eHealth Integration Sample Code v2.0.3
Source Code Build Guide

2 February 2016

Approved for external use

National E-Health Transition Authority

National E-Health Transition Authority Ltd
Level 25, 56 Pitt Street
Sydney, NSW 2000
Australia
www.nehta.gov.au

Acknowledgements

Council of Australian Governments

The National E-Health Transition Authority is jointly funded by the Australian Government and all State and Territory Governments.

Disclaimer

The National E-Health Transition Authority Ltd (NEHTA) makes the information and other material ('Information') in this document available in good faith but without any representation or warranty as to its accuracy or completeness. NEHTA cannot accept any responsibility for the consequences of any use of the Information. As the Information is of a general nature only, it is up to any person using or relying on the Information to ensure that it is accurate, complete and suitable for the circumstances of its use.

Document control

This document is maintained in electronic form and is uncontrolled in printed form. It is the responsibility of the user to verify that this copy is the latest revision.

Document information

Key information

Owner	Head of Delivery	
Contact for enquiries	NEHTA Help Centre	
	t:	1300 901 001
	e:	help@nehta.gov.au

Product version history

Product version	Date	Release comments
1.0	February 2014	Initial release (HIPS 4.1.0)
2.0	February 2015	See release note (NEHTA-2040:2015) for details of changes and bug fixes.
2.0.1 2.0.2		Unpublished updates.
2.0.3	February 2016	See release note (NEHTA-2185:2016) for details of changes and bug fixes.

Table of contents

1. Purpose

The purpose of this document is to provide straightforward instructions for source code build for eHISC.

2. Source Code Build Prerequisites

This section outlines the major prerequisites that an organisation will need to obtain to compile the source code into executable binaries and deployment packages for eHISC.

2.1 Development Environment

For eHISC the development environment will require the following software packages to be able to build the eHISC source code.

2.1.1 eHISC Core and eHISC-UI

1. Operating System: Microsoft Windows 7 or 8.1
2. Operating system features[1]:
    - 2.1. .NET Framework 3.5[2] Features
        - 2.1.1. Net Framework 3.5
        - 2.1.2. WCF Activation
            - 2.1.2.1. HTTP Activation
            - 2.1.2.2. Non-HTTP Activation
    - 2.2. .NET Framework 4.0
    - 2.3. Message Queuing (MSMQ)
        - 2.3.1. MSMQ Active Directory Domain Services Integration
        - 2.3.2. MSMQ HTTP Support
        - 2.3.3. MSMQ Triggers
        - 2.3.4. Multi-casting Support
    - 2.4. Web Server (Internet Information Services)
    - 2.5. Management Tools
        - 2.5.1. IIS Management Console
        - 2.5.2. IIS Management Scripts and Tools
        - 2.5.3. IIS Management Service
    - 2.6. World Wide Web Services
        - 2.6.1. Application Development
            - 2.6.1.1. .Net Extensibility

[1] Some of the following Windows operating system features may be required. Feature naming may differ between Windows operating system versions.

[2] Depending on operating system and .NET Framework version installed, may also be listed as 3.5.1, 4.0 or 4.5.

2.6.1.2. ASP.NET

2.6.2. Common HTTP Features

2.6.2.1. Default Content

2.6.2.2. Directory Browsing

2.6.2.3. HTTP Errors

2.6.2.4. HTTP Redirection

2.6.2.5. Static Content

2.6.3. Health and Diagnostics

2.6.3.1. HTTP Logging

2.6.3.2. Logging Tools

2.6.3.3. Request Monitor

2.6.3.4. Tracing

2.6.4. Performance

2.6.4.1. Dynamic Content Compression

2.6.4.2. Static Content Compression

2.6.5. Security

2.6.5.1. Basic Authentication

2.6.5.2. Client Certificate Mapping Authentication

2.6.5.3. IIS Client Certificate Mapping Authentication

2.6.5.4. IP and Domain Restrictions (IP Security)

2.6.5.5. Request Filtering

2.6.5.6. URL Authorization

2.6.5.7. Windows Authentication

2.7. Windows Process Activation Service

2.7.1. .Net Environment

2.7.2. Configuration APIs

2.7.3. Process Model

3. Commercial, licensed software:

3.1.1. Visual Studio 2013 Professional: Complete

3.1.2. SQL Server

3.1.2.1. SQL 2008 R2 (Standard or Express): Complete

OR

3.1.2.2. SQL 2012 (Standard or Express): Complete

4. Free, unlicensed software:
    4.1. Visual Studio updates & packages:
        4.1.1. NuGet Package Manager
        4.1.2. Visual Studio 2013 Update 2+[3]
        4.1.3. Entity Framework Power Tools[4]
    4.2. SQL Server Compact 4 Service Pack 1[5]
    4.3. MSBuild extension components
        4.3.1. MSBuild.Community.Tasks
        4.3.2. MSBuildContrib

3. Obtaining & Installing System Certificates

DHS provides a number of PKI certificates for interactions with national systems. The purpose and requirements for usage of these certificates is further explained in http://www.medicareaustralia.gov.au/provider/vendors/pki/files/dhs-ehealth-record-and-nash-pki-certificates-svig-february-2013.pdf.

eHISC employs several of the DHS certificates during its communication with external systems. For development purposes, it is necessary to obtain and install these certificates on the Development workstation.

The following DHS certificates are utilised by eHISC:

DHS Certificate Type	System	Also Known As	Usage
DHS Site PKI Certificate	HI	"Medicare" certificate	Used by eHISC when interacting with the HI Service for IHI lookup and validation.
NASH PKI Certificate for Healthcare Provider Organisations	PCEHR[6]	"PCEHR" or "NASH" certificate	Used by eHISC when interacting with the PCEHR.

Perform the following steps to obtain and install required system certificates:

1. Start the Microsoft Management Console (Start → Run → "mmc.exe").
2. Add the Certificates snap-in to access the Local Machine stores.
    a. File → Add/Remove Snap-In.
    b. Select the "Certificates" Snap-In and click the "Add" button.
    c. Select the "Computer account" option and click the "Next" button.
    d. If prompted to select the computer to manage, select "Local computer" and click the "Finish" button.

[3] Only required if using Visual Studio 2013.
[4] Only required if reverse engineering code first classes.
[5] Only required if working with the HIPS "demo harness".
[6] **Disclaimer**: PCEHR means the My Health Record, formally the "Personally Controlled Electronic Health Record", within the meaning of the *My Health Records Act 2012* (Cth), formerly called the *Personally Controlled Electronic Health Records Act 2012* (Cth).

   e. Click the "OK" button on the Add or Remove Snap-Ins dialog.

3. Repeat the following steps to import each certificate.
   a. Right-click on the Personal store and select All Tasks → Import from the context menu.
   b. Click the "Next" button.
   c. Click the "Browse" to open the file browser dialog.
   d. Set file type filter to "Personal Information Exchange (*.pfx, *.p12)", then browse to and select the certificate to be imported (e.g. "fac_sign.p12").
   e. Click the "OK" button to close the file browser dialog, then the "Next" button.
   f. Enter the password for the certificate, ensure the "Mark as exportable" and "Include extended properties" options are selected, then click the "Next" button.
   g. Ensure the "Place all certificates in the following store" option is selected, and select the "Personal" store, then click the "Next" button.
   h. Click the "Finish" button to complete the task.
   i. Right-click on the installed certificate and click "All Tasks->Manage Private Keys…"
   j. Add the user that will be running the eHISC application to obtain the key – the Process Model Identity if using IIS and usually the local logged on user if running from within Visual Studio.
4. Repeat step 4 as required for each certificate.

Additional Information

The following notes explain how these certificates are linked to records in the eHISC database. In general this linking will already have been performed; the information below is to provide additional context only.

Each Certificate that is installed must be linked to a record in the *hips.HealthProviderOrganisation* table within the eHISC database.

- One record covers both the "Medicare" and "PCEHR" certificate as they link to a hospital for use.
- The HPIO is taken from the Subject-DC field within the "PCEHR" certificate, and in numeric.
- The Name is taken from the Subject-O field within the "PCEHR" certificate.
- The PcehrCertSerial is taken from the Serial Number field (without the spaces) within the "PCEHR" certificate.

The HiCertSerial is taken from the Serial Number field (**without the spaces**) within the "Medicare" certificate.

4. eHISC-Core Solution

4.1 Building and Running the eHISC-Core Solution for the First Time

4.1.1 eHISC Database

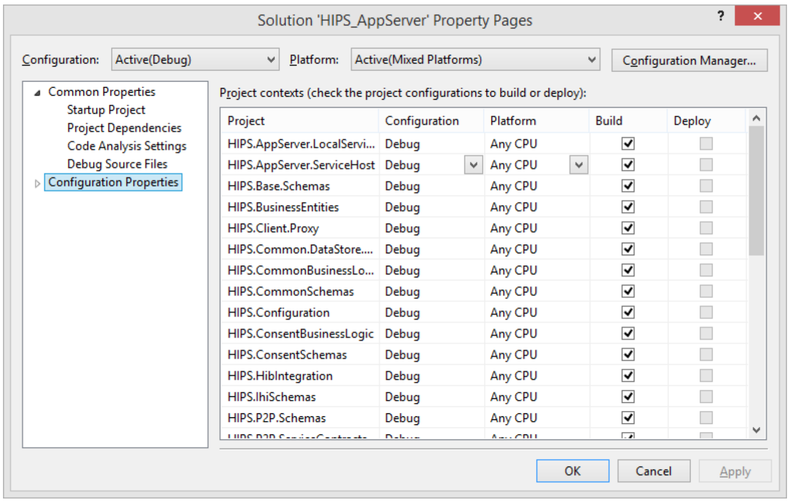
Perform the following steps to build and load the eHISC Database:

1. Following the "eHISC Release **Error! Unknown document property name.** – Initial and Clean Installation Guide" run ALL the scripts from the eHISC\CORE\Database folder after the database has been created. These are pre-collated SQL scripts of all the database scripts included in the source code.
   Note: It is important to read the "Database Preparation" section thoroughly to ensure the database is created with the correct permissions and the scripts that require local data added (such as certificate serial numbers, etc.) have been correctly altered before running the scripts.

4.1.2 eHISC AppServer

Perform the following steps to build and run the *eHISC_AppServer* solution for the first time:

1. Open the solution.
2. Right-click on the solution node in Solution Explorer and select "Properties" from the context menu.
3. On the "Solution Property Pages" dialog, select the "Configuration Properties" node.
   a. Ensure the active solution configuration is set to the "Debug" configuration.
   b. Ensure the platform for each project is set to "Any CPU".
   c. Click the "OK" button.



Solution 'HIPS_AppServer' Property Pages

Configuration: Active(Debug)   Platform: Active(Mixed Platforms)

Project contexts (check the project configurations to build or deploy):

Project	Configuration	Platform	Build	Deploy
HIPS.AppServer.LocalServi...	Debug	Any CPU	☑	☐
HIPS.AppServer.ServiceHost	Debug	Any CPU	☑	☐
HIPS.Base.Schemas	Debug	Any CPU	☑	☐
HIPS.BusinessEntities	Debug	Any CPU	☑	☐
HIPS.Client.Proxy	Debug	Any CPU	☑	☐
HIPS.Common.DataStore....	Debug	Any CPU	☑	☐
HIPS.CommonBusinessLo...	Debug	Any CPU	☑	☐
HIPS.CommonSchemas	Debug	Any CPU	☑	☐
HIPS.Configuration	Debug	Any CPU	☑	☐
HIPS.ConsentBusinessLogic	Debug	Any CPU	☑	☐
HIPS.ConsentSchemas	Debug	Any CPU	☑	☐
HIPS.HibIntegration	Debug	Any CPU	☑	☐
HIPS.IhiSchemas	Debug	Any CPU	☑	☐
HIPS.P2P.Schemas	Debug	Any CPU	☑	☐

4. Build the solution.
5. Right-click on the *HIPS.AppServer.LocalServiceHost* project in Solution Explorer (under the "Development Service Host" Solution Folder) and select "Set as StartUp Project" from the context menu.
6. Review the contents of the *app.config* file in the *HIPS.AppServer.LocalServiceHost* project and ensure it matches your development configuration. In particular, ensure the "PcehrDataStoreConnectionString" in the *connectionStrings* section matches the SQL Server database engine instance name and database to be employed for development.
7. Run the solution in debug mode (F5).
8. If prompted by Windows Firewall to allow network connections, click to allow.

9. Ensure the service host window opens and reflects that the services started successfully:

10. If you encounter a message similar to:
"HTTP could not register URL http://+:PortNumber/Servicename. Your process does not have access rights to this namespace"

Then it is possible that your local user cannot create the specified URL due to permissions. The solution is to add the running local user to the ACL (access control list). Using the following command permission can be given to the running process, which is usually the user that is logged on to the development machine:

netsh http add urlacl url=http://+:***<PortNumber>***/ user=***<LocalUser>***

4.2 Debugging Setup Requirements

Visual Studio adds in an XML node when debugging called "VsDebuggerCausalityData" to the SOAP messages. This is used by Visual Studio to help it debug the WCF calls using system diagnostics.

This XML node cannot be added to the SOAP requests when sent to the National Repository (PCEHR), thus it must be removed. This is done by running the WCF Diagnostics Registration Tool - vsdiag_regwcf.exe. This is located (on a 64 bit installation) in "C:\Program Files\Microsoft Visual Studio 11.0\Common7\IDE", thus it may not be included in the path of the VS Developer Command Prompt.

To uninstall the following command needs to be run from a command prompt as an administrator (located as part of the standard Visual Studio 2013 Tools):

```
vsdiag_regwcf.exe –u
```

5. eHISC-UI Web Solution

5.1 Assumed Prerequisites

Prior to configuring your Development workstation to work with eHISC-UI Web:

1. Ensure you have configured the environment for eHISC-Core development and are able to run a local development copy of the services provided by eHISC_AppServer.

5.2 Building and Running the eHISC-UI Solution for the First Time

Perform the following steps to build and run the *eHISC-UI* Web solution for the first time:

1. Open the *eHISC.Web* solution.

2. Build the solution.

3. Right-click on the *HIPS.Web.UI* project in Solution Explorer and select "Set as StartUp Project" from the context menu.

4. Review the contents of the *web.config* file in the *HIPS.Web.UI* project and ensure it matches your development configuration. In particular, ensure:
    - The “HipsWebDataStoreConnectionString” in the *connectionStrings* section matches the SQL Server database engine instance name and database to be employed for development.
5. Run the solution in debug mode (F5).
6. If prompted for credentials, enter Windows credentials for the local development workstation.

NOTE: The eHISC.Web solution has a local reference to the compiled HIPS.AppServer dlls. If the HIPS.AppServer source code is modified and rebuilt then the referenced dlls in the eHISC\UI\eHISC Web UI Source\trunk\HIPS_Web\References folder must be updated. This can be either done manually or via a Post-Build Event from the HIPS.AppServer.LocalServiceHost project. There is an existing set of “IF EXIST” statements within this project properties Build Events that can be used to automate the copying process of the referenced dlls for the Web UI during a build cycle.