `param4`.
	`MyCommand param1 [param2] {param3 | param4}`

How to Contact TIBCO Support

For comments or problems with this manual or the software it addresses, please contact TIBCO Support as follows.

- For an overview of TIBCO Support, and information about getting started with TIBCO Support, visit this site:

  http://www.tibco.com/services/support

- If you already have a valid maintenance or support contract, visit this site:

  https://support.tibco.com

  Entry to this site requires a username and password. If you do not have a username, you can request one.

Chapter 1 Installation

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. Please see the readme.txt file for the availability of this software version on a specific operating system platform.

This chapter describes how to install TIBCO ActiveMatrix Implementation Type for WCF Host on all supported platforms.

Topics

Installation Overview

This section provides an overview of TIBCO ActiveMatrix Implementation Type for WCF Host installation.

Installation Modes

Three installation modes are available: GUI, console, and silent. For details on each mode, see Installing TIBCO ActiveMatrix Implementation Type for WCF Host on page 7.

Installation Types

Two installation types are available: typical and custom. The typical installation type installs all the products in the package on the specified platform. The custom installation type allows you to select the components to be installed. When you select the custom installation type, the following components are available:

- **Design** Installs the design time component of the product.
- **Runtime** Installs the runtime component of the product.

Supported Platforms

Windows

- Windows Server 2008 on x86 and x86-64
- Windows Server 2003 on x86 and x86-64
- Windows XP Professional SP2 on x86

Installer Account

Microsoft Windows

You must have administrator privileges for the machine on which TIBCO ActiveMatrix Implementation Type for WCF Host is installed. If you do not have administrator privileges, the installer exits. You must then log out of the system and log in as a user with the required privileges, or request your system administrator to assign the privileges to your account.

If you want to install the product on a network drive, you must ensure that the account used for installation has permission to access the network drive.

On Microsoft Windows 2003, before you start the installation, ensure that the files `vpd.properties` and `vpd.properties.tibco.`*hostname*, are available in the *USER_HOME*`\Windows\` folder. If not, copy these files from `C:\Windows\` to the *USER_HOME*`\Windows\` directory.

Installer Log File

The installer log file, `tibco_universal_installer.`*timestamp*`.`*username*`_install.txt`, is written to the `.TIBCO` folder of the user's home directory. The installer log file captures the following information:

- Installation environment details such as the user that invoked the installer, hostname, Java home in the environment, operating system details, and so on.
- List of assemblies installed.

System Memory

A minimum of 2 GB of physical memory is required.

Disk Space

Disk Space Before Installation

Before installing TIBCO ActiveMatrix Implementation Type for WCF Host you must extract the contents of the installation archive to a temporary directory. The installer files consume up to 80 MB of disk space.

Temporary Disk Space Required by the Installer

The installer requires at least 70 MB of free space in the temporary directory. On Microsoft Windows, temporary directory location is `%SystemDrive%:\Documents and Settings\`*user_name*`\Local Settings\Temp`

If your system does not have sufficient disk space in the default temporary area, you can run the installer with a different temporary directory by using the following option when starting the installer:

`-is:tempdir /`*new_temp*

where */new_temp* has sufficient free disk space.

The installer calculates the disk space required in the product home location for the selected components before the actual installation (copying of files to system) begins. The installer will proceed only if sufficient free disk space is available in the product home location.

If the disk space required is consumed by another process while the installer is copying the files, the disk space available can be reduced. In such a case, the installer may fail and will then give a failure message.

While performing installation, it is recommended not to run other processes that consume disk space in the product home location.

Disk Space After Installation

TIBCO ActiveMatrix Implementation Type for WCF Host consumes 90 MB of disk space under *TIBCO_HOME*.

Installation Directory

TIBCO ActiveMatrix Implementation Type for WCF Host is installed in the directory *AMX_HOME*`/extensions/wcfhost/1.0`. This directory contains the following subdirectories:

- `doc` Documentation.
- `samples` Sample ActiveMatrix SOA project and a TIBCO General Interface client.
- `uninstaller_archives` Uninstall data.

Software Requirements

This section lists required and optional software products.

TIBCO ActiveMatrix Service Grid 2.2.1

Required. See the respective product documentation for the list of requirements and procedures for installing TIBCO ActiveMatrix Service Grid.

Microsoft .NET Framework 3.0

Required for running services in the WCF Host container.

Microsoft Visual Studio 2008

Required for developing the Microsoft WCF services.

Pre-Installation Procedure

Before installing TIBCO ActiveMatrix Implementation Type for WCF Host, back up your existing `containers` folder that exists under *TIBCO_HOME*`/amx/2.X`:

- Create the folder structure "`backup/`<*AMX Version*>" under *TIBCO_HOME*`/amx/2.X`.

  For example, if you are installing TIBCO ActiveMatrix Implementation Type for WCF Host on top of TIBCO ActiveMatrix Service Grid 2.2.1, create the backup folder structure as *TIBCO_HOME*`/amx/2.2/backup/2.2.1`

- Make a backup of the "`containers`" folder under the newly created `backup` folder.

  For example, copy the "`containers`" folder from *TIBCO_HOME*`/amx/2.2` to *TIBCO_HOME*`/amx/2.2/backup/2.2.1`

Installing TIBCO ActiveMatrix Implementation Type for WCF Host

Before installing TIBCO ActiveMatrix Implementation Type for WCF Host, shut down the Management Daemon process, ActiveMatrix Administrator servers, and ActiveMatrix nodes running on the machine.

After you complete the installation, restart in the following order:

1. ActiveMatrix Administrator server
2. Management Daemon process
3. ActiveMatrix nodes

This section describes the steps to install TIBCO ActiveMatrix Implementation Type for WCF Host using the modes described in the sections:

- Installing in GUI Mode on page 7
- Installing in Console Mode on page 8
- Installing in Silent Mode on page 9

Installing in GUI Mode

In the GUI mode, the installer presents panels that allow you to make choices about product features, product installation location, and so on.

To install the product in GUI mode:

1. Open the physical media or download the TIBCO ActiveMatrix Implementation Type for WCF Host package.
2. Extract the contents of the package to a temporary directory.
3. Navigate to the temporary directory.
4. Run **TIBCOUniversalInstaller**.
5. The Welcome screen appears. Click **Next**.
6. The License Agreement screen appears. After reading through the license text, click **I accept the terms of the license agreement** and then click **Next.**
7. The Important Notice screen appears. Read through the information and click **Next**.
8. In the Environment drop-down list, select the installation environment you selected for TIBCO ActiveMatrix Service Grid. Click **Next**.

9. Select **Typical** to install all the available features or **Custom** to choose the features to install. Click **Next**.
10. If you selected **Custom** in step 9, the Product Features screen appears. This screen lists the components available for installation. By default, all the components are selected. Uncheck the checkbox next to the component you don't want installed and click **Next**. If you did not select Custom, proceed to step 11.
11. The Pre Install summary screen appears. Verify the list of products selected for install and then click **Install**. Click **Yes to All** to close any popup windows that display during installation.
12. The next **TIBCO ActiveMatrix .NET** dialog reminds you that the VC8-compiled TIBCO EMS C# Client must be installed on the sytem, as well as in the GAC, before ActiveMatrix Implementation Type for WCF Host can be used. For more information about this requirement, see Post-Installation Procedures in *TIBCO ActiveMatrix Service Grid Installation* guide. Click **Next**.
13. The Post Install Summary screen appears. This screen summarizes the installation process. Click **Finish** to complete the installation process and close the installer window.

Installing in Console Mode

In the console mode, you can install the product in a non-Windows environment.

The following procedure explains how to install the product in console mode.

1. Open the physical media or download the TIBCO ActiveMatrix Implementation Type for WCF Host package.
2. Extract the contents of the package to a temporary directory.
3. Using a console window, navigate to the temporary directory.
4. Run `TIBCOUniversalInstaller.exe -console`.
5. Complete the installation by responding to the console window prompts.

While running in the console mode, you can use the following keys to move through the installation process:

`Enter`: Moves forward in the installer.

`BackSpace or 2`: Goes back to the previous screen.

`3`: Cancels the Wizard and exits the installation or uninstallation.

`5`: Redisplays the current screen.

Installing in Silent Mode

In silent mode, the universal installer does not prompt for any inputs during installation. Instead, the inputs are read from a response file that can be provided as a command line parameter. If no value is specified, the installer uses the default `TIBCOUniversalInstaller.silent` file.

The `TIBCOUniversalInstaller.silent` file is packaged in the directory that contains the universal installer. You must edit the file with information for your environment before launching the silent installation. The file includes comments that describe the installation properties you can set. While you can use the `TIBCOUniversalInstaller.silent` file, it's good practice to copy the file to a different name and use that file for the silent install.

The following procedure explains how to install the product in silent mode. If errors occur during installation, they will be listed in the installation log file contained in *User_Home*`/.TIBCO` directory.

1. Open the physical media or download the TIBCO ActiveMatrix Implementation Type for WCF Host package.
2. Extract the contents of the package to a temporary directory.
3. Using a console window, navigate to the temporary directory.
4. Make a copy of the `TIBCOUniversalInstaller.silent` file and rename the file.
5. Using a text editor, open the copied file and update the install location and features to install.
6. Run **TIBCOUniversalInstaller -silent -V responseFile="***myfilename***.silent"**.

   When installation completes, a line similar to the following is written to the installer log file:

   ```
   Install, com.tibco.installer.util.TIBCOInstaller, dbg.Debug,
   The installation has completed. Please check the log file for
   additional information.
   ```

Post-Installation Procedures

This section describes post-installation procedures you must perform before deploying and running WCF Host components.

Enable and Activate WCF Host Containers on Existing ActiveMatrix Nodes

Enable and activate the WCF Host container on the nodes that were created before you installed TIBCO ActiveMatrix Implementation Type for WCF Host 1.0.0. For information on how to enable and activate a container, see the Working with Containers section in Chapter 7 of *TIBCO ActiveMatrix Administration* guide.

Uninstalling the Software

To uninstall TIBCO ActiveMatrix Implementation Type for WCF Host:

1. Remap service units containing WCF Host components and then relocate service assemblies:
   a. In ActiveMatrix Administrator, remap all WCF Host service units to nodes running on other ActiveMatrix machines.
   b. Relocate the service assemblies that contain the remapped service units.
2. Stop any nodes with activated WCF Host containers.
3. Stop any instances of TIBCO Business Studio.
4. Run the uninstaller to uninstall TIBCO ActiveMatrix Implementation Type for WCF Host:
   a. Navigate to the *TIBCO_HOME*_uninstall directory and run **universal_uninstall**. A splash screen displays. Click **Next**.
   b. Keep the default selection **Custom Uninstall**. Click **Next**.
   c. A screen displays uninstall options for all the installed TIBCO ActiveMatrix products. See *TIBCO ActiveMatrix Service Grid Installation* for details about uninstalling ActiveMatrix products. Deselect all checkboxes except TIBCO ActiveMatrix Implementation Type for WCF Host and then click **Next**.

      When uninstalling, popup windows appear warning that an updated file needs to be deleted. Click **Yes to All** to remove the file and subsequent files in the assembly. Because popup windows are generated for each installed assembly, you may see multiple popup windows.
   d. Review the Pre-Uninstall Summary and click **Uninstall**.
   e. The summary screen appears. Click **Finish** to exit the uninstall wizard. After uninstalling the software, the uninstaller may prompt you to reboot your computer.
5. Follow the Post-Uninstallation Procedure, page 12.

Post-Uninstallation Procedure

Install the previous version of ActiveMatrix Services Support `dlls` as follows:

1. Ensure that `AMXServicesSupport.dll` and `CommonLoggingTraceListener.dll` are removed from the Global Assembly Cache.
2. Copy the "`containers`" directory that you backed up as a part of the Pre-Installation Procedure to *TIBCO_HOME*`/amx/2.X` that will overwrite the existing containers folder.

   For example, copy the "`containers`" directory from the folder *TIBCO_HOME*`/amx/2.2/backup/2.2.1` into *TIBCO_HOME*`/amx/2.2`.
3. Navigate to *TIBCO_HOME*`/amx/2.X/containers/.NET/bin/runtime`.
4. Run `TIBCO ActiveMatrix Services Support GAC Assembly Registration.msi`.
5. The **TIBCO ActiveMatrix Setup** wizard appears.
6. Complete the screens in the wizard to install the previous `dll` versions into the Global Assembly Cache.

Chapter 2 Developing WCF Host Components

The process of creating a WCF Host component and component implementation involves Microsoft Visual Studio and TIBCO Business Studio. You create the WCF Host components after you create the component implementation.

Topics

Overview

Microsoft Windows Communication Foundation (WCF) is a service-oriented programming model for constructing distributed applications.

A WCF service is composed of three parts:

- a service class that implements the service contract to be provided,
- an environment to host the service, and
- one or more endpoints to which clients will connect.

Once you have designed and implemented the service contract, you configure how the service is exposed to clients. This involves specifying the address where the services can be found, the transport and message encoding it uses to send and receive messages, and the type of security it requires. WCF supports two ways to configure services: in code and through configuration files.

ActiveMatrix Implementation Type for WCF Host integrates WCF services with the ActiveMatrix platform. A WCF Host component functions as the host environment for a Microsoft WCF service. A Microsoft WCF service hosted using the Implementation Type for WCF Host can consume services deployed within the ActiveMatrix environment as well as external Microsoft WCF Services.

ActiveMatrix Implementation Type for WCF Host requires that you generate WSDL files that represent the WCF services. Also, it requires that you configure a services and references within an application configuration file. After you import the WCF Host implementation, the associated WSDL/schemas, and the application config file into TIBCO Business Studio, amend the application configuration file with ActiveMatrix specific configuration in the WCF Config File Editor.

Development Process Summary

The process for developing WCF Host components can be summarized as follows:

Microsoft Visual Studio

1. Create implementation assemblies and application configuration files. If the implementation is a client of an ActiveMatrix service, reference the service proxy following the procedures described in Invoking ActiveMatrix Services from WCF Host Component Implementations on page 22.

Microsoft svcutil

2. Generate WSDL files from implementation assemblies. See Generating WSDL Files From WCF Services on page 25.

TIBCO Business Studio

3. Create an ActiveMatrix SOA project.
4. Import assemblies, application files, and WSDL and schema files into the project. See Importing WSDL and Schema Files, Implementation Assemblies and Configuration Files on page 27.

WCF Host Config File Editor

5. Resolve references to schema definitions in WSDL files. See Resolving XSD Schema References on page 27.
6. Map service contracts to port types. Mapping Service Contracts to Port Types on page 30.

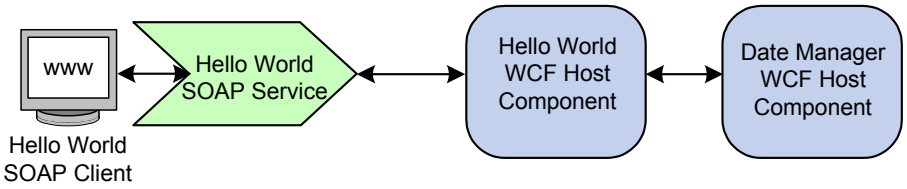
Composite Editor

7. Add one or more WCF Host components to the composite. See Creating a WCF Host Component on page 32.
8. Configure the WCF Host component's implementation. See Configuring a WCF Host Component's Implementation on page 35.
9. Update the WCF Host component's implementation. See Updating a WCF Host Component's Implementation on page 35

Sample Hello World Service

TIBCO ActiveMatrix Implementation Type for WCF Host contains a sample hello world service. As shown in Figure 1, the example consists of two services—Hello World service and Date Manager service—that work together to return an acknowledgement string when a user submits a name to a Hello World SOAP client.

Figure 1 Hello World Example



The hello world sample files include:

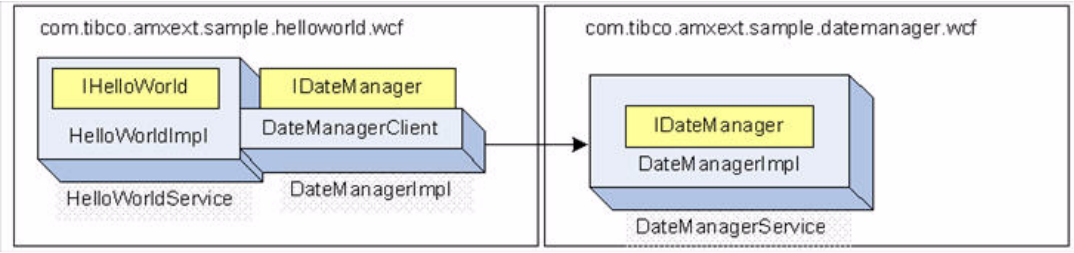
- An ActiveMatrix SOA project containing a composite with two WCF Host components and a SOAP service,
- A deployable service assembly archive, and
- Visual Studio projects containing the WCF Host implementation source for the two services.

The files for the sample are located in the folder *AMX_HOME*`\extensions\wcfhost\1.0\samples\HelloDateTime`. This folder contains a `readme.txt` file that describes how to run the service and invoke it from a client in TIBCO Business Studio and the following subfolders:

- `com.tibco.amxext.sample.hellodatetime.soa` ActiveMatrix SOA project
- `com.tibco.amxext.sample.datemanager.wcf` Visual Studio project implementing Date Manager service
- `com.tibco.amxext.sample.helloworld.wcf` Visual Studio project implementing Hello World service and Date Manager proxy

The relationship between the files in the two Visual Studio projects is illustrated in Figure 2.

Figure 2 Sample Visual Studio Projects



Developing WCF Host Component Implementations

ActiveMatrix imposes certain limitations on WCF services that serve as WCF Host components. There are general limitations that apply to all WCF services and specific limitations that concern the WCF attributes and attribute properties supported by WCF Host components.

It is expected that all proxy references within the WCF implementation code are closed once they are no longer used. This is required for the WCF services hosted within TIBCO ActiveMatrix Service Grid to release resources. If proxy references are not closed, resources are leaked and performance is adversly affected.

General Limitations

WCF services that are hosted in ActiveMatrix interact with clients and other services via ActiveMatrix communication channels. In other words, a WCF Host component does not use a WCF channel to communicate between other ActiveMatrix services or clients external to ActiveMatrix.

Therefore a WCF Host component cannot do the following:

- Configure the channel in runtime code (for example, the code in the `main` method of an existing self hosted WCF service will not be executed). It must be configured in the `app.config` file.
- Access any of the WCF Channel or Framework contexts, such as security or transactions.
- Rely on data being delivered in any of the message headers.
- Support sessionful WCF services. All the deployed services are stateless services and each request is handled as if the binding were sessionless.

In addition,

- Endpoint bindings for WCF Host components are not used at runtime. Only the service behavior of the endpoint is preserved. All other endpoint configuration is disregarded.
- Endpoint binding information for client endpoints (partners) are also disregarded at runtime with the exception of the service behavior and timeout settings (OpenTimeout, CloseTimeout, ReceiveTimeout, SendTimeout).
- Client proxy variables (used to reference ActiveMatrix Service Grid partner services) must be explicity closed. If the client proxies are not closed, threads in the node process grow eventually resulting in the following error:

```
CallProxy error: OutOfMemoryError, msg = unable to create new
native thread
```

Here is a sample showing a service implementation method which is referencing an ActiveMatrix Service Grid service using the clientProxy varaible.

```
        public AddressResponseElement
AddressOperation(AddressRequestElement request)
        {
          AddressServiceClient clientProxy = null;
          AddressResponseElement reply = null;
            try {
              AddressServiceClient clientProxy = new
AddressServiceClient();
              // TODO - Add Service implemention here
            }
            catch (Exception ex) {
                // TODO - Process error if needed then re-throw
                throw ex;
            }
            finally {
                clientProxy.Close();
            }
            return reply;
         }
```

Attributes

In WCF, a service is defined as a group of operations known as a service contract. You define an operation by creating a method and annotating it with the `OperationContract` attribute. To create a service contract, you group together the operations by declaring them within an interface annotated with the `ServiceContract` attribute.

Once you have designed and implemented the service contract, you can configure the execution behavior of the service runtime. To configure the behavior, you mark service classes with `ServiceBehavior` attributes and service methods with `OperationBehavior` attributes. The following sections list supported and unsupported attributes and attribute properties.

Supported Attributes

- DataContractFormatAttribute
- FaultContractAttribute
- ServiceKnownTypeAttribute

- OperationContractAttribute

Supported Properties	Unsupported Properties
Action	AsyncPattern
IsOneWay	HasProtectionLevel
Name	IsInitiating
ReplyAction	IsTerminating
TypeId	Protection Level

- ServiceBehaviorsAttribute

Supported Properties	Unsupported Properties
ConfigurationName	AutomaticSessionShutdown
IncludeExceptionDetailInFaults	ConcurrencyMode
Name	MaxItemsInObjectGraph
Namespace	UseSynchronizationContext
TypeId	ValidateMustUnderstand
	InstanceContextMode (WCF hosted services behave like sessionless call. If PerSession or Singleton is set, the service will behave as PerCall.)

- ServiceContractAttribute

Supported Properties	Unsupported Properties
ConfigurationName	CallbackContract
Name	HasProtectionLevel
Namespace	ProtectionLevel
TypeId	SessionMode (can only use default value of Allowed)
	ValidateMustUnderstand

Unsupported Attributes

- CallBackBehaviorAttribute

- DeliveryRequirementsAttribute
- MessageBodyMemberAttribute
- MessageContractAttribute
- MessageContractMemberAttribute
- MessageHeaderArrayAttribute
- MessageHeaderAttribute
- MessageParameterAttribute
- MessagePropertyAttribute
- OperationBehaviorAttribute
- PeerHopCountAttribute
- TransactionFlowAttribute
- XMLSerializerFormatAttribute

Invoking ActiveMatrix Services from WCF Host Component Implementations

To invoke a service from a WCF Host component, you generate a proxy class (See WCF documentation for information on how to generate a proxy class) and invoke the proxy class in the WCF Host component implementation.

To enable the WCF Host component to reference an ActiveMatrix service:

1. Create the proxy instance using the prototype that has an `endpointConfigurationName` as the only parameter.
2. Configure the client reference in the component's application config file. See Mapping Service Contracts to Port Types on page 30.

Logging From WCF Host Component Implementations

This section describes the logging features available to WCF Host component implementations.

Standard Output and Error Logging

The standard output and error streams from WCF Host component implementations are directed to the Management Daemon console.

TraceSource Logging

Within ActiveMatrix, a global, static, preconfigured `TraceSource` named `Funcs.ts` is provided for WCF Host component implementations. You can directly use this `TraceSource` in your implementation:

```
Funcs.ts.TraceEvent(TraceEventType.Warning, 8820,
"ValidationCallBack: Matching schema not found.  No validation
occurred. " + args.Message);
```

The size of the message supplied to a `TraceSource` method should be less than 2K. Messages that are longer are truncated.

Log Message Destinations

To control where log messages are directed, you set the logging configuration for the WCF Host container or the WCF Host service unit that contains the component implementation. For information on logging configurations, see Working with Logging Configurations in *TIBCO ActiveMatrix Administration*.

- If you configure file-based logging for the WCF Host container, the log statements are sent to the specified log(s), depending on where you have specified the log roll over.
- If you configure file-based logging for WCF Host service units, the log statements from the Java portion of the Service Assembly (SA) or Service Unit (SU) framework are sent to the specified log(s), depending on where you have specified the log roll over.

Since the ActiveMatrix WCF Host container consists of Java, C++, and C# code, a separate log file is created for each AppDomain in the hosted CLR. Each WCF Host component is assigned its own AppDomain. There is one log file for the default AppDomain (`CLRHost_DefaultDomain.log`) and one log for each AppDomain hosting a component implementation. The `AppDomain` log files contain log statements from both C# portion of the container and from the application code that uses the `Funcs.ts` log object.

To use the `Funcs.ts` object, the application needs to reference the `AMXServicesSupport.dll` provided by TIBCO.

Since the AppDomain log files use the standard Microsoft text TraceListener, the log statements are not in CBEF format, so cannot be imported into TIBCO Business Studio.

Service Unit Log File Names

The name of the AppDomain log files are constructed as follows:

1. Begin with the file name entered in the logging configuration in ActiveMatrix Administrator. Remove the trailing suffix `.log` if it exists. For a file named `MyService.log` the component is:

```
MyService
```

2. Append an '_' underscore.

```
MyService_
```

3. Append the AppDomain.CurrentDomain.FriendlyName component. The procedure for constructing this component is described in AppDomain Name on page 1.
   - — If the name of the composite is `hw`, the namespace of the composite is `http://www.example.com/hw/` (this can be changed in the composite's property page in the Composite Editor, if needed) and
   - — If the componentname is `HelloWorld`, the AppDomain.CurrentDomain.FriendlyName is `www_example_com_hw__HelloWorld`.

   The name appears as:

`MyService_www_example_com_hw__HelloWorld`

4. Append `.log`.

`MyService_www_example_com_hw__HelloWorld.log`

AppDomain Name

Each AppDomain must have a unique name. TIBCO uses an algorithm to generate this unique AppDomain name. The algorithm concatenates the following strings, each separated by an underscore:

- Normalized composite namespace after removing the leading "http://". To normalize means replacing reserved file characters with underscores. The reserved characters include periods and slashes.
- Component Name.

For example:

targetNamespace="`http://www.example.com/hw/`" and component name = `HelloWorldComp` yields the AppDomain Name: `www_example_com_hw__HelloWorldComp`

Developing WCF Host Components

This section discusses the following aspects of developing WCF Host components:

Generating WSDL Files From WCF Services

A WCF service does not require a WSDL to function, but integration within ActiveMatrix requires that you define the service contract in WSDL format. WCF services support the generation of the contract definition in WSDL form using the Microsoft `svcutil` utility:

```
C:\Program Files\Microsoft SDKs\Windows\v6.0A\bin\svcutil.exe
```

You can generate WSDL files in two ways:

- Through an export mechanism supported by the utility. See Generating WSDL Files From a WCF Assembly on page 25.
- Through calls to a running WCF service. See Generating WSDL Files From a Running Service on page 26.

Generating WSDL Files From a WCF Assembly

To generate WDSL files from a WCF assembly from a command prompt, run:

`svcutil /t:metadata` *service implementation assembly*

For example:

```
svcutil /t:metadata
C:\samples\HelloWorld\HelloWorldSvc\bin\HelloWorldSvc.dll
```

The command generates three (or more) files:

- `tempuri.org.wsdl` An abstract WSDL file containing the port type, operation, and message definitions
- `tempuri.org.xsd` Types imported by the WSDL file

- `schemas.microsoft.com.2003.10.Serialization.xsd` Standard Microsoft types

Since there is no clear association between these file names and the service they originated from, if you have more than one WCF Host component in the composite, there could be name conflicts when you import the files into TIBCO Business Studio.

You should change the file names to reflect the name of the service that they represent.

Generating WSDL Files From a Running Service

To generate WSDL files from a running service from a command prompt, run

`svcutil /t:metadata` *service URL*`?wsdl`

For example:

```
svcutil /t:metadata http://localhost:8082/ihelloworld?wsdl
```

The command generates three (or more) files:

- `tempuri.org.wsdl` An abstract WSDL file containing the port type, operation, and message definitions
- `tempuri.org.xsd` Types imported by the WSDL file
- `schemas.microsoft.com.2003.10.Serialization.xsd` Standard Microsoft types

You can also generate the concrete WDSL file from a running service using a browser. When you supply the service URL followed by `?wsdl`, a concrete WSDL is displayed in the browser, which can be saved to a file.

Importing WSDL and Schema Files, Implementation Assemblies and Configuration Files

1. Import the WSDL and schema files into the ActiveMatrix SOA project's Service Descriptors folder.

   The WSDL file that the component implements must either be collocated with the implementation assembly or must be in the project's Service Descriptors folder. It cannot be in a subfolder of that folder. All WSDL files must be in the same location; they cannot be split between two locations.

   The WSDL files display error badges due to unresolved schema references. See Resolving XSD Schema References on page 27 to know how to resolve the errors.

2. Import the service implementation assemblies and config files into the SOA project folder.

   The implementation assembly (*xyz*`.dll`) and its configuration file (*xyz*`.dll.config`) must be in the same folder location. The folder name cannot contain spaces.

   The config files display error badges due to unresolved service contract references. You resolve the errors in Mapping Service Contracts to Port Types on page 30.

Resolving XSD Schema References

WSDL files generated with `svcutil` have `import` statements for importing XSD file namespaces that elements use in the WSDL. However, the schema files are not explicitly named using the `schemaLocation` attribute if WSDL is generated from a WCF assembly. For example:

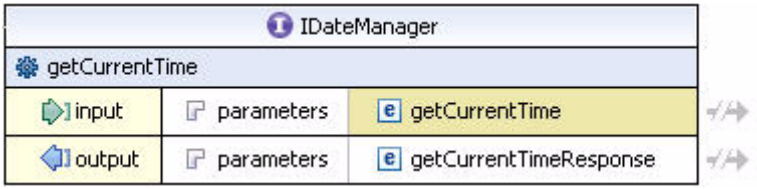
```
<xsd:schema targetNamespace="http://tempuri.org/Imports">
  <xsd:import namespace="http://tempuri.org/"/>
  <xsd:import
    namespace="http://schemas.microsoft.com/2003/10/Serialization/
"/>
</xsd:schema>
```

The `schemaLocation` attribute refers to a `url` if WSDL is generated from a running service. For example:

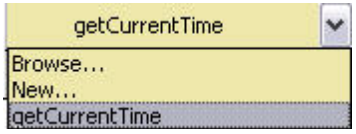
```
<xsd:schema targetNamespace="http://tempuri.org/Imports">
  <xsd:import schemaLocation="http://localhost:8089/?xsd=xsd0"
namespace="http://tempuri.org/" />
  <xsd:import schemaLocation="http://localhost:8089/?xsd=xsd1"
namespace="http://schemas.microsoft.com/2003/10/Serialization/" />
  <xsd:import schemaLocation="http://localhost:8089/?xsd=xsd2"
namespace="http://schemas.datacontract.org/2004/07/AddressProxy"
/>
</xsd:schema>
```

TIBCO Business Studio requires that the imported namespaces identify the XSD file explicitly using the `schemaLocation` attribute. To resolve the schema references:

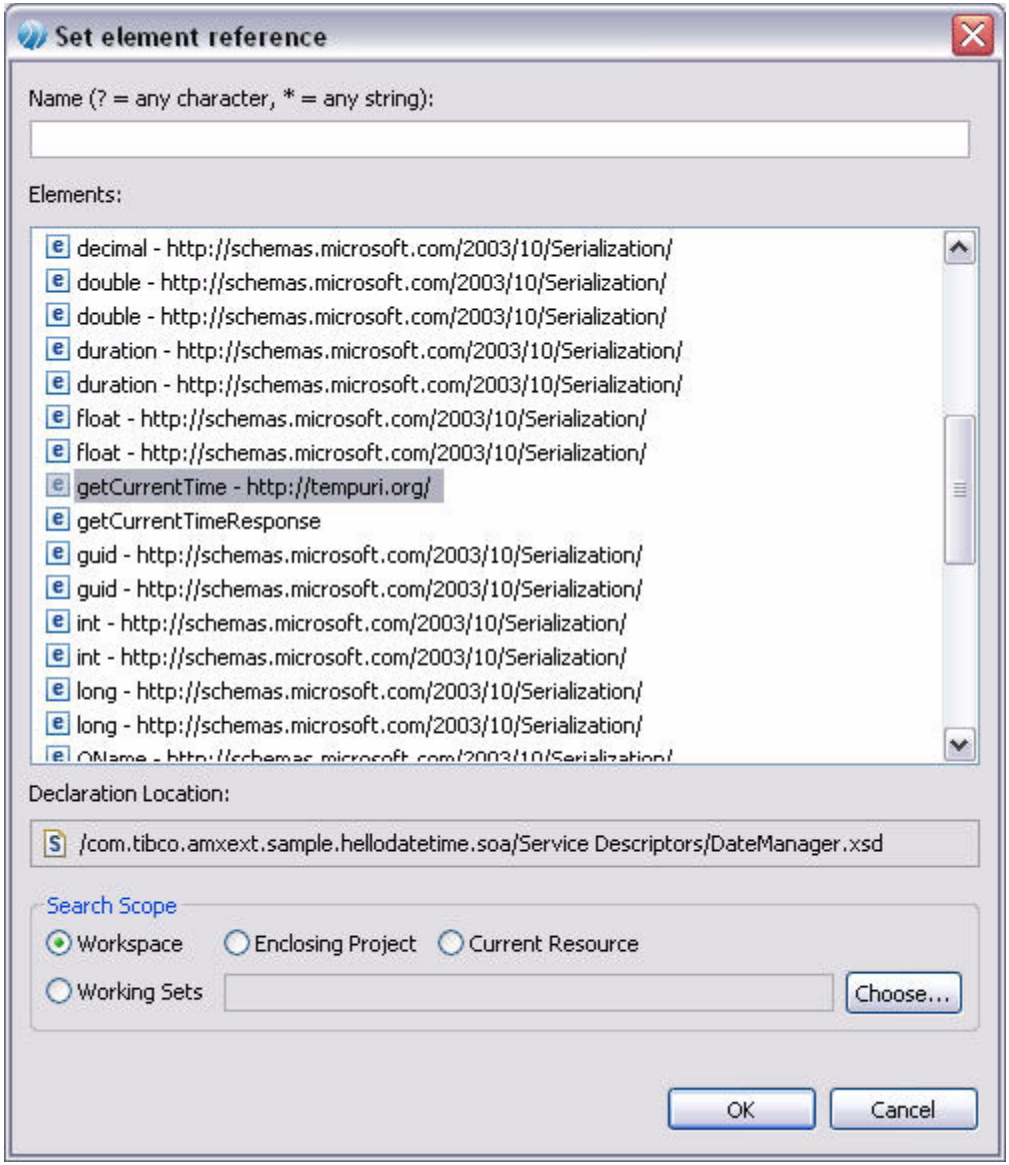
1. Right-click the WSDL file and select **Open With > WSDL Editor**. The port type graphic identifies unresolved messages with a broken reference icon . For example:



2. Right-click a message with a broken reference icon. A drop-down list displays.



3. Click **Browse...**. An element browser displays.
   a. In the Search Scope area, click the **Workspace** radio button.
   b. In the **Name(?=any character, *=any string):** enter the element name and select the correct element according to the schema file displayed in the **Declaration Location** field.



c. Click **OK** and then click the **Save** button . The remaining message references should be resolved. For any outstanding unresolved references, repeat steps 2. and 3.

4. Right-click the WSDL file and select **Open With > Text Editor**.
5. There will be multiple schema elements. Delete the schema elements with the `import` statements without `schemaLocation` attributes or, with `schemaLocation` attributes referring to a url.
6. Save and close the file.

Remove WSDL Endpoint Reference

WSDL files generated from a running service with wsHttpBindings may have an endpoint reference similar to that shown here included in the bindings. If so, TIBCO Business Studio flags an error (duplicate defined endpoint). The error is cleared by removing these configuration settings in a text editor:

```
<wsa10:EndpointReference>
<wsa10:Address>http://localhost:8088/Calculator1</wsa10:Address>
<Identity
xmlns="http://schemas.xmlsoap.org/ws/2006/02/addressingidentity">
<Upn>lnutsch@na.tibco.com</Upn>
</Identity>
</wsa10:EndpointReference>
```

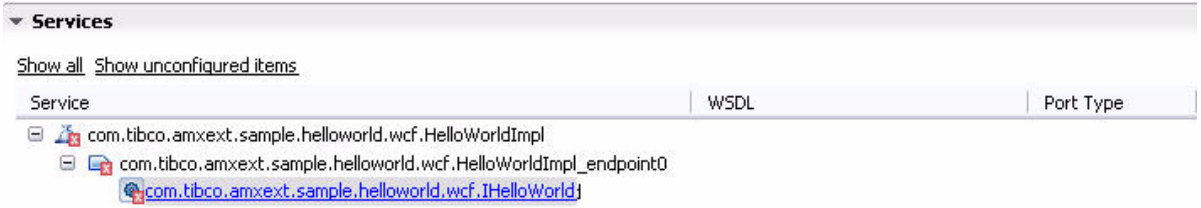
Mapping Service Contracts to Port Types

WCF service contracts are developed without referencing WSDL artifacts. However, ActiveMatrix requires information on the mapping between port types and WCF service contracts. To map the WCF service contracts defined in config files to port types:

1. Right-click the config file and select **Open With > WCF Config File Editor**. The editor opens to the Service tab.

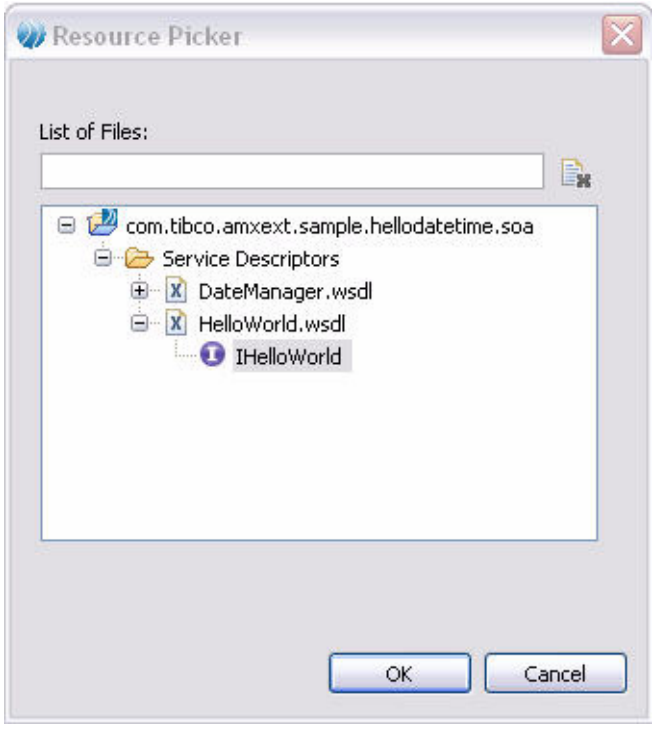
Service contracts

2. The Service tab displays unresolved service contracts.



3. Click the contract. A resource picker displays.

   a. Expand the list of files and select the appropriate port type from the resource picker.



   b. Click **OK**. The contract is mapped to a WSDL file and port type and the error badge disappears.

4. Click the **Save** button .

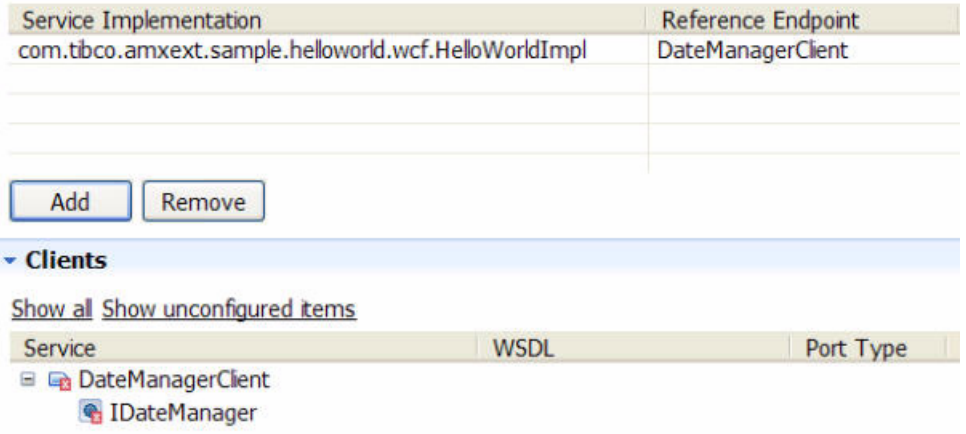
Reference endpoint configuration

5. If the service implementation references endpoints, click the **Client** tab at the bottom of the editor. The Client tab contains a **Reference Endpoint Configuration** table listing the service implementation and the endpoints that it references. In order to add the references to the WCF Host component, you manually add items to the table.

   a. For each endpoint that the service implementation references, click the **Add** button. A row is added to the Service Implementation column.

      In the Reference Endpoint column, choose an endpoint that component references from the drop-down list in the Endpoint column. Each endpoint that you choose is added as a component reference. An endpoint is added

to the Clients area under the table. The endpoint displays an error badge because the service contract it exports is not mapped to a port type.



b. For each endpoint in the Clients area, click the unmapped service contract and map to a port type following the process in step 2.

6. Click the **Save** button .

Creating a WCF Host Component

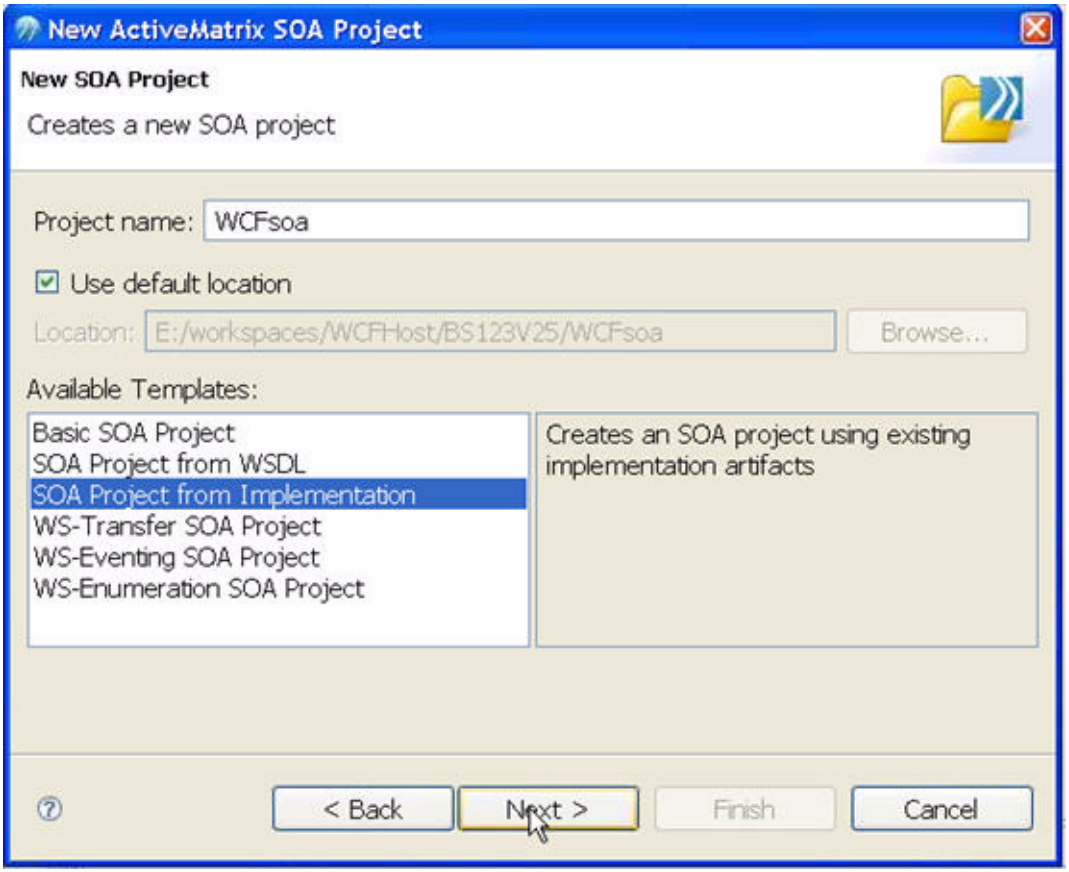
To create a WCF Host component:

1. Start TIBCO Business Studio.
2. Create an ActiveMatrix SOA project as follows. Alternately, you can use the wizard to create a component as described in the following section, SOA Project from Implementation Using the Wizard on page 33.
   a. When you select a project template, choose the Basic SOA Project.
   b. Open the project's composite.
   c. Click the canvas and click the WCF Host component icon in the popup toolbar (see Popup Toolbars in *TIBCO ActiveMatrix Composite Editor User's Guide*) or click WCF Host in the Components group in the Palette and click the canvas. A WCF Host component is added to the Components area.
   d. Type a name in the component's activated input field. The name cannot contain spaces.
3. Click the **Save** button .

SOA Project from Implementation Using the Wizard

The SOA Project from Implementation project wizard creates a project, composite, services, and component whose port types are defined by an imported implementation. Before the wizard is used, the service contracts of the imported implementation should be mapped to the port types as specified in the following section, Mapping Service Contracts to Port Types on page 30. To create an SOA project from an existing implementation:

1. Select **File > New > ActiveMatrix Resources... > ActiveMatrix SOA Project**.
2. Click **Next**. The New ActiveMatrix SOA Project screen displays:



3. Type a name in the Project name field.

4. Select a project template **SOA Project from Implementation** from the Available Templates list and click **Next**.

Specify component

5. In the **Component Name** field, accept the default name or type a new name and in the **Component Implementation** field, select `WCF Host` and click **Next**.
6. In the **Implementation DLL/EXE** field, click the **Workspace...** button to use an implementation that has already been imported into the workspace. The Implementation DLL field is filled in.
7. In the Class field, select the class that provides the component implementation and click **Next**.
8. In the Available Port Types list, check the checkboxes next to the port types you want to expose and click **Next**.
9. For each port type, check the checkboxes next to the service types you want the composite to expose. For each service, click the Service Name column and rename the service as desired and click **Next**.
10. To accept the default folders and folder names for the project, click **Finish**. Otherwise, click **Next** and proceed with the remaining steps.

Specify asset types

11. Uncheck the checkboxes next to the asset types you don't want to create in the project.
12. Click **Next** to step through the asset types and configure the names of special folders containing the assets.
13. Click **Finish.**

    When you complete the task, Business Studio creates a project containing the following resources:

    — A composite containing one or more services and a component.

    — An HTTP shared resource profile named ResourceProfile0.

    It also:

    — Adds the WSDL file to the Service Descriptors folder.

    — Creates an HTTP shared resource named HTTP_Server in the Shared Resources folder.

    — Creates a substitution variables file in the Shared Resources folder.

Configuring a WCF Host Component's Implementation

To configure a WCF Host component implementation:

1. In the Properties view, click the **Implementation** tab.
2. Click the **Browse** button to the right of the **Implementation Class** field.
3. Choose the class from the SOA project
4. Click **OK**. The Class and Location field are filled in. The services and references you configured in Mapping Service Contracts to Port Types on page 30 are automatically added to the component.
5. Click the **Save** button  .

Updating a WCF Host Component's Implementation

To update the component after you change the implementation or config files of a WCF Host component:

1. Reimport updated implementation files into the project following the procedure described in Importing WSDL and Schema Files, Implementation Assemblies and Configuration Files on page 27.
2. Click the component.
3. In the Properties view, click the **Implementation** tab.
4. If the reimported files have the same names as the old ones, click **Load Implementation** icon  at the top right of the tabbed notebook heading. Any changes to the component's service or references are reflected in the composite and property views.
5. Click the **Save** button  .

Index

Symbols

C

D

I

M

P

R

S

T

U

V

W