Application Synchronization
AS Adapter Interface Partner Guide

Hologic Confidential Information

MAN-11520 Revision 001

Technical Support
USA: +1.800.760.8342
Europe: +32.2.711.4690
Asia: +852 37.48.77.00
All other: +1.781.999.7750
Email: **imgsupport@hologic.com**

HOLOGIC®

Hologic Inc.
600 Technology Drive
Newark, DE 19702
USA
Tel: +1.800.447.1856

EC REP **Hologic BV**
(EU Representative)
Da Vincilaan 5
1930 Zaventem
Belgium
Tel: +32 2 711 46 80
Fax: +32 2 725 20 87

For more information about Hologic products and services, visit **www.hologic.com**.

CE 2797

Contents

1.0. Introduction

Hologic, Inc. develops and markets a full line of digital mammography products including the Selenia® and Selenia Dimensions® digital mammography systems, the SecurView®, MultiView™ workstation product family, and the SecurXchange™ archive/router product family.

The Hologic Application Synchronization (AS) option provides an interface to allow for run-time user and patient context synchronization between a Hologic workstation application and independent third-party applications (for example, RIS, dictation, and reporting applications).

Application Synchronization has the ability to communicate with each application in its preferred communication method (such as TCP/IP, serial port, SharedFile, etc.) and knows which messages each application supports.

Application Synchronization operates such that synchronization with local applications (for example, thin client of third-party application running on the Hologic workstation) or applications running on a separate computer are handled transparently.

The AS Adapter is a component that wraps the complex functionality of Hologic Application Synchronization.

1.1. Audience

This guide is intended to assist developers of third-party applications (TPAs) to understand the AS Adapter interface, and to communicate detailed information on how to implement the interface for a Hologic partner's TPA (for example, RIS, dictation, reporting application).

1.2. System Overview

A Hologic workstation and a TPA, also referred to as a connected application (CA) may communicate on the same computer or over a network.

The AS Adapter does not communicate directly with a Hologic workstation, but rather uses Application Synchronization (AS). This was done intentionally to provide multi-tier communication and synchronization between Hologic workstations and multiple connected applications.

The primary channel for communication with a Hologic workstation is TCP/IP. To enable synchronization in a network environment, the communication between the AS Adapter and AS also uses TCP/IP. However, the low-level communication is abstracted and is completely transparent for connected application developers using the AS Adapter. Only the high-level functionality is exposed.

1.3. System Requirements

Operating System: The AS Adapter is verified on Windows 7.

Firewall: Since inter-process communication between the AS Adapter and AS uses TCP/IP, any implemented firewall must be set to enable communication on the configured ports. For details on how to configure the AS Adapter's TCP/IP parameters, see section **9.2.**

Microsoft .NET Framework: Any computer on which AS Adapter is used must have Microsoft .NET Framework 2.0 or higher installed.

Hologic has used great care in developing the Application Synchronization software, encryption software, software testing tools, partner guide, AS adapter, and related documentation. However, as stated in the SecurView Integration Agreement between Hologic and your company, Hologic makes no warranty that you will produce an effective interface to Hologic workstations using these tools, and Hologic emphasizes that it is your responsibility to test the integration solution that you produce in order to ensure that it is adequate for your purposes.

2.0. Definitions, Terms, and Abbreviations

Application Synchronization (AS): Windows service that acts as a message translator between Hologic workstations and third-party applications.

AS Adapter: A Microsoft .NET assembly (DLL) that a third-party application will use to communicate with AS.

CLR: Microsoft .NET Common Language Runtime

COM: Component Object Model, reusable software component based on technology in Microsoft Windows that enables software components to communicate.

COM Interop: Microsoft .NET Framework bridge between .NET (managed code) and COM (unmanaged code).

Connected Application (CA): A third-party's application such as RIS, dictation, or reporting, that uses AS to communicate with Hologic workstations.

Managed Code: Code executing under the control of the Microsoft .NET CLR

Message: Basic communication element exchanged between connected applications.

Request: A message sent from the requester to the responder.

Requester: Connected application that sends a request to Application Synchronization and optionally awaits a response to be returned.

Responder: Connected application that receives a request from Application Synchronization and optionally returns a response.

Response: A message sent as a reply from a responder to a requester through Application Synchronization. The response references the corresponding request.

TCP/IP: Transmission Control Protocol/Internet Protocol, a communications protocol for computer networks

Unmanaged Code: Code that runs outside the CLR, such as COM components, ActiveX interfaces, and Win32 API functions.

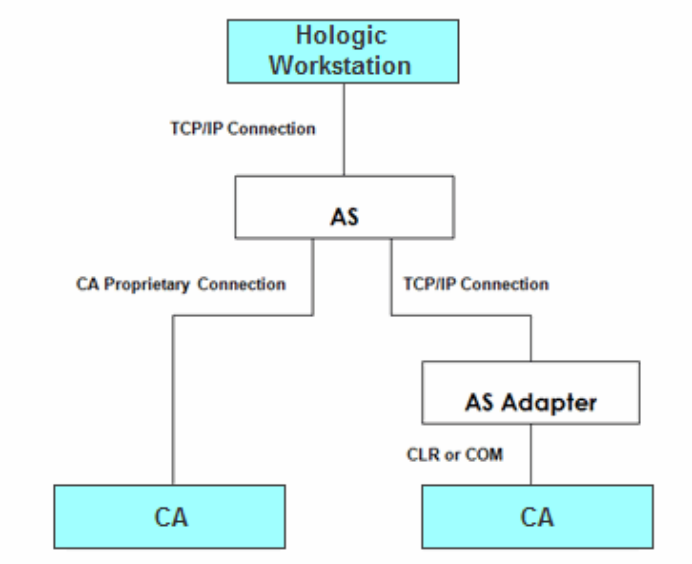
VM: Value Multiplicity. The number of values permitted in a data element property.

3.0. Synchronization Protocol

The following statements describe the interactions that Application Synchronization facilitates:

- A Hologic workstation or any connected application can initiate communication.
- A Hologic workstation and connected application can send requests to one another. A response may be returned.
- A connected application may use the AS Adapter to communicate with a Hologic workstation. All low-level communication tasks and message translations are completely hidden from the connected application, and are provided by the AS Adapter.
- The AS Adapter has to be configured in order to communicate with AS.
- The configuration file is an XML document.

The following diagram shows the system configuration where several connected applications communicate with each other.



4.0. AS Adapter

The purpose of the AS Adapter is to hide the entire low-level communication and XML message translation logic, and expose only the high-level interfaces for the connected application to send and receive requests with Application Synchronization. Rapid application development is facilitated by allowing a developer of a connected application to populate only relevant user, patient or study information when making a synchronization request, or to extract the same information from incoming synchronization request events.

The AS Adapter is a Microsoft .NET assembly (DLL), and will be running in the process space of the connected application. However, the AS Adapter can communicate and synchronize data with AS located on the same computer, or a different computer across a network.

m ***Note:*** *The AS Adapter interface happens to use the SecurView name because SecurView was the first Hologic workstation to support Application Synchronization.*

Sending request means calling the appropriate function of the AsAdapter class (or ISecurView interface for unmanaged COM connected applications).

Receiving request means handling incoming AsAdapter class events (or ISecurViewEvents for unmanaged COM connected applications).

The managed connected application may convey the User, Patient or Study data to the AsAdapter class by calling its methods, and receive these data using AsAdapter class events.

The unmanaged COM connected application may convey the IUser, IPatient or IStudy helper interfaces to the `ISecurView` interface and receive them from `ISecurViewEvents`.

A Hologic workstation and a connected application can synchronize: 1) the user through a single login and logout functionality, 2) the currently displayed patient, and 3) the patient's status.

While we encourage every partner to implement all the synchronization requests as both sender and receiver, we realize this may not be feasible. Through a configuration of AS, it is possible to indicate individually whether or not a connected application is capable of sending each type of synchronization request, and whether or not a connected application is capable of receiving each type of synchronization request.

4.1. User Synchronization

A Hologic workstation and a connected application can synchronize the user context before they exchange patient synchronization messages. The supported methods and events are:

- AS Adapter triggers an ***OnLogin*** event when a user logs in to the Hologic workstation.
- AS Adapter triggers an ***OnLogout*** event when a user logs out of the Hologic workstation.
- CA invokes the ***Login*** method when a user logs in to the CA.
- CA invokes the ***Logout*** method when a user logs out of the CA.

For details, see ***Login***, ***Logout*** methods, ***OnLogin***, ***OnLogout*** events, in sections **11.1.2**, **11.1.4** and **11.2.1** for managed, or sections **12.1.2, 12.2** and **12.3.1** for unmanaged connected applications.

4.2. Patient Synchronization

A Hologic workstation and a connected application can synchronize the displayed patient. The supported methods and events are:

- AS Adapter triggers an ***OnOpenPatient*** event when the displayed patient is changed on the Hologic workstation.
- CA invokes the ***OpenPatient*** method when the active or current patient record is changed on the CA.

  m ***Note:*** *A CA can use special values in the patient information parameter of the* ***OpenPatient*** *method to trigger the SecurView or MultiView viewer to close if a patient is currently open for review.*

For details, see ***OpenPatient*** method, ***OnOpenPatient*** event, in sections **11.1.2**, **11.1.4**, and **11.2.2** for managed, or sections **12.1.2**, **12.2**, and **12.3.2** for unmanaged connected applications.

A Hologic workstation and a connected application can synchronize the status of a patient. SecurView supports sending and receiving a patient status update. MultiView supports receiving a patient status update.

The supported methods and events are:

- AS Adapter triggers an ***OnUpdatePatientState*** event when a patient/study is marked as read on the SecurView workstation.

- CA invokes the ***UpdatePatientState*** method when a patient/study is marked as read on the CA.

For details, see ***UpdatePatientState*** method, ***OnUpdatePatientState***, in sections **11.1.2**, **11.1.4**, and **11.2.2** for managed, or sections **12.1.2**, **12.2**, and **12.3.2**, for unmanaged connected applications.

5.0. Managed and Unmanaged Connected Applications

The AS Adapter consists of a set of managed classes and interfaces.

For connected applications that use a managed code environment, all public classes and interfaces exposed by the AS Adapter and the MessageObjects may be used directly from the managed code (AsAdapter class).

Connected applications that use an unmanaged code environment must use the COM Interop interfaces (ISecurView) exposed from the managed classes (AsAdapter).

For more information about using AS Adapter through the .NET COM Interop, please see Appendix **B.1. Registering .NET Assemblies** and Appendix **B.2. Deploying the Required Files.**

The AS Adapter is working in a managed environment, so any computer on which it is used must have the .NET Framework 2.0 or higher installed.

6.0. Request/Response Mechanism

Application Synchronization uses requests and responses for communication.

Through a configuration of AS, it is possible to indicate whether or not a connected application is capable of responding and/or capable of receiving a response. If the Response mechanism is implemented in either direction, it must be implemented fully, meaning pending responses must be handled.

The requester sends a request to the responder, and then optionally waits for a response. If sending a response is supported, the responder sends a response (according to the figure in section **6.2. Pending Request/Response Mechanism**) after it has finished performing the request.

If AS is expecting a response and the request is not answered within the timeout period specified in AS configuration to wait for a response from the connected application that received the request, AS will send an appropriate response to the requester.

The Request/Response Mechanism figure in section **6.2** shows the standard mechanism described above, plus the additional pending mechanism, where the responder can send a pending message to indicate that it will handle the message but needs more time to complete the task, (for example, waiting for user input).

AS communicates all incoming requests to the connected application by triggering synchronous events. If the connected application supports sending a response, it may do so only in a subsequent Response method call.

If sending responses is supported, a connected application must respond to each request it receives, to indicate if the request was successfully or unsuccessfully handled.

If receiving responses from AS is supported, a connected application should handle the response event that corresponds to each request that the connected application sends to AS.

There are two mechanisms that a connected application can use to send or receive a response:

6.1. Standard Request/Response Mechanism

Upon receiving a request (handling the request event), the connected application must call the `AsAdapter.Response (ISecurView.Response)` method once to acknowledge that the request was successfully or unsuccessfully handled.

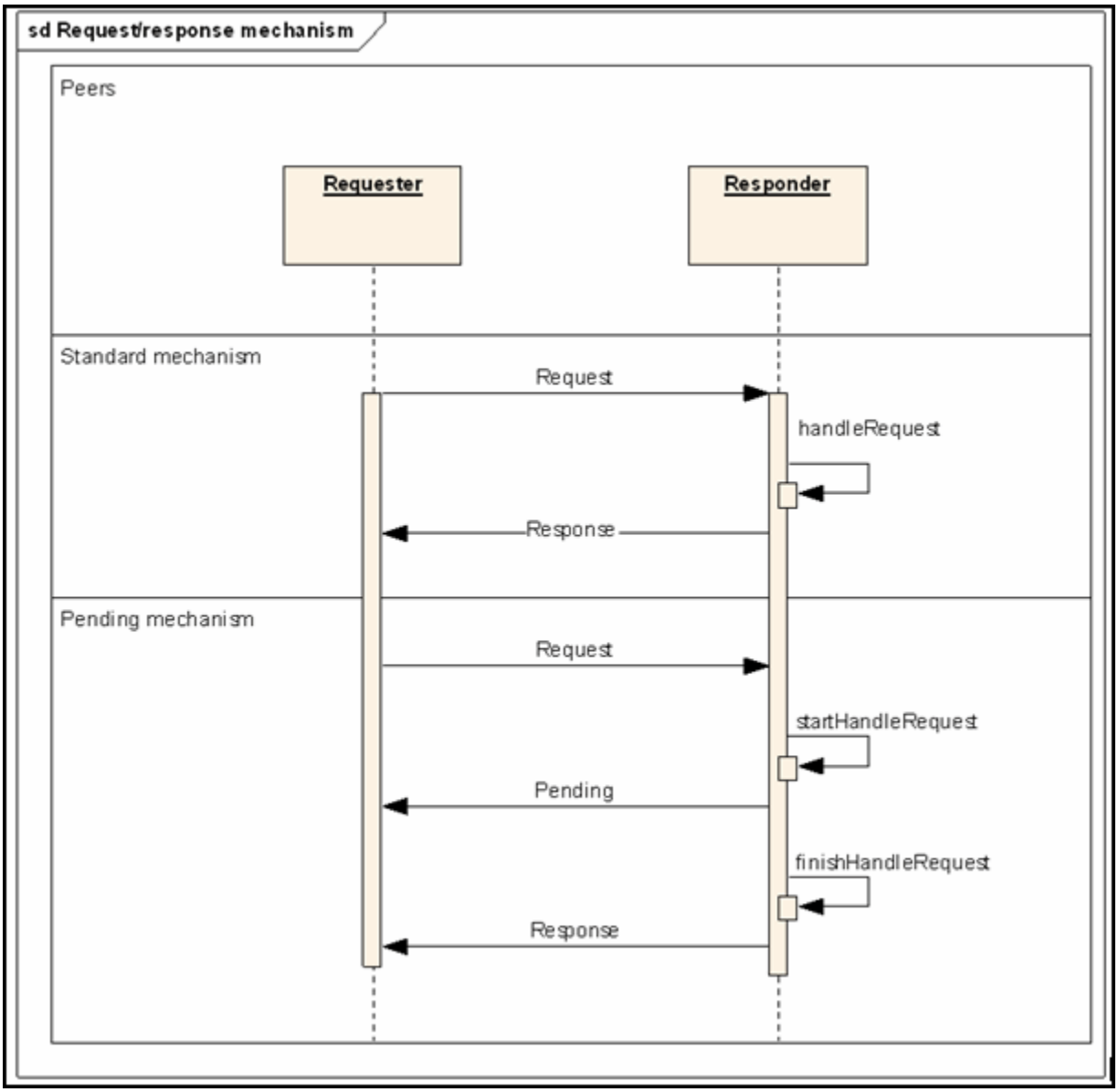
Each request is identified with a unique request ID. When calling the `Response` method, the connected application must pass the corresponding unique request ID, along with success, failed, or not supported status info.

After the connected application sends a request, AS will return a response that the connected application should handle appropriately.

6.2. Pending Request/Response Mechanism

If the connected application needs time to fulfil a request, it must send one or more pending responses first, followed by the response that indicates whether the request was successfully or unsuccessfully handled. Both calls to the `Response` method must pass the same corresponding unique request ID.

After sending a request, the connected application should be prepared to handle one or more pending response events that may arrive before a response event.



Request/Response Mechanisms

7.0. Connected Application Sending and Receiving Requests

7.1. Sending Requests

The AS Adapter provides functionality to send requests initiated by a connected application to Hologic workstations and other connected applications using AS.

Invoking methods of the AsAdapter class (or `ISecurView` interface) sends a request to AS.

The request is by its nature asynchronous, however the invoked function will return synchronously in order not to block the calling thread, therefore simplifying the development effort to integrate the AS Adapter.

If a connected application is configured in AS to accept responses, the response event `OnResponse` will be triggered from the AsAdapter class (or `ISecurViewEvents` interface) for each request issued by the connected application, and the connected application may take an appropriate action.

7.2. Receiving Requests

The AS Adapter also provides functionality to receive requests initiated by Hologic workstations and other connected applications using AS. Depending on the type of incoming request, the AS Adapter will fire the corresponding event defined in the AsAdapter class (or `ISecurView` Events source interface for COM CAs).

The event will be triggered only if:

- AS is configured to dispatch the request from a Hologic workstation or other connected application to this connected application. Note: AS has a configuration user interface that manages the content of the configuration file Devices.hlx.
- The connected application subscribes to receive an event from the AsAdapter class.

7.3. Managed Connected Application Requests

For more information about the request methods and events please see the corresponding references: **11.1.2** and **11.1.4**.

7.3.1. AsAdapter Request (CA sends request)

The AsAdapter class supports the following request methods:

```
namespace Hologic.SecurView.AppSync.Adapters
```

Methods with User and Patient parameter list:

```
string Login(User user);
string Logout(User user);
string OpenPatient(User user, Patient patient);
string UpdatePatientState(User user, Patient patient);
```

The return value for all request methods is the unique request ID (requestUID). It is used to match the corresponding response that will come with the `OnResponse` event.

7.3.2. AsAdapter Events (CA receives request)

The AsAdapter class provides the following request events:

```
namespace Hologic.SecurView.AppSync.Adapters
```

Events with User and Patient:

```
void OnLogin(User user, string requestUID);
void OnLogout(User user, string requestUID);
void OnOpenPatient(User user, Patient patient, string requestUID);
void OnUpdatePatientState(User user, Patient patient, string
requestUID);
void OnError(Exception exception);
```

7.4. Unmanaged Connected Application Requests

7.4.1. ISecurView Interface Request (CA sends request)

The ISecurView interface supports the following request methods:

```
namespace Hologic.SecurView.AppSync.Adapters
```

Methods with IUser and IPatient parameter list:

```
string Login(IUser user);
string Logout(IUser user);
string OpenPatient(IUser user, IPatient patient);
string UpdatePatientState(IUser user, IPatient patient);
```

The return value for all request methods is the unique request ID (requestUID). It is used to match the corresponding response that will come with the `OnResponse` event.

7.4.2. ISecurViewEvents Events (CA receives request)

The ISecurViewEvents interface is a source interface provided only for unmanaged COM connected applications. The unmanaged connected applications must implement the ISecurViewEvents sink interface.

It provides the following request events:

```
namespace Hologic.SecurView.AppSync.Adapters
```

Events with `IUser` and `IPatient`:

```
void OnLogin(IUser user, string requestUID);
void OnLogout(IUser user, string requestUID);
void OnOpenPatient(IUser user, IPatient patient, string requestUID);
void OnUpdatePatientState(IUser user, IPatient patient, string
requestUID);
void OnError(Exception exception);
```

8.0. Connected Application Response

8.1. Managed Connected Application Response

8.1.1. AsAdapter class Response to Request received by CA

If a connected application implements sending a response to a received request, the connected application must call this method of the AsAdapter class to indicate that the request was successfully or unsuccessfully handled.

The AsAdapter class provides the following response methods:

```
namespace Hologic.SecurView.AppSync.Adapters
```

```
void Response(string requestUID, ResponseStatus status, string
reason);
```

8.1.2. Response Event for Request sent by CA

A managed connected application will receive a response event for each request sent by the connected application using the AsAdapter class. Each response is matched to a particular request using the unique requestUID.

If a connected application implements receiving responses to sent requests, the `OnResponse` event will be triggered from the AsAdapter class, and the connected application must handle it.

The AsAdapter class provides the following response event:

```
namespace Hologic.SecurView.AppSync.Adapters
```

```
void OnResponse(string requestUID, ResponseStatus status, string
reason);
```

8.2. Unmanaged Connected Application Response

8.2.1. ISecurView interface Response to Request received by CA

If a connected application implements sending a response to a received request, the connected application must call this method of the ISecurView interface to indicate that the request was successfully or unsuccessfully handled.

The ISecurView interface provides the following response methods:

```
namespace Hologic.SecurView.AppSync.Adapters
```

```
void Response(string requestUID, ResponseStatus status, string
reason);
```

8.2.2. ISecurViewEvent Response Event for Request sent by CA

An unmanaged connected application will receive a response event for each request sent by the connected application using the ISecurViewEvents source interface. Each response is matched to a particular request using the unique requestUID.

If a connected application implements receiving responses to sent requests, the `OnResponse` event will be triggered from the ISecurViewEvents interface, and the connected application must handle it.

The ISecurViewEvents interface provides the following response event:

```
namespace Hologic.SecurView.AppSync.Adapters
```

```
void OnResponse(string requestUID, ResponseStatus status, string
reason);
```

9.0. Configuration

The configuration of the AS Adapter resides in an XML document. The configuration utilizes the standard .NET configuration system, where the configuration file has the same name as the assembly suffixed by '.config'. The file name is `Hologic.SecurView.AppSync.Adapters.dll.config`.

This file must reside in the same directory as the executable and the `Hologic.SecurView.AppSync.Adapters.dll` assembly.

This AS Adapter configuration file contains a subset of two XML nodes (`<APPLINKER_PROPERTY_DEFINITION> and <DEVICE>`) of the AS configuration file Devices.hlx, relevant to AS Adapter's Connected Application.

Devices.hlx is the configuration file for AS. It is located in the C:₩ProgramData₩Hologic₩AppSync₩Configuration directory. Please do not edit this file, as it may affect the operation of AS.

Always configure AS first using the user interface, then the AS Adapter's configuration file.

There are two ways to configure the AS Adapter's configuration file, and both require manual editing.

- Update all node values and the DEVICE name attribute in the `Hologic.SecurView.AppSync.Adapters.dll.config` file to match the corresponding fields in the AS user interface.

Or

- Copy the `<APPLINKER_PROPERTY_DEFINITION>` and `<DEVICE name="Adapter-5101">` nodes from Devices.hlx directly to the `Hologic.SecurView.AppSync.Adapters.dll.config` configuration file.

m ***Notes:***

- *If any of the XML nodes are not identical in the two configuration files, Application Synchronization will not function properly.*
- *In order to ease the complexity of testing during the development stage of your application, your application will be able to communicate directly with the Hologic Simulator without Application Synchronization in between. For more information on configuring that test environment please see* *Appendix C: Testing Your Application's AS Adapter Interface*.

9.1. Configuring AS

The AS Adapter will communicate with the Hologic workstation and other connected applications through AS. Therefore the AS's peer properties must be set for use by the connected application (through the AS Adapter).

```
<APPLINKER_PROPERTY_DEFINITION>
    <NAME id="1">Hologic Bridge</ NAME>
    <IP_ADDRESS id="2">201.202.1.1</IP_ADDRESS>
    <DWS_PORT id="3">5100</DWS_PORT>
</APPLINKER_PROPERTY_DEFINITION>
```

m ***Note**: Do not use the loopback IP address 127.0.0.1 in the `<IP_ADDRESS>` tag. Always use the real IP address of the computer where AS is running.*

9.2. Configuring Adapter Properties

9.2.1. COMM Node

This is the sample XML document that explains the XML schema for the AS Adapter's configuration:

```
<DEVICE name=”Adapter-5101”>
    <COMM comm_type_id="1" type="TCPIP">
      <IP_ADDRESS>201.203.0.1</IP_ADDRESS>
      <PORT_NUMBER>5101</PORT_NUMBER>
    </COMM>
</DEVICE>
```

Since the communication mechanism between the AS Adapter and AS is TCP/IP, the name (alias), IP address, and listening port of the connected application must be defined.

Each connected application must have a locally unique name (alias).

m ***Note**: Do not use the loopback IP address 127.0.0.1 in the `<IP_ADDRESS>` tag. Always use the real IP address of the computer where the AS Adapter is running.*

10.0. AS Adapter Logging

The AS Adapter logging is intended to provide services for application debugging and auditing.

The AS Adapter logging utilizes the log4net logging system.

10.1. Logging Configuration

The log4net environment is flexible and fully configurable using the XML configuration files.

The log4net.xml configuration file is located in the same executable directory where the AS Adapter (Hologic.SecurView.AppSync.Adapters.dll) assembly is located.

For more information about configuring please see the log4net online documentation:
http://logging.apache.org/log4net/release/manual/configuration.html

10.2. Logging Output

The output log statements are written to a log file located in the "Logs" directory, which is a sub-directory of the one where the AS Adapter assembly is located.

The default name for the log file is `log.txt.<file index>` (for example, `log.txt.0, log.txt.1`). The default logging is based on a RollingFileApender whose maximum file size is configured to 250 KB. Once the log.txt.0 log file exceedes that limit, the output will be written to the next file index, log.txt.1

Loggers *may* be assigned levels. The following levels are defined in order of increasing priority:

- ALL
- DEBUG
- INFO
- WARN
- ERROR
- FATAL
- OFF

For auditing purposes, the logging level is set to INFO. The INFO log level will output the messages of level INFO and all levels below it in the above mentioned

hierarchy (INFO, WARN, ERROR, FATAL). For development or debugging purposes, the log level may be set to DEBUG or ALL.

11.0. AS Adapter Reference for a Managed CA

11.1. AsAdapter Class

```
namespace Hologic.SecurView.AppSync.Adapters
```

11.1.1. AsAdapter Communication Methods

The user of the AsAdapter class has explicit control of when to initiate and terminate listening for incoming requests and responses from AS.

Method AsAdapter.Start

```
void Start();
```

Start listening for incoming requests and responses. If your application does not accept requests or responses, this method should not be called.

Method AsAdapter.Stop

```
void Stop();
```

Stop listening for incoming requests and responses. Stop should be called only if Start was called previously to initiate a listening connection.

11.1.2. AsAdapter Request Methods

The AsAdapter class provides the following methods that can be called when the connected application wants to send a request to a Hologic workstation or other connected applications.

Method AsAdapter.Login

```
string Login(User user);
```

Initiate login request, for the specified user.

Parameters:

- `User`: User information.

Return Value: Unique request identifier.

Method AsAdapter.Logout

```
string Logout(User user);
```

Initiate logout request, for the specified user.

Parameters:

- `User`: User information.

Return Value: Unique request identifier.

Method AsAdapter.OpenPatient

```
string OpenPatient(User user, Patient patient);
```

Initiate patient synchronization request, for the specified patient and user.

Parameters:

- `User`: User information. Optional parameter, not required if CA does not support login/logout. When a Hologic workstation receives an OpenPatient request, it does not expect User information.
- `Patient`: Patient information.

Return Value: Unique request identifier.

Method AsAdapter.UpdatePatientState

```
string UpdatePatientState(User user, Patient patient);
```

Initiate request for updating patient study information.

Parameters:

- `User`: User information. Optional parameter, not required if CA does not support login/logout. When a Hologic workstation receives an UpdatePatientState request, it does not expect User information.
- `Patient`: Patient information.

Return Value: Unique request identifier.

11.1.3. AsAdapter Response Methods

Method AsAdapter.Response

```
void Response(string requestUID, ResponseStatus status, string
reason);
```

Parameters:

- `requestUID`: must match the value received with the corresponding request event.
- `ResponseStatus`: enum {Success, Failed, Pending, NotSupported}
- `reason`: empty string unless status is Failed. If the status is Failed, the CA's class may provide the reason for failure. The other CAs may use it for logging and debugging purposes.

11.1.4. AsAdapter Request Events

```
namespace Hologic.SecurView.AppSync.Adapters
```

The AsAdapter class provides the following request events that can be used only by managed connected applications.

The events are triggered when a Hologic workstation or other connected applications issue a request:

Event AsAdapter.OnLogin

```
void OnLogin(User user, string requestUID);
```

Receives login request, for the specified user.

Parameters:

- `User`: User information.
- `requestUID`: Unique request identifier.

Event AsAdapter.OnLogout

```
void OnLogout(User user, string requestUID);
```

Receives logout request, for the specified user.

Parameters:

- `User`: User information.
- `requestUID`: Unique request identifier.

Event AsAdapter.OnOpenPatient

```
void OnOpenPatient(User user, Patient patient, string requestUID);
```

Receives the patient synchronization request, for the specified patient and user.

Parameters:

- `User`: User information.
- `Patient`: Patient information.
- `requestUID`: Unique request identifier.

Event AsAdapter.OnUpdatePatientState

```
void OnUpdatePatientState(User user, Patient patient, string
requestUID);
```

Receives the request for updating patient study information.

Parameters:

- `User`: User information.
- `Patient`: Patient information.
- `requestUID`: Unique request identifier.

Event AsAdapter.OnResponse

```
void OnResponse(string requestUID, ResponseStatus status, string
reason);
```

Parameters:

- `requestUID`: Used to map a response uniquely to each individual request. The call to a request method such as Login, Logout, OpenPatient, or UpdatePatientState will return the unique request identifier (requestUID). The returned UID should be checked that it matches the requestUID received in the OnResponse event.
- `ResponseStatus`: enum {Success, Failed, Pending, NotSupported}
- `reason`: provided with Failed status

Event AsAdapter.OnError

```
void OnError(Exception exception);
```

The AsAdapter operates as an asynchronous system, so any exception is communicated back to the connected application only using an OnError event. All recoverable errors are handled within the AsAdapter class, and all errors and warnings are logged. Only exceptions that are not handled are thrown back to the connected application using an OnError event. The system may not continue to operate as expected if this event is fired.

Parameters:

- `exception`: The standard .NET Exception, conveying the reach error information.

11.2. Data Objects

11.2.1. User

Each property value must comply with the format and VM identified in section **14.5**.

```
namespace Hologic.SecurView.AppSync.Common.MessageObjects
```

Properties:

```
string LoginNameXApp { get; set; }
string PasswordXApp { get; set; }
string RealName { get; set; }
```

11.2.2. Patient

Each property value must comply with the format and VM identified in section 14.1.

```
namespace Hologic.SecurView.AppSync.Common.MessageObjects
```

Properties:

```
string PatientsName  { get; set; }
string PatientID { get; set; }
string BirthDate { get; set; }
string BirthTime { get; set; }
string EthnicGroup { get; set; }
string PatientComments { get; set; }
string Sex { get; set; }
string State { get; set; }
string Note { get; set; }
string ReasonForNote { get; set; }
string Exams { get; set; }
Study MostRecentStudy { get; }
List<Study> Study { get; set; }
List<string> OtherPatientName { get; set; }
List<string> OtherPatientID { get; set; }
```

11.2.3. Reader

Each property value must comply with the format and VM identified in section 14.4.

```
namespace Hologic.SecurView.AppSync.Common.MessageObjects
```

Properties:

```
string RealName { get; set; }
```

11.2.4. Report

Each property value must comply with the format and VM identified in section 14.3.

```
namespace Hologic.SecurView.AppSync.Common.MessageObjects
```

Properties:

```
string State { get; set; }
string Created { get; set; }
List<Reader> Reader { get; set; }
```

11.2.5. Study

Each property value must comply with the format and VM identified in section **14.2**.

```
namespace Hologic.SecurView.AppSync.Common.MessageObjects
```

Properties:

```
string StudyInstanceUID  { get; set; }
string StudyID  { get; set; }
string AccessionNumber  { get; set; }
string Age  { get; set; }
string Occupation  { get; set; }
string Size  { get; set; }
string Weight  { get; set; }
string StudyDate  { get; set; }
string StudyTime  { get; set; }
string StudyDescription  { get; set; }
string InstitutionName  { get; set; }
string History  { get; set; }
string Type  { get; set; }
string ReadTwice  { get; set; }
string CAD  { get; set; }
string State  { get; set; }
string Physician  { get; set; }
List<Report> Report { get; set; }
List<string> OperatorsName { get; set; }
List<string> Modalities { get; set; }
```

12.0. AS Adapter Reference for an Unmanaged COM CA

Interfaces exposed to an unmanaged COM connected application.

12.1. Interface ISecurView

```
namespace  Hologic.SecurView.AppSync.Adapters
```

12.1.1. ISecurView Communication Methods

The user of ISecurView interface has explicit control of when to initiate and terminate listening for incoming requests and responses from AS.

Method ISecurView.Start

```
void Start();
```

Start listening for incoming requests and responses. If your application does not accept requests or responses, this method should not be called.

Method ISecurView.Stop

```
void Stop();
```

Stop listening for incoming requests and responses. Stop should be called only if Start was called previously to initiate a listening connection.

12.1.2. ISecurView Request Methods

The following methods can be called when the connected application wants to send a request to a Hologic workstation or other connected applications.

Method ISecurView.Login

```
string Login(IUser user);
```

Initiate login request, for the specified user.

Parameters:

- `IUser`: User information.

Return Value: Unique request identifier.

Method ISecurView.Logout

```
string Logout(IUser user);
```

Initiate logout request, for the specified user.

Parameters:

- `IUser`: User information.

Return Value: Unique request identifier.

Method ISecurView.OpenPatient

```
string OpenPatient(IUser user, IPatient patient);
```

Initiate patient synchronization request, for the specified patient and user.

Parameters:

- `IUser`: User information. Optional parameter, not required if CA does not support login/logout. When a Hologic workstation receives an OpenPatient request, it does not expect User information.
- `IPatient`: Patient information.

Return Value: Unique request identifier.

Method ISecurView.UpdatePatientState

```
string UpdatePatientState(IUser user, IPatient patient);
```

Initiate request for updating patient study information.

Parameters:

- `IUser`: User information. Optional parameter, not required if CA does not support login/logout. When a Hologic workstation receives an UpdatePatientState request, it does not expect User information.
- `IPatient`: Patient information.

Return Value: Unique request identifier.

12.1.3. ISecurView Response Methods

Method ISecurView.Response

```
void Response(string requestUID, ResponseStatus status, string
reason);
```

Parameters:

- `requestUID`: must match the value received with the corresponding request event.
- `ResponseStatus`: enum {Success, Failed, Pending, NotSupported}
- `reason`: empty string unless status is Failed. If the status is Failed, the CA's class may provide the reason for failure. The other CAs may use it for logging and debugging purposes.

12.2. Source Interface ISecurViewEvents

```
namespace Hologic.SecurView.AppSync.Adapters
```

This interface is required only for COM Interop. It is the outgoing event interface which the COM Sink implements. It is a dispinterface, so that it remains friendly to scripting clients.

The following events are triggered when a Hologic workstation or other connected application issues a request.

Event ISecurViewEvents.OnLogin

```
void OnLogin(IUser user, string requestUID);
```

Receives login request, for the specified user.

Parameters:

- `IUser`: User information.
- `requestUID`: Unique request identifier.

Event ISecurViewEvents.OnLogout

```
void OnLogout(IUser user, string requestUID);
```

Receives logout request, for the specified user.

Parameters:

- `IUser`: User information.
- `requestUID`: Unique request identifier.

Event ISecurViewEvents.OnOpenPatient

```
void OnOpenPatient(IUser user, IPatient patient, string requestUID);
```

Receives the patient synchronization request, for the specified patient and user.

Parameters:

- `IUser`: User information.
- `IPatient`: Patient information.
- `requestUID`: Unique request identifier.

Event ISecurViewEvents.OnUpdatePatientState

```
void OnUpdatePatientState(IUser user, IPatient patient, string
requestUID);
```

Receives the request for updating patient study information.

Parameters:

- `IUser`: User information.
- `IPatient`: Patient information.
- `requestUID`: Unique request identifier.

Event ISecurViewEvents.OnResponse

```
void OnResponse(string requestUID, ResponseStatus status, string
reason);
```

Parameters:

- `requestUID`: Used to map a response uniquely to each individual request. The call to a request method such as Login, Logout, OpenPatient, or UpdatePatientState will return the unique request identifier (requestUID). The returned UID should be checked that it matches the requestUID received in the OnResponse event.
- `ResponseStatus`: enum {Success, Failed, Pending, NotSupported}
- `reason`: provided with Failed status

Event ISecurViewEvents.OnError

```
void OnError(Exception exception);
```

The AsAdapter operates as an asynchronous system, so any exception is communicated back to the connected application using an OnError event. All recoverable errors are handled within the AS Adapter, and all errors or warnings are logged. Only exceptions that are not handled are thrown back to the connected application using an OnError event. The system may not continue to operate as expected if this event is fired.

Parameters:

- `exception`: The standard .NET Exception, conveying the reach error information.

12.3. Data Interfaces

An unmanaged connected application must access the data objects through data interfaces:

12.3.1. Interface IUser

Each property value must comply with the format and VM identified in section **14.5**.

```
namespace Hologic.SecurView.AppSync.Common.MessageObjects
```

Properties:

```
string LoginNameXApp { get; set; }
string PasswordXApp { get; set; }
string RealName { get; set; }
```

12.3.2. Interface IPatient

Each property value must comply with the format and VM identified in section **14.1**.

```
namespace Hologic.SecurView.AppSync.Common.MessageObjects
```

Properties:

```
string PatientsName { get; set; }
string PatientID { get; set; }
string BirthDate { get; set; }
string BirthTime { get; set; }
string EthnicGroup { get; set; }
string PatientComments { get; set; }
string Sex { get; set; }
string State { get; set; }
string Note { get; set; }
string ReasonForNote { get; set; }
string Exams { get; set; }
IStudy MostRecentStudy { get; }
IEnumerator Studies { set; }          // Collection of IStudy
IEnumerator StudiesOrderedByDate { get; }    // Collection of IStudy
IEnumerator OtherPatientNames { get; set; }  // Collection of strings
IEnumerator OtherPatientIDs { get; set; }    // Collection of strings
```

12.3.3. Interface IReader

Each property value must comply with the format and VM identified in section **14.4**.

```
namespace Hologic.SecurView.AppSync.Common.MessageObjects
```

Properties:

```
string RealName { get; set; }
```

12.3.4. Interface IReport

Each property value must comply with the format and VM identified in section **14.3**.

```
namespace Hologic.SecurView.AppSync.Common.MessageObjects
```

Properties:

```
string State { get; set; }
string Created { get; set; }
IEnumerator Readers { get; set; } // Collection of IReader
```

12.3.5. Interface IStudy

Each property value must comply with the format and VM identified in section **14.2**.

```
namespace Hologic.SecurView.AppSync.Common.MessageObjects
```

Properties:

```
string StudyInstanceUID  { get; set; }
string StudyID  { get; set; }
string AccessionNumber  { get; set; }
string Age  { get; set; }
string Occupation  { get; set; }
string Size  { get; set; }
string Weight  { get; set; }
string StudyDate  { get; set; }
string StudyTime  { get; set; }
string StudyDescription  { get; set; }
string InstitutionName  { get; set; }
string History  { get; set; }
string Type  { get; set; }
string ReadTwice  { get; set; }
string CAD  { get; set; }
string State  { get; set; }
string Physician  { get; set; }
IEnumerator Modalities  { get; set; } // Collection of strings
IEnumerator Reports { get; set; }           // Collection of IReport
IEnumerator OperatorsNames { get; set; }    // Collection of strings
```

13.0. Data Object Property Types

The following value element types are supported:

TypeName	Value
Bool	Boolean value: “True” or “False”
Date	Date as ‘yyyymmdd’ (Year Month Day), for example, 20051107 Or “n/a” if date is invalid or unknown
DateTime	[Date] [Space] [Time], for example, “20051107 113400000”, “20051108 n/a”
Double	Floating point number, might include “e” to indicate start of exponent, for example, “-3.2e-2”
Int	32-bit integer value, for example, “100”
String	Text value, for example, “Patient name”
Time	Time as ‘HHMMSSmmm’ (Hour Minute Second Milliseconds) Or “n/a” if time is invalid or unknown
UInt	32-bit unsigned integer value

14.0. Data Objects and Interfaces Properties

The following tables define the properties used in data objects and interfaces.

Note that in the column "VM" (value multiplicity), a suffix "C" means that the multiplicity specified applies only if the condition specified in the "Description" column is met. All properties with value multiplicity [0..N] or [1..N] are defined as collection properties.

14.1. Patient (IPatient) Properties

The data object ***Patient*** (or IPatient interface) is used for patient synchronization messages (OpenPatient, UpdatePatientState).

Element name	Type	VM	Description
PatientsName	String	0..1	DICOM Tag Patient's Name (0010,0010)
OtherPatientName	String	0..N	DICOM Tag Other Patient Names (0010,1001)
PatientID	String	0..1	Identification element: DICOM Tag Patient ID (0010,0020) When the value of PatientID is "CLEAR" in an OpenPatient message sent by the CA, if configured to do so SecurView and MultiView will close the viewer if a patient is currently open for review.
OtherPatientID	String	0..N	Identification element: DICOM Tag Other Patient IDs (0010,1000)
BirthDate	Date	0..1	DICOM Tag Patient's Birth Date (0010,0030)
BirthTime	Time	0..1	DICOM Tag Patient's Birth Time (0010,0032)
EthnicGroup	String	0..1	DICOM Tag Ethnic Group (0010,2160)
PatientComments	String	0..1	DICOM Tag Patient Comments (0010,4000)
Sex	String	0..1	DICOM Tag Patient's Sex (0010,0040)
State	String	0..1	Patient's reading state, must be one of "Not Read", "Changed", "Old" or "Read"
Note	String	0..1	Patient lock, must be "Pending" if present
ReasonForNote	String	0..1C	Must not be present if "Note" is not present, else must be one of "Consultation required", "Additional images required", "Additional images arrived", "Pending"

Exams	Uint	0..1	Total number of available studies for the patient
Study	Study Node	0..N	Identification element: A study of the patient

Since SecurView makes no assumptions about the connected application's capabilities, it will provide all available patient and study information in the patient synchronization messages it sends (OpenPatient, UpdatePatientState). MultiView can be configured to include PatientID, AccessionNumber, and/or StudyInstanceUID in the message it sends.

While it is recommended that a connected application also provides all available patient and study information in the patient synchronization messages it sends (OpenPatient, UpdatePatientState), SecurView uses the fields that are present among PatientID (with optional OtherPatientID), AccessionNumber, StudyID, and StudyInstanceUID to identify the patient/study. MultiView can be configured to evaluate or ignore PatientID, AccessionNumber, and/or StudyInstanceUID.

14.2. Study (IStudy) Properties

The data object ***Study*** (or IStudy interface) is used for patient synchronization messages (OpenPatient, UpdatePatientState). It includes study identification data. For status synchronization, it may include additional report information.

Element name	Type	VM	Description
StudyInstanceUID	String	0..1	Identification element: DICOM Tag Study Instance UID (0020,000D) When the value of StudyInstanceUID is "1.2.276.0.28.3.9.99999" in an OpenPatient message sent by the CA, if configured to do so SecurView and MultiView will close the viewer if a patient is currently open review.
StudyID	String	0..1	DICOM Tag Study ID (0020,0010) When the value of StudyID is "CLEAR" in an OpenPatient message sent by the CA, if configured to do so SecurView will close the viewer if a patient is currently open for review. MultiView ignores StudyID.
AccessionNumber	String	0..1	Identification element: DICOM Tag Accession Number (0008,0050)

Element name	Type	VM	Description
			When the value of AccessionNumber is "CLEAR" in an OpenPatient message sent by the CA, if configured to do so SecurView and MultiView will close the viewer if a patient is currently open for review.
Age	String	0..1	DICOM Tag Patient's Age (0010,1010)
Occupation	String	0..1	DICOM Tag Occupation (0010,2180)
Size	Double	0..1	DICOM Tag Patient's Size (0010,1020)
Weight	Double	0..1	DICOM Tag Patient's Weight (0010,1030)
Modalities	String	0..N	DICOM Tags SOP Class UID (0008,0016) and Modality (0008,0060) Each value is formatted as "[SOPClassUID]"-"[Modality]" Including the quotes, for example, "1.2.840.10008.5.1.4.1.1.1.2"-"MG"
StudyDate	Date	0..1	DICOM Tag Study Date (0008,0020)
StudyTime	Time	0..1	DICOM Tag Study Time (0008,0030)
StudyDescription	String	0..1	DICOM Tag Study Description (0008,1030)
InstitutionName	String	0..1	DICOM Tag Institution Name (0008,0080)
OperatorsName	String	0..N	DICOM Tag Operators' Name (0008,1070) Order is by series datetime, descending
History	String	0..1	DICOM Tag Additional Patient History (0010,21B0)
Type	String	0..1	If present, must be one of "Screening", "Diagnostic"
ReadTwice	Bool	0..1	True if study is to be read twice
CAD	Bool	0..1	True if there is a CAD report for the study
State	String	0..1	Study reading state. If present, must be one of "Not Read", "Changed", "Old", "Read", "Read Once"
Physician	String	0..1	DICOM Tag Referring Physician's Name (0008,0090)
Report	Report Node	0..NC	This element may be included if the message is an UpdatePatientState message.

14.3. Report (IReport) Properties

The data object ***Report*** (or IReport interface) may be used for update patient state synchronization messages. It is a child element of the Study node. A report element indicates that a user read the study.

Element name	Type	VM	Description
State	String	1	Must be "Read"
Reader	Reader Node	1	Real name of the person doing the reporting
Created	DateTime	1	Date and time the report was created

m ***Note****: MultiView does not send update patient state synchronization messages at this time.*

14.4. Reader (IReader) Properties

The data object ***Reader*** (or IReader interface) includes the information of a report creator. It may be used in update patient state synchronization messages. It is a child element of the Report node.

Element name	Type	VM	Description
RealName	String	1	Real name of the reader.

14.5. User (IUser) Properties

The data object ***User*** (or IUser interface) includes information used to identify a user for user synchronization and optionally for patient synchronization messages.

Element name	Type	VM	Description
LoginNameXApp	String	1	Login name of the user on the CA
PasswordXApp	String	1	Login password of the user on the CA
RealName	String	0..1	Real name of the CA user. Always present when SecurView sends a message. MultiView does not include in sent messages. Not used by SecurView or MultiView in received messages.

15.0. Encryption

All data supplied to AS Adapter will be encrypted before sending to AS. Also, all data in requests received from AS are encrypted and will be decrypted before an event is triggered.

16.0. Deployment

Each partner that is using AS Adapter will choose its own particular deployment strategy. Therefore, Hologic does not provide an installer for the AS Adapter, as it is expected that the AS Adapter will be packed and deployed along with the connected application.

16.1. Deploying the required files

The following files and directories must be in the same directory as the client executable:

```
BlowfishNET.dll
Hologic.Logging.Adapter.dll
Hologic.Logging.Data.dll
Hologic.SecurView.AppSync.Adapters.dll
Hologic.SecurView.AppSync.Adapters.dll.config
Hologic.SecurView.AppSync.Common.dll
Hologic.SecurView.AppSync.Encryption.dll
Log.ClassificationList.xml
Log.Configuration.xml
Log.Events.xml
log4net.dll
log4net.xml
\MessageFormatFiles\Configure.xslt
\MessageFormatFiles\Login.xslt
\MessageFormatFiles\Logout.xslt
\MessageFormatFiles\OpenPatient.xslt
\MessageFormatFiles\Response.xslt
\MessageFormatFiles\StudyUpdate.xslt
```

These files are distributed with the AS Adapter CD, located in the ".₩Lib" directory, and should be managed and copied to the required destination by the connected application's developer or installer.

m ***Note**: Run the "VerifyAsAdapter.bat" file located in the same directory as the client executable, to verify that all required files are deployed, and to make the configuration files writable.*

Appendix A:Using AsAdapter from a Managed CA

These code excerpts are only general guidelines for how to use the AS Adapter and data objects in a managed connected application. For more information, please see the sample application supplied on the AS Adapter CD.

Referencing AsAdapter and Data Objects:

```
using Hologic.SecurView.AppSync.Adapters;
using Hologic.SecurView.AppSync.Common.MessageObjects;
```

Creating the AsAdapter object:

```
AsAdapter adapter = new AsAdapter();
```

Initiate listening for incoming requests and responses

```
adapter.Start();
```

Subscribing to incoming request events:

```
adapter.OnLogin += new OnLoginEventHandler(OnLogin);
adapter.OnLogout += new OnLogoutEventHandler(OnLogout);
adapter.OnOpenPatient += new OnOpenPatientEventHandler(OnOpenPatient);
adapter.OnUpdatePatientState += new
         OnUpdatePatientStateEventHandler(OnUpdatePatientState);
adapter.OnResponse += new OnResponseEventHandler(OnResponse);
adapter.OnError += new OnErrorEventHandler(OnError);
```

Terminate listening for incoming requests and responses

```
adapter.Stop();
```

Unsubscribing to incoming request events:

```
adapter.OnLogin -= new OnLoginEventHandler(OnLogin);
adapter.OnLogout -= new OnLogoutEventHandler(OnLogout);
adapter.OnOpenPatient -= new OnOpenPatientEventHandler(OnOpenPatient);
adapter.OnUpdatePatientState -= new

  OnUpdatePatientStateEventHandler(OnUpdatePatientState);
adapter.OnResponse -= new OnResponseEventHandler(OnResponse);
adapter.OnError -= new OnErrorEventHandler(OnError);
```

Sending Request to AS:

```
User user = new User();
user.LoginNameXApp = "login";
user.PasswordXApp = "pwd";
user.RealName = "Nick Kramer";

Patient patient = new Patient();
patient.PatientsName = "Smith^John";
patient.PatientID = "JS Test 1";

string requestUID = adapter.Login(user);
string requestUID = adapter.Logout(user);

// User is an optional parameter for OpenPatient / UpdatePatientState.
string requestUID = adapter.OpenPatient(user, patient);
string requestUID = adapter.UpdatePatientState(user, patient);
```

Receiving Request from AS:

```
static void OnUpdatePatientState(User user, Patient patient, string
requestUID)
{
  Console.WriteLine("OnUpdatePatientState Event (User:{0}, Pwd:{1},
Patient  Name:{2}, PatientId:{3}, RequestUID:{4})",
  user.LoginNameXApp, user.PasswordXApp, patient.PatientsName,
         patient.PatientID, requestUID);
  adapter.Response(requestUID, ResponseStatus.Success, "");
}

static void OnOpenPatient(User user, Patient patient, string
requestUID)
{
  Console.WriteLine("OnOpenPatient Event (User:{0}, Pwd:{1}, Patient
Name:{2},PatientId:{3}, RequestUID:{4})",
  user.LoginNameXApp, user.PasswordXApp, patient.PatientsName,
         patient.PatientID, requestUID);
  adapter.Response(requestUID, ResponseStatus.Success, "");
}

static void OnResponse(string requestUID, ResponseStatus status,
string reason)
{
  Console.WriteLine("OnResponse Event (RequestUID:{0}, Status:{1},
Reason:{2})",   requestUID, status.ToString(), reason);
}

static void OnLogin(User user, string requestUID)
{
  Console.WriteLine("OnLogin Event (User:{0}, Pwd:{1},
RequestUID:{2})",             user.LoginNameXApp, user.PasswordXApp,
requestUID);
  adapter.Response(requestUID, ResponseStatus.Success, "");
}

static void OnLogout(User user, string requestUID)
{
  Console.WriteLine("OnLogout Event (User:{0}, Pwd:{1},
RequestUID:{2})",      user.LoginNameXApp, user.PasswordXApp,
requestUID);
  adapter.Response(requestUID, ResponseStatus.Success, "");
}

static void OnError(Exception exception)
{
```

```
  Console.WriteLine("Non-recoverable error: {0}”, exception-
>ToString());
}
```

Appendix B:Using ISecurView .NET from COM

These code excerpts are only general guidelines for how to use the AS Adapter with ISecurView, ISecurViewEvents and data interfaces in an unmanaged connected application. For more information, please see the sample application supplied on the AS Adapter CD.

B.1. Registering .NET Assemblies

Connected applications that use an unmanaged code environment must use the AS Adapter as a COM object. In that case the AS `Hologic.SecurView.AppSync.Adapters.dll` and `Hologic.SecurView.AppSync.Common.dll` assemblies must be registered in the system registry. This process is done automatically by the Hologic Application Synchronization installer application on systems where Application Synchronization is installed. If the AS Adapter is to be used on a system where AS is not installed, the connected application's installation procedure is responsible for registering the assemblies.

Both assemblies are not strongly named; therefore they must reside in the same directory as the unmanaged client executable.

To register the assemblies, go to the application executable directory and execute "RegisterAssemblies.bat".

B.2. Deploying the Required Files

For the list of required files, see section **16.0**.

B.2.1. Deploying the required files for unmanaged Visual C++ clients

The required files and directories must be in the same directory as the unmanaged client executable (*.EXE).

If the client for AS Adapter is a C++ COM DLL, the required files must be located where the EXE client of that particular COM DLL is located.

B.2.2. Deploying the required files for unmanaged Visual Basic clients

If the client for AS Adapter is VB6 EXE, DLL or OCX, the following rules apply:

If the client executable is a compiled VB6 COM client running outside the VB6 IDE, the executable directory is simply the directory containing the compiled client executable (*.EXE).

When running a VB6 COM client within the VB6 IDE, the executable directory becomes the directory containing the VB6.EXE itself (for example, `C:\Program Files\Microsoft Visual Studio\VB98`), *NOT* where the unmanaged client executable is!

B.2.3. Deploying the required files for Internet Explorer clients

If the COM client for the AS Adapter, or the AS Adapter COM itself is run within Internet Explorer, the required files **must be in the same directory as iexplore.exe** (for example, `C:\WINDOWS\ServicePackFiles\i386\iexplore.exe`), *NOT* where the unmanaged client executable is!

B.2.4. Deploying the required files for AS Adapter sample application

Since the AsAdapterSampleAppForUnmanagedClient is a Visual C++ project, the required files must be located in its executable directory (for example, `.\AsAdapterSampleAppForUnmanagedClient\Debug`)

In the `.\AsAdapterSampleAppForUnmanagedClient\AsAdapterTestApp.vcproj` sample project, these files are copied to the target directory in the Pre-Build Event step. Open the AsAdapterTestApp.vcproj properties dialog, and in the "Configuration Properties" tree-node select the Pre-Build Event.

B.3. Reference ISecurView .NET from COM

From the point of view of client and server code, the differences between COM and the .NET Framework are largely invisible. Microsoft Visual Basic clients can view a .NET object in the object browser, which exposes the object methods and syntax, properties, and fields exactly as if it were any other COM object.

The process for importing a type library is slightly more complicated for C++ clients.

To reference .NET object members from an unmanaged C++ client, reference the COM type library file (supplied with our API) with the **#import** directive. When referencing a type library from C++, the **raw_interfaces_only** option must be specified.

Importing Libraries

Specify the **raw_interfaces_only** option in the **#import** directive. For example:

C++

```
#import "mscorlib.tlb" // Provides wrapper for Exception class.
#import ".\Hologic.SecurView.AppSync.Common.tlb" raw_interfaces_only
#import ".\Hologic.SecurView.AppSync.Adapters.tlb" raw_interfaces_only
```

Accessing the IUser and IPatient interfaces

```
namespace Hologic_SecurView_AppSync_Common
{
    IUserPtr     ptrUser(__uuidof(User));
    IPatientPtr  ptrPatient(__uuidof(Patient));
    IStudyPtr    ptrStudy(__uuidof(Study));
    IReportPtr   ptrReport(__uuidof(Report));
    IReaderPtr   ptrReader(__uuidof(Reader));
}
```

Accessing the ISecurView interface

```
namespace Hologic_SecurView_AppSync_Adapters
{
    ISecurViewPtr ptrUser(__uuidof(AsAdapter));
}
```

Calling a set Property

```
ptrUser->put_LoginNameXApp(_bstr_t("userXApp"));
```

Calling a get Property

```
CComBSTR user;
ptrUser->get_LoginNameXApp(&user);
```

Calling a Method

```
ptrSecurView->Login(ptrUser);
```

Handling the Incoming Request Events

Only code excerpts relevant to event handling are included in this sample:

```
class CYourClass :
  …,
public IDispEventImpl<IDC_AsAdapter, CYourClass,
  &Hologic_SecurView_AppSync_Adapters::DIID_ISecurViewEvents,
  &Hologic_SecurView_AppSync_Adapters::LIBID_Hologic_SecurView_AppSync
_Adapters,      1, 0>
{
public:
  // Sink Map for events
  BEGIN_SINK_MAP(CYourClass)
    SINK_ENTRY_EX(IDC_AsAdapter,
         Hologic_SecurView_AppSync_Adapters::DIID_ISecurViewEvents, 1,
OnLogin)
    SINK_ENTRY_EX(IDC_AsAdapter,
         Hologic_SecurView_AppSync_Adapters::DIID_ISecurViewEvents, 2,
OnLogout)
    SINK_ENTRY_EX(IDC_AsAdapter,
         Hologic_SecurView_AppSync_Adapters::DIID_ISecurViewEvents, 3,
                OnOpenPatient)
    SINK_ENTRY_EX(IDC_AsAdapter,
         Hologic_SecurView_AppSync_Adapters::DIID_ISecurViewEvents, 4,
                OnUpdatePatientState)
    SINK_ENTRY_EX(IDC_AsAdapter,
         Hologic_SecurView_AppSync_Adapters::DIID_ISecurViewEvents, 5,
                OnResponse)
    SINK_ENTRY_EX(IDC_AsAdapter,
         Hologic_SecurView_AppSync_Adapters::DIID_ISecurViewEvents, 6,
OnError)
  END_SINK_MAP()

  // Event Handlers
  HRESULT __stdcall OnLogin(Hologic_SecurView_AppSync_Common::IUserPtr
user,           CComBSTR requestUID);
  HRESULT __stdcall
OnLogout(Hologic_SecurView_AppSync_Common::IUserPtr user,
  CComBSTR requestUID);
  HRESULT __stdcall
OnOpenPatient(Hologic_SecurView_AppSync_Common::IUserPtr          user,
Hologic_SecurView_AppSync_Common::IPatientPtr patient, CComBSTR
  requestUID);
  HRESULT __stdcall

  OnUpdatePatientState(Hologic_SecurView_AppSync_Common::IUserPtr
user,           Hologic_SecurView_AppSync_Common::IPatientPtr patient,
CComBSTR              requestUID);
  HRESULT __stdcall OnResponse(CComBSTR requestUID,
                Hologic_SecurView_AppSync_Adapters::ResponseStatus
responseStatus,         CComBSTR reason);
  HRESULT __stdcall OnError(mscorlib::_Exception* exception);

//---------------------------------------------------------------------
-----
// Advise and unadvise connection to source events
//---------------------------------------------------------------------
-----

HRESULT CYourClass::Advise()
{
  // Connect this container to the AsAdapters outgoing interface
  if (ptrISecurView_ == NULL)
         _com_issue_error(E_POINTER);
```

```
  // connect the sink and source, m_spActiveCase is the source COM
object
  HRESULT hr = DispEventAdvise(ptrISecurView_);
  if (FAILED(hr))
         _com_issue_error(hr);

  return hr;
}


HRESULT CYourClass::Unadvise()
{
  // Disconnect this container from the AsAdapters outgoing interface
  HRESULT hr(S_OK);

  if ((m_dwEventCookie != 0xFEFEFEFE) && (ptrISecurView_ != NULL))
  {
         hr = DispEventUnadvise(ptrISecurView_);
         if (FAILED(hr)) _com_issue_error(hr);
  }
  return hr;
}

//-----------------------------------------------------------------------
-----
// Event Handlers
//-----------------------------------------------------------------------
-----

HRESULT.CYourClass::OnLogin(Hologic_SecurView_AppSync_Common::IUserPtr
user, CComBSTR requestUID)
{
  ptrISecurView_->Response(requestUID,
  Hologic_SecurView_AppSync_Adapters::ResponseStatus_Success,
_bstr_t(""));          return S_OK;
}

HRESULT.CYourClass::OnLogout(Hologic_SecurView_AppSync_Common::IUserPt
r user,  CComBSTR requestUID)
{
  ptrISecurView_->Response(requestUID,
  Hologic_SecurView_AppSync_Adapters::ResponseStatus_Success,
_bstr_t(""));
  return S_OK;
}

HRESULT
CYourClass::OnOpenPatient(Hologic_SecurView_AppSync_Common::IUserPtr
user,     Hologic_SecurView_AppSync_Common::IPatientPtr patient,
CComBSTR requestUID)
{
  ptrISecurView_->Response(requestUID,
  Hologic_SecurView_AppSync_Adapters::ResponseStatus_Success,
_bstr_t(""));
  return S_OK;
}

HRESULT.CYourClass::OnUpdatePatientState(Hologic_SecurView_AppSync_Com
mon::IUserPt    r user, Hologic_SecurView_AppSync_Common::IPatientPtr
patient, CComBSTR      requestUID)
{
  ptrISecurView_->Response(requestUID,
  Hologic_SecurView_AppSync_Adapters::ResponseStatus_Success,
_bstr_t(""));
  return S_OK;
}
```

```
HRESULT CYourClass::OnResponse(CComBSTR requestUID,
  Hologic_SecurView_AppSync_Adapters::ResponseStatus responseStatus,

  CComBSTR reason)
{
  // Use the incoming parameters …
  return S_OK;
}

HRESULT CYourClass::OnError(mscorlib::_Exception* exception)
{
  CComQIPtr<mscorlib::_Exception> spException(exception);
  _bstr_t errorDescription = spException->GetToString();
  MessageBox (errorDescription, _T("Hologic AsAdapter Error"),
MB_ICONERROR);
  return S_OK;
}
```

B.4. Borland Compiler Addendum

When the AS Adapter DLL is used by source code compiled with a Borland (Delphi or C++) compiler, it throws an Arithmetic Exception (floating point error). To automatically suppress Arithmetic Exceptions during compilation, add the following lines of code:

```
const MCW_EM = DWord($133f);
begin
Set8087CW(MCW_EM);
end;
```

Appendix C:Testing Your Application's AS Adapter Interface

Your application will be using the AS Adapter to communicate with a Hologic workstation through Application Synchronization.

In order to ease the complexity of testing during the development stage of your application, Hologic has provided the Hologic Simulator (included on the distribution CD), which can simulate the operation of a Hologic workstation with Application Synchronization.

m ***Note**: Application Synchronization (AS) is not included on the AS Adapter distribution CD. For more information about AS installation and configuration, please contact Hologic.*

These are the recommended steps to expedite the development, integration, and testing of your application:

- Before you start development of your application with the AS Adapter, it is advisable first to run a demo with the Hologic Sample Application and the Hologic Simulator, to get an indication of how the communication works. For more information how to run this demo, please see **Appendix D: Run Demo with Hologic Sample Application and Hologic Simulator**.
- During the development phase of your application, you will be testing its communication directly with the Hologic Simulator. You may use the same configuration settings as explained in **Appendix D: Run Demo with Hologic Sample Application and Hologic Simulator,** however you will need to substitute your application (YourApplication.exe) in place of the Hologic Sample Application (AsAdapterTestApp.exe).
- Once you complete development of your application with the AS Adapter and are ready for integration test, you may try a demo of the simplified system (Hologic Sample Application, Application Synchronization and Hologic Simulator), as explained in **Appendix E: Run Demo with Hologic Sample Application, Application Synchronization (AS) and Hologic Simulator**.
- The final step of the integration testing would be to test your application with Application Synchronization and the Hologic Simulator.

  You may use the same configuration settings as explained in **Appendix E: Run Demo with Hologic Sample Application, Application Synchronization (AS) and Hologic Simulator**, however you will need to substitute your

application (YourApplication.exe) in place of the Hologic Sample Application (AsAdapterTestApp.exe).

Appendix D: Run Demo with Hologic Sample Application and Hologic Simulator

This demo shows how the Hologic Sample Application that is using the AS Adapter sends and receives messages with the Hologic Simulator.

This is a simplified demo, in which Application Synchronization is not used as a communication bridge between the two applications.

This configuration is usually used during the development stage of your application, when Application Synchronization is not needed. Your application will be able to communicate directly with the Hologic Simulator, just as the Hologic Sample Application does.

You may re-use the same configuration settings as explained in this demo when testing your application. You will need to substitute your application instead of the Hologic Sample Application.

There are two scenarios most common for testing:

- Hologic Sample Application and the Hologic Simulator reside on the same computer (single computer installation).
- Hologic Sample Application and the Hologic Simulator reside on different computers (two computer installation).

In terms of setup and testing, both scenarios are equivalent, except that there are two different IP addresses in the two computer installation scenario.

m ***Note:*** *The "Hologic Sample Application with AS Adapter" shown in all diagrams below is the AsAdapterSampleAppForUnmanagedClient.*

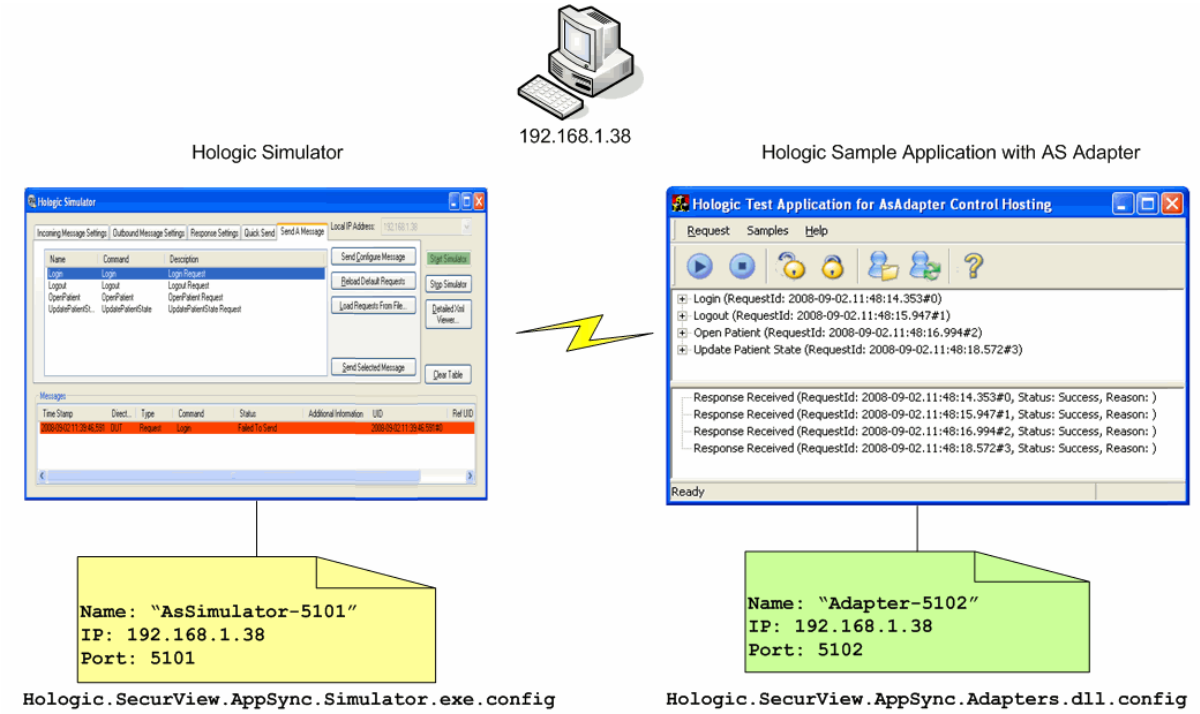
m ***Note:*** *To start receiving incoming requests and responses from the Hologic Simulator, you must press the "Start" button.*

For information regarding the Hologic Sample Application log, please see section **10.0**.

D.1. Single Computer Installation

In a single computer installation, the Hologic Sample Application and the Hologic Simulator are installed on the same computer.

m ***Note**: This test configuration does not include Application Synchronization.*



The Hologic Sample Application will have the following parameters:

- Name: `"Adapter-5102"`
- IP Address: Address of the computer on which `"Adapter-5102"` is running (for example, 192.168.1.38)

  m ***Note**: Do not use the loopback address 127.0.0.1*
- Port: 5102

The Hologic Simulator will have the following parameters:

- Name: `"AsSimulator-5101"`
- IP Address: Address of the computer on which `"AsSimulator-5101"` is running (for example, 192.168.1.38)

  m ***Note**: Do not use the loopback address 127.0.0.1*
- Port: 5101

▶ **To install on a single computer:**

1 Copy the entire directory structure of the Hologic Sample Application from the distribution CD (folder.₩`AsAdapterSampleAppForUnmanagedClient\Release`) to the destination folder (for example, `C:\YourTestApp`).

Directory and file structure must be:

```
YourTestApp\AsAdapterTestApp.exe
YourTestApp\BlowfishNET.dll
YourTestApp\Hologic.Logging.Adapter.dll
YourTestApp\Hologic.Logging.Data.dll
YourTestApp\Hologic.SecurView.AppSync.Adapters.dll
YourTestApp\Hologic.SecurView.AppSync.Adapters.dll.config
YourTestApp\Hologic.SecurView.AppSync.Common.dll
YourTestApp\Hologic.SecurView.AppSync.Encryption.dll
YourTestApp\Log.ClassificationList.xml
YourTestApp\Log.Configuration.xml
YourTestApp\Log.Events.xml
YourTestApp\log4net.dll
YourTestApp\log4net.xml

YourTestApp\MessageFormatFiles\Configure.xslt
YourTestApp\MessageFormatFiles\Login.xslt
YourTestApp\MessageFormatFiles\Logout.xslt
YourTestApp\MessageFormatFiles\OpenPatient.xslt
YourTestApp\MessageFormatFiles\Response.xslt
YourTestApp\MessageFormatFiles\StudyUpdate.xslt
```

2 Copy the Hologic Sample Application executables and all required dependent assemblies or DLLs to the same directory.

3 Mark all *.xml files and directories as writable.

4 Edit the `\Hologic.SecurView.AppSync.Simulator.exe.config` file:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="SimulatorSetting" type="a" />
  </configSections>
  <SimulatorSetting>
    <LocalSettings>
      <LocalSetting>
        <Name>AsSimulator-5101</Name>
        <Port>5101</Port>
      </LocalSetting>
    </LocalSettings>
    <RemoteHosts>
      <Host>
        <Name>Adapter-5102</Name>
        <IPAddress>192.168.1.38</IPAddress>
        <Port>5102</Port>
        <Encryption>HologicEncryption</Encryption>
      </Host>
    </RemoteHosts>
  </SimulatorSetting>
</configuration>
```

5 Edit the `YourTestApp\Hologic.SecurView.AppSync.Adapters.dll.conf` file:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <section name="APPLINKER_PROPERTY_DEFINITION" type="a" />
    <section name="DEVICE" type="b" />
  </configSections>
  <APPLINKER_PROPERTY_DEFINITION>
    <NAME>AsSimulator-5101</NAME>
    <IP_ADDRESS>192.168.1.38</IP_ADDRESS>
    <HOLOGIC_AS_PORT>5101</HOLOGIC_AS_PORT>
  </APPLINKER_PROPERTY_DEFINITION>
  <DEVICE name="Adapter-5102">
    <COMM type="TCPIP">
      <IP_ADDRESS>192.168.1.38</IP_ADDRESS>
      <PORT_NUMBER>5102</PORT_NUMBER>
    </COMM>
  </DEVICE>
</configuration>
```

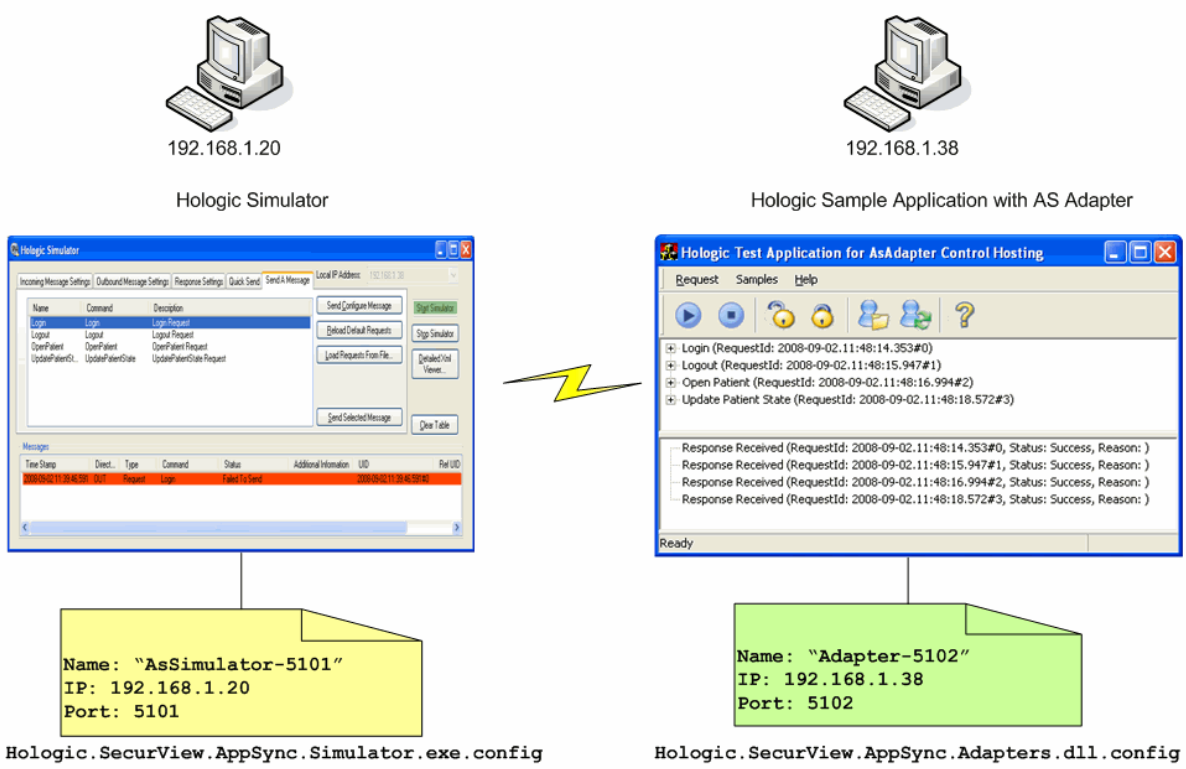
m ***Note**: You may use the AdapterConfigTool.exe to configure this file. The Hologic Adapter Configuration tool is a utility designed to provide an easy method to configure the AS Adapter.*

6 Edit the `YourTestApp\log4net.xml` file: Set the "level" node to have DEBUG value:
`<level value="DEBUG" />`

D.2. Two Computer Installation

In a two computer installation, the Hologic Sample Application and the Hologic Simulator reside on different computers.

m ***Note**: This test configuration does not include Application Synchronization.*



The Hologic Sample Application will have the following parameters:

- Name: `"Adapter-5102"`
- IP Address: Address of the computer on which `"Adapter-5102"` is running (for example, 192.168.1.38)

  m ***Note**: Do not use the loopback address 127.0.0.1*
- Port: 5102

The Hologic Simulator will have the following parameters:

- Name: `"AsSimulator-5101"`
- IP Address: Address of the computer on which `"AsSimulator-5101"` is running (for example, 192.168.1.20)

  m ***Note**: Do not use the loopback address 127.0.0.1*
- Port: 5101

To install on two computers, follow the same procedure as in D.1, except the IP Addresses for the Hologic Simulator and the Hologic Sample Application will be different.

Appendix E: Run Demo with Hologic Sample Application, Application Synchronization (AS) and Hologic Simulator

This demo shows how the Hologic Sample Application that is using the AS Adapter sends and receives messages with the Hologic Simulator through Application Synchronization.

This configuration is usually used prior to integration testing of your application, when Application Synchronization is needed.

You may re-use the same configuration settings as explained in this demo when testing your application. You will need to substitute your application instead of the Hologic Sample Application and change the name "Adapter-5102" to the alias of your application (for example, "YourApplication-5102") in the `Hologic.SecurView.AppSync.Simulator.exe.config` file.

There are three most common scenarios for testing:

- The Hologic Sample Application, Hologic Simulator and Application Synchronization reside on the same computer (single computer installation).
- The Hologic Sample Application and Application Synchronization reside on one computer, while the Hologic Simulator resides on different one (two computer installation).
- The Hologic Sample Application resides on one computer, while the Hologic Simulator and Application Synchronization reside on different one (two computer installation).

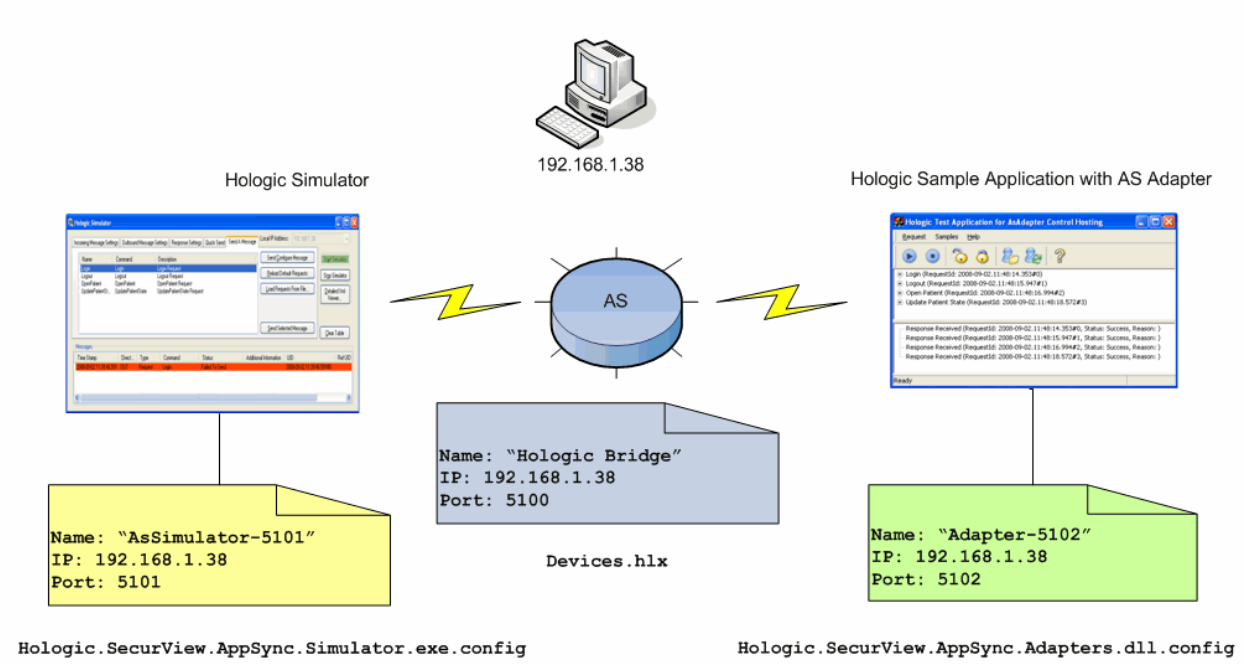
m ***Note**: Application Synchronization (AS) is not included on the AS Adapter distribution CD. For more information about AS installation and configuration, please contact Hologic.*

m ***Note**: The Hologic Sample Application with AS Adapter shown in all diagrams below is the AsAdapterSampleAppForUnmanagedClient.*

m ***Note**: To start receiving incoming requests and responses from the Hologic Simulator, you must press the "Start" button.*

For information regarding the Hologic Test Application log, please see **10.0. AS Adapter Logging**.

E.1. Single Computer Installation



1 Edit the `Hologic.SecurView.AppSync.Simulator.exe.config` file:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="SimulatorSetting" type="a" />
  </configSections>
  <SimulatorSetting>
    <LocalSettings>
      <LocalSetting>
        <Name>AsSimulator-5101</Name>
        <Port>5101</Port>
      </LocalSetting>
    </LocalSettings>
    <RemoteHosts>
      <Host>
        <Name>Hologic Bridge</Name>
        <IPAddress>192.168.1.38</IPAddress>
        <Port>5100</Port>
        <Encryption>HologicEncryption</Encryption>
      </Host>
    </RemoteHosts>
  </SimulatorSetting>
</configuration>
```

2 Edit the `YourTestApp\Hologic.SecurView.AppSync.Adapters.dll.conf` file:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <section name="APPLINKER_PROPERTY_DEFINITION" type="a" />
    <section name="DEVICE" type="b" />
  </configSections>
  <APPLINKER_PROPERTY_DEFINITION>
    <NAME>Hologic Bridge</NAME>
    <IP_ADDRESS>192.168.1.38</IP_ADDRESS>
    <HOLOGIC_AS_PORT>5100</HOLOGIC_AS_PORT>
  </APPLINKER_PROPERTY_DEFINITION>
  <DEVICE name="Adapter-5102">
    <COMM type="TCPIP">
      <IP_ADDRESS>192.168.1.38</IP_ADDRESS>
      <PORT_NUMBER>5102</PORT_NUMBER>
    </COMM>
  </DEVICE>
</configuration>
```

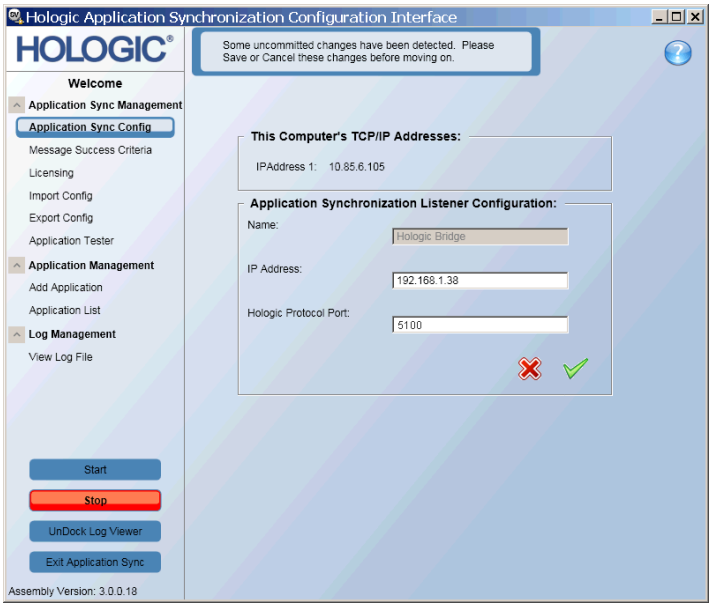
m ***Note**: You may use the AdapterConfigTool.exe to configure this file. The Hologic Adapter Configuration tool is a utility designed to provide an easy method to configure the AS Adapter.*

E.1.1. Configure Application Synchronization

Start Application Synchronization. Next, open the Application Synchronization configuration (right-click the system tray icon).

▶ **To enter the Application Synchronization Listener Configuration:**

1 Click **Application Sync Config**. The Application Synchronization Config page appears.



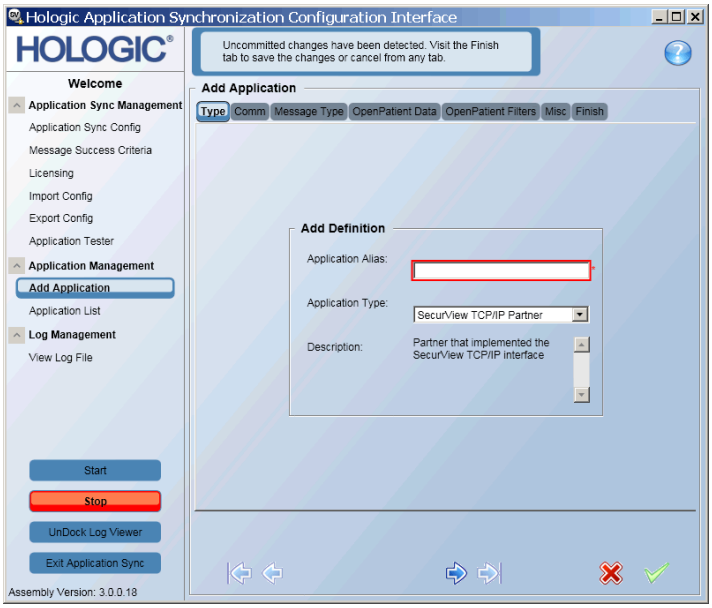
2 Enter (or confirm) the IP address and Hologic Protocol Port for Hologic Bridge.

3 Click ✔ (Save) to validate the IP address and Hologic protocol port number.

▶ **To configure the connected applications:**

To configure Hologic Simulator and Hologic Sample Application with AS Adapter, add two connected applications of the SecurView TCP/IP Partner application type and configure with the corresponding Alias name, IP address, and port.

1 Click **Add Application**. The Type page appears.



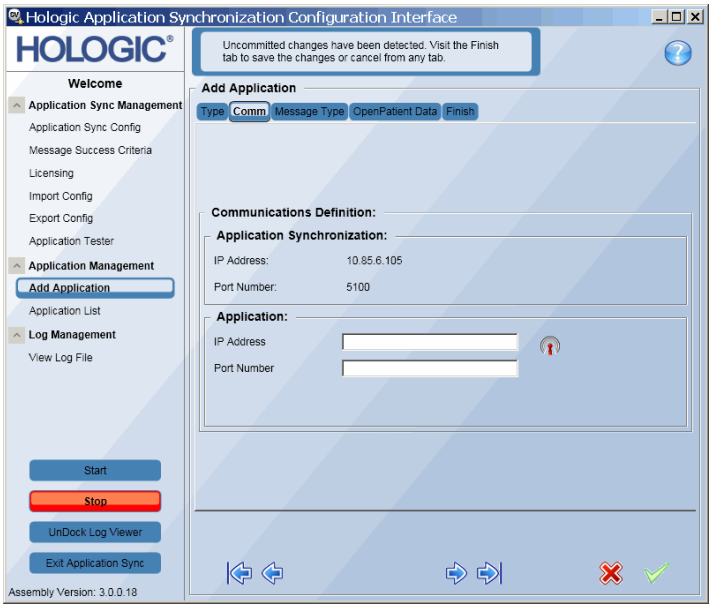
2 In the **Application Type** drop-down list, select SecurView TCP/IP Partner.

m ***Note:*** *When testing your application, if Responses were not implemented select the* ***SecurView TCP/IP Partner (no response)*** *application type.*

3 In the **Application Alias** field, enter the application alias (AS Simulator – 5101 and Adapter – 5102 respectively) and click (Next).

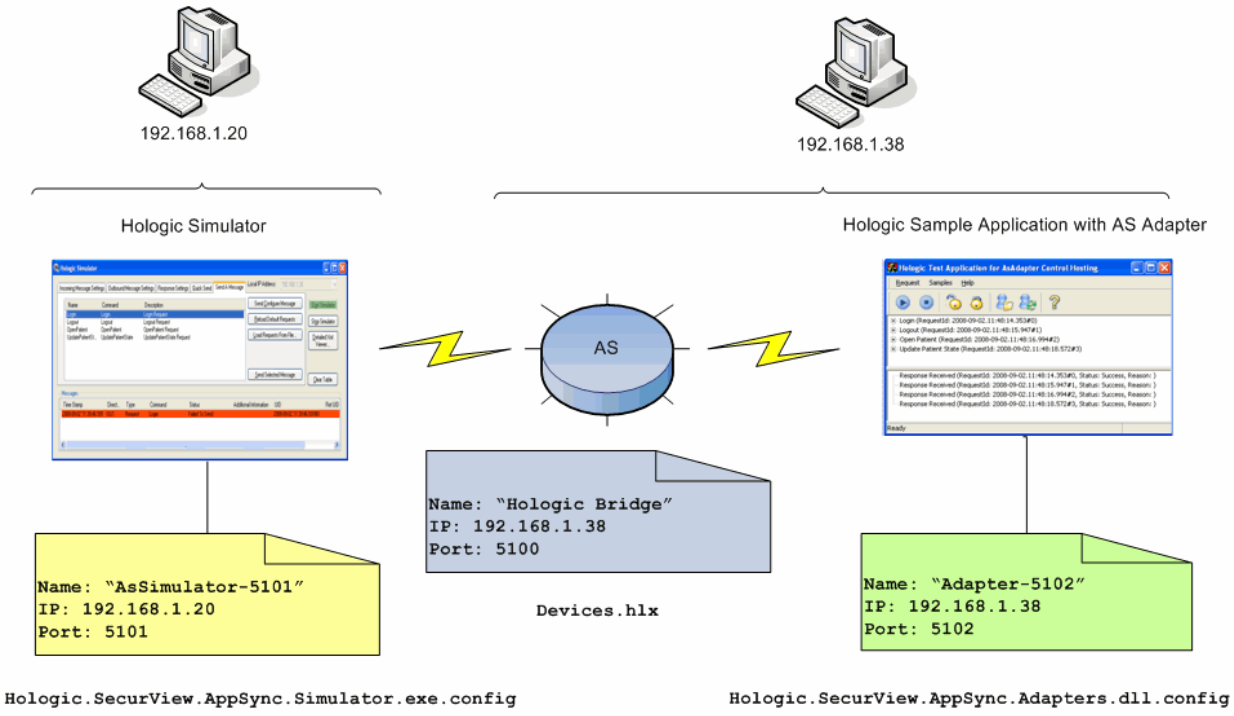
4 On the Comm page, enter the IP address and port number of the connected application. For the Hologic Simulator, the IP address is 192.168.1.38 and the port

is 5101. For Hologic Sample Application with AS Adapter, the IP address is 192.168.1.38 and the port is 5102.



5 Select the **Finish** tab and click **Save**.

E.2. Two Computer Installation (AS with Hologic Sample Application)



1 Edit the `Hologic.SecurView.AppSync.Simulator.exe`.config file:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="SimulatorSetting" type="a" />
  </configSections>
  <SimulatorSetting>
    <LocalSettings>
      <LocalSetting>
        <Name>AsSimulator-5101</Name>
        <Port>5101</Port>
      </LocalSetting>
    </LocalSettings>
    <RemoteHosts>
      <Host>
        <Name>Hologic Bridge</Name>
        <IPAddress>192.168.1.38</IPAddress>
        <Port>5100</Port>
        <Encryption>HologicEncryption</Encryption>
      </Host>
    </RemoteHosts>
  </SimulatorSetting>
</configuration>
```

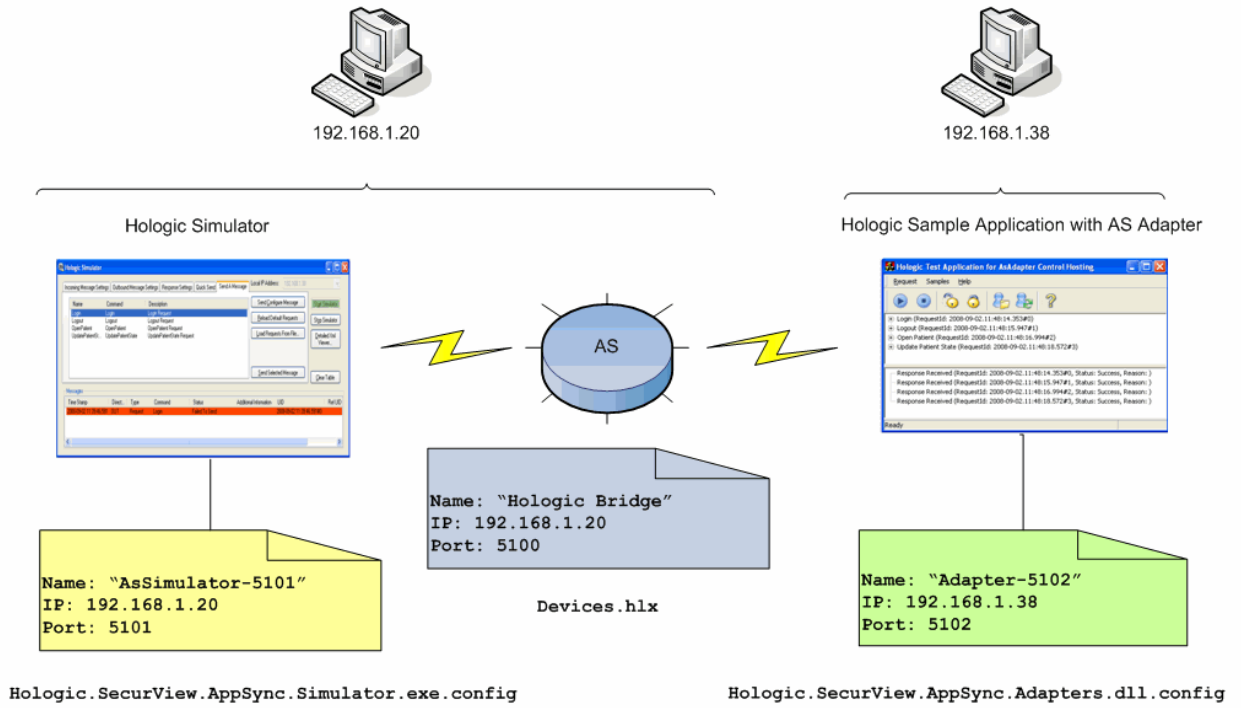
2 Edit the `YourTestApp\Hologic.SecurView.AppSync.Adapters.dll.conf` file:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <section name="APPLINKER_PROPERTY_DEFINITION" type="a" />
    <section name="DEVICE" type="b" />
  </configSections>
  <APPLINKER_PROPERTY_DEFINITION>
    <NAME>Hologic Bridge</NAME>
    <IP_ADDRESS>192.168.1.38</IP_ADDRESS>
    <HOLOGIC_AS_PORT>5100</HOLOGIC_AS_PORT>
  </APPLINKER_PROPERTY_DEFINITION>
  <DEVICE name="Adapter-5102">
    <COMM type="TCPIP">
      <IP_ADDRESS>192.168.1.38</IP_ADDRESS>
      <PORT_NUMBER>5102</PORT_NUMBER>
    </COMM>
  </DEVICE>
</configuration>
```

m ***Note**: You may use the AdapterConfigTool.exe to configure this file. The Hologic Adapter Configuration tool is a utility designed to provide an easy method to configure the AS Adapter.*

3 To enter the Application Synchronization Listener Configuration and configure connected applications for Application Synchronization, see section **E.1.1. Configure Application Synchronization**. Enter the IP Addresses as defined for this configuration.

E.3. Two Computer Installation (AS with Hologic Simulator)



1 Edit the `Hologic.SecurView.AppSync.Simulator.exe.config` file:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="SimulatorSetting" type="a" />
  </configSections>
  <SimulatorSetting>
    <LocalSettings>
      <LocalSetting>
        <Name>AsSimulator-5101</Name>
        <Port>5101</Port>
      </LocalSetting>
    </LocalSettings>
    <RemoteHosts>
      <Host>
        <Name>Hologic Bridge</Name>
        <IPAddress>192.168.1.20</IPAddress>
        <Port>5100</Port>
        <Encryption>HologicEncryption</Encryption>
      </Host>
    </RemoteHosts>
  </SimulatorSetting>
</configuration>
```

2 Edit the `YourTestApp\Hologic.SecurView.AppSync.Adapters.dll.conf` file:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <section name="APPLINKER_PROPERTY_DEFINITION" type="a" />
    <section name="DEVICE" type="b" />
  </configSections>
  <APPLINKER_PROPERTY_DEFINITION>
    <NAME>Hologic Bridge</NAME>
    <IP_ADDRESS>192.168.1.20</IP_ADDRESS>
    <HOLOGIC_AS_PORT>5100</HOLOGIC_AS_PORT>
  </APPLINKER_PROPERTY_DEFINITION>
  <DEVICE name="Adapter-5102">
    <COMM type="TCPIP">
      <IP_ADDRESS>192.168.1.38</IP_ADDRESS>
      <PORT_NUMBER>5102</PORT_NUMBER>
    </COMM>
  </DEVICE>
</configuration>
```

m ***Note**: You may use the AdapterConfigTool.exe to configure this file. The Hologic Adapter Configuration tool is a utility designed to provide an easy method to configure the AS Adapter.*

3 To enter the Application Synchronization Listener Configuration and configure connected applications, see **E.1.1. Configure Application Synchronization**. Enter the IP Addresses as defined for this configuration.