<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>3</td>
</tr>
<tr>
<td>1.1 What this guide covers</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Open Source Licensing</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Quick links</td>
<td>4</td>
</tr>
<tr>
<td>1.4 Acknowledgements</td>
<td>4</td>
</tr>
<tr>
<td>1.5 Conventions used in this manual</td>
<td>4</td>
</tr>
<tr>
<td>Requirements</td>
<td>7</td>
</tr>
<tr>
<td>2.1 Developer Skills</td>
<td>7</td>
</tr>
<tr>
<td>2.2 Applications</td>
<td>8</td>
</tr>
<tr>
<td>2.3 Websites</td>
<td>9</td>
</tr>
<tr>
<td>2.4 Tools &amp; Extensions</td>
<td>9</td>
</tr>
<tr>
<td>Components</td>
<td>11</td>
</tr>
<tr>
<td>3.1 Visual Studio Projects</td>
<td>11</td>
</tr>
<tr>
<td>3.2 Other Information</td>
<td>14</td>
</tr>
<tr>
<td>Source Control</td>
<td>15</td>
</tr>
<tr>
<td>4.1 GitHub</td>
<td>15</td>
</tr>
<tr>
<td>4.2 Helpful Links</td>
<td>17</td>
</tr>
<tr>
<td>Coding Standards</td>
<td>19</td>
</tr>
<tr>
<td>5.1 General Guidelines</td>
<td>19</td>
</tr>
<tr>
<td>5.2 Comments</td>
<td>20</td>
</tr>
<tr>
<td>Windows Installer</td>
<td>23</td>
</tr>
<tr>
<td>6.1 Organize Your Setup</td>
<td>23</td>
</tr>
<tr>
<td>6.2 Specify Application Data</td>
<td>24</td>
</tr>
<tr>
<td>6.3 Configure the Target System</td>
<td>24</td>
</tr>
<tr>
<td>6.4 Customize the Setup Appearance</td>
<td>25</td>
</tr>
<tr>
<td>6.5 Define Setup Requirements and Actions</td>
<td>25</td>
</tr>
<tr>
<td>New Releases</td>
<td>27</td>
</tr>
<tr>
<td>7.1 Building the Tool</td>
<td>27</td>
</tr>
<tr>
<td>7.2 Distributing the Release</td>
<td>29</td>
</tr>
<tr>
<td>ArcGIS Registration</td>
<td>31</td>
</tr>
<tr>
<td>8.1 Registration During Build</td>
<td>31</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>8.2</td>
<td>Registration During Installation</td>
</tr>
<tr>
<td>9</td>
<td>Documentation</td>
</tr>
<tr>
<td>9.1</td>
<td>reStructuredText</td>
</tr>
<tr>
<td>9.2</td>
<td>GitHub</td>
</tr>
<tr>
<td>9.3</td>
<td>ReadTheDocs</td>
</tr>
<tr>
<td>10</td>
<td>Issue Logging</td>
</tr>
<tr>
<td>10.1</td>
<td>ALERC Forum</td>
</tr>
<tr>
<td>10.2</td>
<td>GitHub Issues Log</td>
</tr>
<tr>
<td>11</td>
<td>Database Updater</td>
</tr>
<tr>
<td>11.1</td>
<td>Components</td>
</tr>
<tr>
<td>11.2</td>
<td>Source Control</td>
</tr>
<tr>
<td>11.3</td>
<td>Building Releases</td>
</tr>
<tr>
<td>11.4</td>
<td>Distribution</td>
</tr>
<tr>
<td>11.5</td>
<td>Scripts</td>
</tr>
<tr>
<td>12</td>
<td>Appendix</td>
</tr>
<tr>
<td>12.1</td>
<td>GNU Free Documentation License</td>
</tr>
</tbody>
</table>
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts and no Back-Cover Texts. A copy of the license is included in the Appendix section.
The most up to date version of this documentation can be found in HTML and PDF form on ReadTheDocs.

1.1 What this guide covers

This guide explains a little about how the HLU Tool source code is structured, what you may need to develop and support it, how to build and distribute it, and how to maintain the associated online guides.

Requirements (page 7) summarises the applications, websites, tools, skills & experience that a developer may need.

Components (page 11) outlines the major components of the tool’s source code.

Source Control (page 15) introduces the basics of the GitHub version control repository where the source code is hosted.

Coding Standards (page 19) proposes some general guidelines for coding standards, metadata and comments when making code changes.

Windows Installer (page 23) summarises the features and configuration options that are used for creating an installer for the tool.

New Releases (page 27) lists the steps to follow when building the tool and distributing a new release.

ArcGIS Registration (page 31) describes how and why the tool is registered as an ArcGIS extension.

Documentation (page 35) gives an overview of the associated online guides and how to maintain them.

Issue Logging (page 39) introduces the online forum and issue log where problems and enhancements can be proposed and discussed.

Appendix (page 49) contains a copy of the GNU Free Documentation License v1.3 covering this guide.

1.2 Open Source Licensing

The code for the HLU Tool is ‘open source’ and is released under GNU General Public License (GPL) v3. Users are free to install it on as many computers as they like, and to redistribute it according to the GPLv3 license.

This guide is released under GNU Free Documentation License (FDL) v1.3. Permission is granted to copy, distribute and/or modify this document under the terms of the license.
These licenses are designed to guarantee the user’s freedom to share and change the software and documentation under the terms of the licenses (but note that no additional restrictions may be applied to any new products resulting from changes to the HLU Tool or associated documentation).

Please remember, however, that the tool cost a lot of money to develop and still requires further development and ongoing support. Hence any contributions towards costs would be gratefully received. Enquiries can be made via the ALERC forum.

1.3 Quick links

The following are links to some of the websites relevant to the use, support and development of the tool:

- ALERC Forum - Announcements, bug reports, user Q&A and feature discussions.
- Releases - Release notes and installers for ArcGIS and MapInfo systems.
- Source Code - Repositories for the source code of the tool and associated online guides.
- Issue Log - Known issues and existing change requests.
- User Guide - Online guide for users of the tool.
- Technical Guide - Online guide for administrators and technical users.
- Developer’s Guide - Online guide for developers (this guide).

1.4 Acknowledgements

Many thanks are due to all the LRCs in the south-east of England and their staff who have, and continue to, fund and support the development of the HLU Tool. It takes a small army of developers, testers and users to build a truly useful tool (especially users who care enough to test new releases, report bugs and discuss feature requests).

1.5 Conventions used in this manual

The following typographical conventions are used in this manual:

Ctrl-A Indicates a key, or combination of keys, to press.

Commit Indicates a label, button or anything that appears in user interfaces.

Tools... -> About Indicates a menu choice, or a combination of menu choices, tab selections or GUI buttons.

C:\Program Files\HLU Tool Indicates a filename or directory name.

Tip: Tips can help save time or provide shortcuts.

Note: Notes explain things in more detail or highlight important points.
Caution: Warnings where developers should pay attention.
Requirements

Software developers will come from different backgrounds, have different skills and experience and may use different development environments and tools. Whilst one developer may prefer to have formal training and extensive experience in a particular programming language or concept before tackling a task, another may be comfortable to use their experience in other languages or platforms to pick things up as they go.

As a result, it is not possible to come up with a definitive list of all the skills, experience and tools that a developer must have or use in order to undertake development and support of the HLU Tool. Therefore the following is just a guide to the applications, websites and tools that a developer may wish to use and the skills & experience that a developer may need.

2.1 Developer Skills

The following is a list of programming concepts, languages, skills & experience that developers may need before attempting to develop or support the HLU Tool and associated guides.

- An understanding and experience of general software development concepts such as design, coding, testing, debugging, issue management and deployment.
- Experience of programming and debugging applications with Microsoft’s Visual Studio.
- Experience in C#, .NET and object-oriented programming concepts.
- Ability to write Structured Query Language (SQL).
- An understanding of how relational databases work and Relational Database Management Systems (RDBMS) such as Microsoft SQL Server and Microsoft Access.
- Familiarity with Microsoft’s .Net Framework component LINQ (Language Integrated Query).
- Experience of using ArcGIS Desktop, in particular for creating & editing data and performing basic geoprocessing functions such as split & merge.
- A understanding of ArcObjects and experience of developing with ArcObjects in .NET.
- Experience of using MapInfo Professional, in particular for creating & editing data.
- Experience of developing MapBasic programs, a BASIC-like programming language used to create custom applications for use with MapInfo.
- A General understanding of XML and XAML used to write Windows Presentation Foundation (WPF) applications. WPF is used for the tool’s user interface.
• Understanding of source code version control system concepts and features, particularly Git and GitHub.

• Familiarity with GitHub Flavoured MarkDown text syntax used for GitHub ReadMe files, issues and comments.

• Ability to read/write reStructuredText markup syntax used to create the user, technical and developer’s guides.

• How online documentation repositories such as ReadTheDocs work.

2.2 Applications

The following is a list of applications that you are likely to need as a minimum to develop and support the HLU Tool:

• Microsoft Visual Studio Professional 2012 (or later)
• Microsoft .Net Framework 3.5 (or higher)
• ArcGIS 10.1 (or later)
• MapInfo 8 (or later)
• SQL Server Express 2008 (or later)
• Microsoft Access 2000 (or later)
• Git for Windows
• GitHub for Windows
• InstallShield Limited Edition
2.3 Websites

The following are websites that you will need to use and become familiar with when developing and supporting the HLU Tool and associated user, technical and developer’s guides:

GitHub repository Hosts the source code for the tool and accompanying user and technical guides (including this developer’s guide), and is also used as the issue tracker for any agreed known issues and proposed change requests for the tool.

See GitHub (page 15) and GitHub Issues Log (page 39) for more details.

ReadTheDocs online guides Creates and hosts the User Guide, Technical Guide and Developer’s Guide (this guide) for the tool.

See ReadTheDocs (page 38) for more details.

ALERC Forum Hosts a discussion forum available to Local Record Centres (LRCs) for the HLU Tool and associated Habitat Framework.

See ALCER Forum (page 39) for more details.

2.4 Tools & Extensions

The following is a list of free tools and extensions that may be useful when developing and maintaining the tool or the associated user, technical and developer’s guides.

Active Python A cross-platform distribution of Python that provides a one-step installation of all essential Python modules.

Sphinx A python based documentation generator that can generate HTML, PDF and other output formats.

Notepad 2-mod A fast and light-weight Notepad-like text editor with syntax highlighting.

Sublime Text 3 The latest version of the sophisticated text editor for code and markup languages.
Components

Within the Visual Studio solution for the HLU Tool there are six projects. Five of these projects are .NET assemblies that relate to various aspects of the tool functionality. The other project is an InstallShield project used to create a Windows Installer for the tool.

The following sections contain a brief description of each of the projects and their contents.

3.1 Visual Studio Projects

3.1.1 HLUGISTool

This is the main assembly that contains all of the user interfaces, the ‘business logic’ and also handles the data connection with the chosen relational database. There are a few ad-hoc classes in the parent folder but the majority of the source code is structured in the following sub-folders.

Converters  There are four converter classes in this folder that are used by the user interfaces and related classes to convert data values to/from more user-friendly visible values.

Data  There are a number of data classes in this folder that support connections between the tool and different relational database systems, including ODBC, Oracle, PostGres, SQL Server and MS Access. The recommended connection methods are SQL Server and Access, and it is not known how much testing has been done using the other connection methods.

Date  This folder contains a vague date class (VagueDate) that handles formatting and validation of the database date values to/from a user-friendly ‘vague’ format.

GISApplication  This folder contains the central business logic classes, including the main class (Gis-App) that is an abstract class for all the core tool functionality.

This folder contains two sub-folders MapInfo and ArcGIS that inherit the GisApp abstract class and override it will all the GIS application specific functionality.

Note:  The ArcGIS sub-folder is only used in the combined ArcGIS/MapInfo variant of the tool.

Icons  This folder contains all the icons and button images that appear on the user-interfaces.

UI  This folder contains all the user interface related source code in three sub-folders:

• UserControls contains bespoke classes used by one or more of the user interfaces.
• **View** contains the source code for user interface presentation. The user interfaces were developed using Windows Presentation Framework (WPF) and consists of a main interface (WindowsMain) and nine other sub-interfaces.

• **ViewModel** contains the business logic associated with each of the user interfaces.

### 3.1.2 HluArcMapExtension

This assembly is an ArcGIS Desktop Extension that provides the interface between the main HLU GIS Tool assembly and ArcGIS. It consists of a main class **HluArcMapExtension** that performs the editing and geoprocessing commands (e.g. select features, update attributes, split/merge features, zoom to features) requested by the main assembly.

This extension also passes back any information requested by the main assembly (e.g. list valid HLU feature layers, query selected features, is editing active). Commands and requests are passed between the main assembly and this assembly using Named Pipes. See [Named Pipes](#) (page 14) for more details.

The assembly must be registered as an ArcGIS Desktop Extension before it can interface with ArcGIS. See [ArcGIS Registration](#) (page 31) for more details.

**Note:** This assembly is only used in the combined ArcGIS/MapInfo variant of the tool.

### 3.1.3 AppModule.InterProcessComm

This assembly contains just the Named Pipes interfaces plus the logic for exception handling and the connection state for inter-process communication.

There are three interfaces defined within this assembly - **IChannelManager**, **IClientChannel** and **InterProcessConnection**. These interfaces are introduced in order to abstract the Named Pipes implementation from clients involved in the IPC.

**Note:** This assembly is only used in the combined ArcGIS/MapInfo variant of the tool.

### 3.1.4 AppModule.NamedPipes

This assembly contains all the .NET Named Pipes classes used by the HLU Tool. It is referenced by both the HLU GIS Tool ‘client’ assembly and the HLU ArcMap Extension ‘server’ assembly for inter-process communication.

Outlined below are the main responsibilities of the classes present in the assembly:

• **NamedPipeNative**: This utility class exposes kernel32.dll methods for Named Pipes communication. It also defines constants for some of the error codes and method parameter values.

• **NamedPipeWrapper**: This class is a wrapper around NamedPipesNative. It uses the exposed kernel32.dll methods to provide controlled Named Pipes functionality.

• **APipeConnection**: An abstract class, which defines the methods for creating Named Pipes connections, reading and writing data. This class is inherited by the ClientPipeConnection and ServerPipeConnection classes, used by client and server applications respectively.
- **ClientPipeConnection**: Used by client applications to communicate with server ones by using Named Pipes.

- **ServerPipeConnection**: Allows a Named Pipes server to create connections and exchange data with clients.

- **PipeHandle**: Holds the operating system native handle and the current state of the pipe connection.

**Note**: This assembly is only used in the combined ArcGIS/MapInfo variant of the tool.

### 3.1.5 ArcObjectsInstaller

This assembly contains a custom *Installer class* that is included within the installer. The installer class is recognised by the Windows installer which instantiates the class and calls various methods when an install/uninstall is performed to register the HluArcMapExtension assembly with ArcGIS. See *ArcGIS Registration* (page 31) for more details.

**Note**: This assembly is only used in the combined ArcGIS/MapInfo variant of the tool.

### 3.1.6 HluSetup_ISLE

This project is an InstallShield Limited Edition installation project that creates a Windows Installer for the tool. The various elements of the installer can be defined using the various views in InstallShield’s user interface. See *Windows Installer* (page 23) for more details.
3.2 Other Information

3.2.1 Named Pipes

Inter-Process Communication (IPC) is a set of techniques for the exchange of data among multiple threads in one or more processes. Processes may be running on one or more computers connected by a network. IPC techniques include Named Pipes, File Mapping, Mailslot, Remote Procedure Calls (RPC), etc.

Named pipes are a mechanism for one-way or bi-directional inter-process communication between two or more processes. Named Pipes are sections of shared memory used by separate processes to communicate with one another. The application that creates a pipe is the pipe server. A process that connects to the pipe server is a client. It is most useful in situations where one application is exchanging frequent short text messages with another, located on the same machine or within the same LAN.

The HLU Tool uses Named Pipes in the ArcGIS implementation for communicating between the tool user-interface and the ArcGIS extension. It allows the user-interface to interrogate data and instigate actions within the ArcGIS desktop process started by the tool.

**Server-side logic**

1. Create a named pipe.
2. Listen (wait) for the client to connect.
3. Once connected, read the client’s request from the pipe and write the response.
4. Disconnect the pipe, and close the handle.

**Client-side logic**

1. Try to open a named pipe.
2. Once open, set the read mode and the blocking mode of the specified named pipe.
3. Send a message to the pipe server and receive its response.
4. Close the pipe.
4.1 GitHub

The HLU Tool source code is hosted by GitHub in the HLUTool repository. GitHub is a well known web-based hosting service for software development projects. It uses the Git Version Control System (VCS) which, like all version control systems, records changes to a file or set of files over time so that specific versions can be recalled later.

If you are signed in to GitHub you can Fork the repository to create your own local copy, or alternatively you can download a Zip copy of all the source files.

4.1.1 Branches

There are two main branches in the HLUTool repository:

- **master**: contains the source code for the ArcGIS/MapInfo variant of the tool
- **master-mapinfo**: contains the source code for the MapInfo only variant of the tool

4.1.2 Commits

There are numerous commits to both of the above branches each of which incrementally contributes towards one of the releases for each variant of the tool. Each commit has a short summary and a longer text description of the changes that it contains and lists the files that were added/deleted/amended. Details of the actual lines added/deleted can also be viewed on GitHub.

New commits relating to changes to the tool in the future should similarly provide a summary and description of the changes, and ideally changes should be split/grouped where possible so that each relates to a single change or fix. This helps to determine what changes were made for each change/fix in the event that they need to be amended or reverted.

4.1.3 Tags

The source code for every version of the tool from v1.0.1 to v2.3.0 is Tagged on GitHub. The HLUTool Tags point to a specific ‘commit’ in a branch to indicate that the commit relates to a released version of the tool.

- tags ‘v1.0.1’ thru ‘v2.3.0’ denote versions relating to the ArcGIS/MapInfo variant master branch
• tags ‘v1.0.1m’ thru ‘v2.3.0m’ denote versions relating to the MapInfo only variant master-mapinfo branch.

4.1.4 Releases

In addition to the source code tags, each variant/version of the tool is listed under HLUTool Releases. Each release relates to one of the above tags, but in addition contains a set of Release Notes together with a download link to a Zip copy of the source code and the Windows Installer setup.exe program for that version. Links to the release for each new version can be sent to users of the tool for them to install.

4.1.5 Collaborating

Currently there are two owners of the HLUTool repository on GitHub, only one of whom is a developer. There has therefore been no need to fork the repository or raise Pull requests. If, in the future, a new developer is to take over development/maintenance of the tool then the simplest solution is for them to also become an owner of the repository so that they have control over source. However, if there is a need to have more than one developer then it may be necessary to agree on a collaborative source control process. According to GitHub there are two popular models of collaborative development:

Fork & Pull

The fork & pull model lets anyone fork an existing repository and push changes to their personal fork without requiring access be granted to the source repository. The changes must then be pulled into the source repository by the project maintainer (owner). This model reduces the amount of friction for new contributors and is popular with open source projects because it allows people to work independently without upfront coordination.

Shared repository model

The shared repository model is more prevalent with small teams collaborating on projects. Everyone is granted push access to the central, shared repository and topic branches are used to isolate changes.

Pull requests are especially useful in the fork & pull model because they provide a way to notify project maintainers about changes in your fork. However, they’re also useful in the shared repository model where they’re used to initiate code review and general discussion about a set of changes before being merged into a mainline branch.
4.2 Helpful Links

For those unfamiliar with Git and GitHub the following links will provide some useful reading material to help explain what Git is and how to use GitHub.

**The Git Parable**  Tom Preston-Warner, a co-founder of GitHub, wrote a long but very informative post *The Git Parable* on his blog that is really worth reading as an introduction to the origins and concepts of Git.

**Pro Git**  Written by Scott Chacon, the entire open source book on learning and using Git is available to read online for free or to purchase as a book.

**Scott Chacon blog**  Scott Chacon, a software developer at GitHub and author of Pro Git, wrote a few posts on his blog that might be useful, especially the last post GitHub Flow.

**Git Reference**  This handy site is great as a glossary reference if you know how to use Git but are always forgetting the commands.

**GitHub Guides**  A series of guides and videos for understanding and using GitHub are available at GitHub Guides.
5.1 General Guidelines

The following are a set of ‘basic principles’, based on those recommended by Bugzilla, that are also relevant to this Developer’s Guide:

- These guidelines exist in order to reduce bugs and make it easier to change things in the future.
- Making it easier to change things is important because an important law of code is: All Code Will Change.
- Code should be as simple as possible. When writing code one of the primary goals should be: ‘make it easy for other programmers to use and read your code’.
- Readable code is more important than clever code.
- If you’re trying to be clever instead of trying to be readable, then maybe you’re trying to make things ‘faster’? But remember: don’t solve a problem before you know it exists.

Tip: If you don’t know (by actual, detailed tests) that your code is slow, don’t worry about making it faster. This isn’t just limited to optimization - many programmers are constantly solving problems that nobody has ever experienced. Don’t do that.

- You cannot introduce new bugs unless you change code (obviously). This means:
  - The more code you change, the more bugs you will create. There are no ‘perfect’ programmers. Everybody makes a few mistakes now and then.
  - The number of bugs introduced by a patch is proportional to the size of the patch.
  - Hence, patches should be as small as possible, and should change only what they need to change.
- An easy way to check if your code is readable, is to ask yourself, ‘Will another programmer understand this line instantly when they look at it? If not, that line either needs to be re-written or needs a comment. See Comments (page 20) for more guidelines on applying comments.
5.2 Comments

5.2.1 XML comments

Many of the existing methods and properties in source code are missing XML documentation comments (e.g. <summary>, <param>, <returns>, <value>, etc.), or the comments are out-of-date. When adding, and whenever possible when amending code, appropriate and comprehensive XML tags and comments should be added or amended to ensure they are up-to-date and sufficiently describe the section of code. The following is a good example:

```csharp
/// <summary>
/// Creates a vague date string of a specified format from start and end dates
/// expressed as days elapsed since 30/12/1899.
/// </summary>
/// <param name="startDateDays">Start date in days since 30/12/1899.</param>
/// <param name="endDateDays">End date in days since 30/12/1899.</param>
/// <param name="dateType">Two-letter date format code as generated by the GetFormat method.</param>
/// <param name="outputFormat">One of the output formats in the HLU.Date.DateType enumeration.</param>
/// <returns>A vague date string in the format specified in the dateType parameter.</returns>
public static string FromTimeSpanDays(int startDateDays, int endDateDays, string dateType, DateType outputFormat)
```

5.2.2 General comments

As well as adding and maintaining XML documents at the start of each class, method or property, additional comments should be added to sections or lines of code that may need further explanation to be understood.

Although there are almost infinite opportunities to refactor and simplify code to obviate the need for comments, explaining yourself exclusively in code has its limits. No matter how simple, concise, and clear your code may end up being, it’s impossible for code to be completely self-documenting. Comments can never be replaced by code alone.

Here are a few rules of when and why comments should be added include:

1. Make sure you explain where and why you have changed the code to help debug potential introduced bugs. See also Comment tasks (page 21).

2. Explain why you chose to do things one way rather than another, especially if the chosen approach is not obvious. The example below not only names the technique used, but also explains why a simpler approach was not taken:

   /* A binary search turned out to be slower than the Boyer-Moore algorithm for the data sets of interest, thus we have used the more complex, but faster method even though this problem does not at first seem amenable to a string search technique. */

3. Don’t just explain what the code is doing something but why the program is doing it. It may be obvious what the code is doing, but not why.

4. Explain how any complex sections of code that cannot be refactored or simplified work and why.

   Tip: This is especially important for existing code - it is safer to document what the code does (once you’ve figured it out) than refactor the code and risk introducing an
5. Any algorithms (calculations, logic flows, etc.) used in the program, no matter how simple they may seem when first written, should be explained.

6. Finally, keep in mind that what seems obvious now may not seem obvious later.

Note: Remember: Code can only tell you how the program works; comments can tell you why it works.

5.2.3 Comment tasks

Comment tasks have been added at the start of each section of code that relates to a Known Issue, Change Request or Fix. Comment tasks are comments that begin with a comment task token and should be formatted as:

```
// TOKEN: reference (description)
```

where:
- `TOKEN` is one of the tokens listed below
- `reference` is a Known Issue or Change Request reference (where applicable)
- `description` is a brief summary of the comment

Visual Studio can be configured so that these comments automatically appear in the Task List window. To configure Visual Studio go to Tools –> Options..., click on the Environment list heading the left to expand the list and click on Task List. Add the following Tokens to the Token List on the right:

- FIXED: Used to indicate where a Known Issue has been fixed
- CHANGED: Used to indicate where changes relating to a Change Request have been applied
- FIX: Used to indicate where a previously unknown issue (e.g. identified during coding/testing) has been fixed
- HACK: Used to indicate when a quick ‘Hack’ has been applied to temporarily resolve a previously unknown issue
- QUERY: Used to indicate where code (possibly relating to a Known Issue or Change Request) may need to be amended/corrected
- TODO: Used to indicate where work relating to a change or fix remains outstanding

Where possible top & tail comment borders should be inserted around the ‘Task List’ comment and related source code to denote where the change/fix/query starts and stops. Additional ‘explanatory’ comments should also be added to explain what the amended code does, or why it was amended. For example:

```
//---------------------------------------------------------------------
// FIXED: KI96 (BAP Habitats)
// Enable editing of bap habitats when they are only associated
// with matrix, formation, management or complex codes (rather
// than habitat codes.
OnPropertyChanged("BapHabitatsAutoEnabled");
//---------------------------------------------------------------------
```

5.2. Comments
The same Comment task can be inserted in multiple locations in the source code if more than one section of code relates to the same change/fix/query. However, the ‘explanatory’ comments should be bespoke for the specifically amended code.
CHAPTER 6

Windows Installer

Since v1.0.1 of the HLU Tool InstallShield Limited Edition has been used to build the Windows Installer for deploying the tool on other systems. InstallShield Limited Edition is a free version of Flexera’s InstallShield included within Visual Studio 2010 (and later) that replaces the functionality previously provided by the Visual Studio Installer Setup Projects. It contains a wide range of features and options for configuring how a Windows application will be installed, many of which are not needed for installing the tool.

**Note:** To create a setup.exe file you need to build the HluSetup_ISLE project using the configuration option ‘SingleImage’.

Below is a summary of the InstallShield Limited Edition features and configuration options that are used for creating a Windows installer for the tool. There are no complex settings used to build a Windows installer for the HLU Tool. In fact, the only ‘advanced’ requirement when installing the ArcGIS/MapInfo variant of the tool is the need to register the HluArcMapExtension.dll assembly as an ArcGIS Desktop extension. This is not ‘directly’ achieved by the installer setup.exe application, but instead is achieved ‘indirectly’ using an Installer Class assembly.

**See also:**
See ArcGIS Registration (page 31) for details of how the ArcGIS Extension is registered when building the tool assemblies in Visual Studio and when installing the tool.

### 6.1 Organize Your Setup

#### 6.1.1 General Information

This tab contains general information about the tool that appears when viewing the properties of the tool once installed and when viewing the list of installed programs in the control panel. More importantly it also contains the product version, product code and upgrade code and the destination installation path.

**See also:**
See Version Numbers (page 27) and Product Code (page 28) for more details.
6.1.2 Upgrade Paths

This tab allows the developer to control which previously installed versions of the tool can be upgraded using this version of the tool. During installation the Windows Installer searches the target system for the specified upgrade code. If found, and the other upgrade properties are met, the target system is upgraded by installing the new version. If the upgrade code is not found on the target system then this version of the tool will be installed as new.

See also:

See Upgrading (page 28) for more details.

The ISPreventDowngrade has also been enabled to prevent the current version of the tool from over-writing later (future) versions of the tool. If users want to install an older version of the tool once a newer version has been installed then the newer version will first need to be manually uninstalled.

6.2 Specify Application Data

6.2.1 Files

This tab is used to specify which files are to be included in the install which folders they will be copied to the target system. The files specified include:

- The **primary output** files created during the build process (.dll and .exe files).
- Any .dll files referenced by the tool and not ordinarily available on a target system.
- An **icon** for the tool.
- A **ReadMe.txt** file giving an overview of the tool, it’s features and where more information can be found.
- A **Licence.txt** file containing details of the GNU General Public License under which the tool is released.
- An empty MapInfo workspace **Empty.wor** used to open MapInfo when first launching the tool.

6.3 Configure the Target System

6.3.1 Shortcuts/Folders

This tab offers a method of designing shortcuts and program folders for the tool. Currently only two shortcuts are created on the target system’s Programs menu:

- **HLU GIS Tool - Launch** This shortcut starts the tool without any optional arguments (normal operation)
- **HLU GIS Tool - Reconfigure** This shortcut starts the tool with the optional ‘/c’ argument which triggers the tool to clear any existing configuration settings and re-run the initial configuration steps.
6.4 Customize the Setup Appearance

6.4.1 Dialogs

This tab allows the dialog pages of the tool’s setup steps to be customised. Developers can configure various features such as which setup dialogs appear, what images and options appear on the dialogs and if the tool is automatically launched after installation.

6.5 Define Setup Requirements and Actions

6.5.1 Requirements

This tab is where you can configure software conditions that must be met on a target system in order for the installation setup for the tool to run. Currently there are only two conditions set:

.NET 3.5 SP1 is installed  Microsoft .NET Framework 3.5 Service Pack 1 (or later) must be installed on the target system.

REALVERSION  A custom condition that ensures that ArcGIS 10.1 or later is installed on the target system by checking the existence of a ‘RealVersion’ registry value under the registry key ‘SOFTWARE\ESRI\ArcGIS’ in the ‘HKLM’ registry root. This is a custom condition that was created using the System Search Wizard.

Note: This custom condition is only included in the ArcGIS/MapInfo variant of the tool as a simple mechanism to ensure it is not installed on a MapInfo only target system in error.
7.1 Building the Tool

When building the tool for a new version/release, rather than for just testing or debugging changes in progress, there are a number of steps to follow or dependencies to consider.

7.1.1 Version Numbers

Assembly version

The HLUGISTool assembly version, using the format Major.Minor.Patch.Build, should be incremented following semantic versioning rules. So whether the increment relates to a major change, minor update or just a patch will depend on what is contained in the new release.

- Major version numbers change whenever there is significant change to the look or functionality or for large or potentially backward-incompatible changes.
- Minor version numbers change when a new minor feature is introduced, or when a set of smaller features are rolled out together.
- Patch numbers change when a new build of the software is released containing small bug fixes.
- Build numbers typically don’t change as a new version is not usually released just for a new build.

**Note:** The assembly version number appears in the ‘About’ pop-up window on user interface.

Product version

The product version in the installer project properties should also be changed to match the assembly version number.

**Note:** The installer product version number is used when installing the tool to ensure an older version of the tool doesn’t overwrite a later version. The version number also appears in the Control Panel Programs and Features list.
7.1.2 Product Code

The installer **Product Code** should be changed to uniquely identify this version of the product. The Windows installer uses the product code at run time to determine whether the same version of the product has already been installed.

**Note:** To create a new GUID that uniquely identifies a new version of the product or click the *Generate a new GUID button* ( {...} ) in the Product Code setting in the General Information tab of the InstallShield project.

A note of all previously used Product Codes is maintained in ‘Releases and Product Codes.txt’ in the source code repository on GitHub.

7.1.3 Upgrading

The installer **Upgrade From Max Version** must reflect the version number of the most recent release so that the Windows installer will find and upgrade all previous of the tool.

7.1.4 ReadMe files

The ReadMe file must be amended to reflect the version number and copyright details of the new release, as well as any new features or changes to system requirements. The ReadMe file is maintained in three different formats; simple text (.txt), rich text (.rtf) and markdown (.md). Each format is used in different circumstances:

- **ReadMe.txt** - Installed with the tool on the target system.
- **ReadMe.rtf** - Displayed during the installation process.
- **ReadMe.md** - Displayed in the source code repository by GitHub.
7.2 Distributing the Release

The tool is currently distributed via GitHub. There are a number of stages involved in distributing a new release of the tool.

7.2.1 GitHub Tags

Once the final commit has been applied for the new version then new tags should be created in the local Git repository for each branch/variant of the tool. It is common practice to use tag names by prefixing the version number with the letter v. For the tool tag descriptions also follow a set pattern by explicitly stating if it is a major, minor or patch release.

**ArcGIS/MapInfo variant** Name: version number prefixed by ‘v’ (e.g. ‘v1.0.8’) Description: Major/Minor/Patch release version number for ArcGIS/MapInfo (e.g. *Minor release v1.0.8 for ArcGIS/MapInfo*)

```
Note: To create the above tag example enter the following in a Git shell whilst the master branch is active:
git tag -a v1.0.8 -m 'Minor release v1.0.8 for ArcGIS/MapInfo'
```

**MapInfo variant** Name: version number prefixed by ‘v’ and suffixed by ‘m’ (e.g. ‘v1.0.8m’) Description: Major/Minor/Patch release version number for MapInfo only (e.g. *Minor release v1.0.8 for MapInfo only*)

```
Note: To create the above tag example enter the following in a Git shell window whilst the master-mapinfo branch is active:
git tag -a v1.0.8m -m 'Minor release v1.0.8 for Mapinfo only'
```

Once the tags have been created in the local repository they should be pushed to the remote GitHub repository.

```
Note: To push new tags to GitHub enter the following in a Git shell window:
git push --tags
```

**Tip:** Existing tags for the tool can be viewed on GitHub under HLUTool Tags.

7.2.2 Release Notes

Each new version/variant of the tool should be accompanied by its own set of release notes. Release notes are written using GitHub Flavored Markdown and should contain the following information as a minimum:

- Version
Once the new tags for each branch/variant have been pushed to the GitHub repository then release notes can be added. To add release notes go to the list of HLUTool Tags and click Add release notes against the required tag.

**Tip:** Existing release for the tool can be viewed on GitHub under HLUTool Releases.

### 7.2.3 Upload Executables

Finally, once each new release has been created on GitHub the associated installer setup.exe executable can be uploaded. This provides an effective way of distributing the tool and ensures that the installer is stored alongside the relevant release notes and source code for each version/variant.

**Note:** To attach the setup.exe installer to a release, edit the release on GitHub and then ‘drag and drop’ the file on the Attach binaries by dropping them here area.
Prior to ArcGIS version 10, Component Object Model (COM) components must be registered with the system before they can be used by ArcGIS. This was typically accomplished using Microsoft’s supplied utility `RegAsm.exe`.

Starting with ArcGIS version 10, however, ESRI has moved away from the COM component category approach and there is now a new ESRI registration utility `ESRIRegAsm.exe` which works independently of the system registry; registration of component information for an assembly is now achieved using this new utility.

COM components must also be registered in the ESRI component categories appropriate to their intended context and function in order for the ArcGIS applications to make use of their functionality. For example, all ArcMap extensions must be registered in the ESRI `MxExtensions` component category.

### 8.1 Registration During Build

Specifying the ‘Register for COM Interop’ option on an assembly will trigger Visual Studio to execute `RegAsm.exe` with the `/codebase` argument during the `Build` and `Clean` processes to register/unregister the assembly as a COM component in the registry of the development machine. Whenever a component is being registered or unregistered for use from COM, two attribute classes within the .NET Framework, `ComRegisterFunctionAttribute` and `ComUnregisterFunctionAttribute`, allow you to specify user-defined methods that will be called automatically.

Therefore, by specifying the ‘Register for COM Interop’ option on the `HluArcMapExtension` assembly the methods `RegisterFunction` and `UnregisterFunction` will automatically be called. These contains user-written code to register and unregister the assembly as an ArcGIS extension in the `MxExtensions` component category (see below):

```csharp
[ComRegisterFunction()]
[ComVisible(false)]
static void RegisterFunction(Type registerType)
{
    // Required for ArcGIS Component Category Registrar support
    ArcGISCategoryRegistration(registerType);
    #if ARC10
    #else
        EsriRegasm(true, registerType);
    #endif
}

[ComUnregisterFunction()]
```
```csharp
[ComVisible(false)]
static void UnregisterFunction(Type registerType)
{
    // Required for ArcGIS Component Category Registrar support
    ArcGISCategoryUnregistration(registerType);
    #if ARC10
    #else
        EsriRegasm(false, registerType);
    #endif
}

/// <summary>
/// Required method for ArcGIS Component Category registration -
/// Do not modify the contents of this method with the code editor.
/// </summary>
private static void ArcGISCategoryRegistration(Type registerType)
{
    string regKey = string.Format("HKEY_CLASSES_ROOT\CLSID\{{{0}}}", registerType.GUID);
    MxExtension.Register(regKey);
}

/// <summary>
/// Required method for ArcGIS Component Category unregistration -
/// Do not modify the contents of this method with the code editor.
/// </summary>
private static void ArcGISCategoryUnregistration(Type registerType)
{
    string regKey = string.Format("HKEY_CLASSES_ROOT\CLSID\{{{0}}}", registerType.GUID);
    MxExtension.Unregister(regKey);
}
```

**Note:** The boolean variable `ARC10` used in the above methods is a compiler *directive* indicating to Visual Basic if the assembly is being compiled for a version of ArcGIS 10 or later or not.

In order to trigger the ESRIRegAsm.exe utility (for ArcGIS 10 onwards) custom calls have also been added to the **HluArcMapExtension.csproj** Visual Studio project file (see below):

```xml
<Target Name="BeforeClean">
    <Exec WorkingDirectory="$(CommonProgramFiles)\ArcGIS\bin" Command="esriRegasm.exe &quot;$(TargetPath)&quot; /p:Desktop /u" Condition="Exists('$(TargetPath)')" />
</Target>

<Target Name="AfterBuild">
    <Exec WorkingDirectory="$(CommonProgramFiles)\ArcGIS\bin" Command="esriRegasm.exe &quot;$(TargetPath)&quot; /p:Desktop" />
</Target>
```

These custom calls execute the ESRIRegAsm utility using command line argument `'/p'` to specify the ArcGIS ‘Desktop’ product. Because the `'/s'` argument is not supplied a dialog box will appear indicating if the registration/unregistration is successful or not.
8.2 Registration During Installation

During the installation of the HLU Tool the HluArcMapExtension assembly must also be registered using the ESRIRegAsm utility and in the MxExtensions component category on the target machine. Because the Visual Studio Build process is not run registration is achieved in a different way than as described above.

Firstly, a custom Installer class assembly ArcObjectsInstaller is included within the installer. The installer class is recognised by the Windows installer which can instantiate the class and call various methods, including the methods Install and Uninstall which are executed when an install/uninstall is performed.

Therefore, when running the setup.exe Windows installer on a target machine the ArcObjectsInstaller assembly is installed and the Install method is executed which performs the following:

- It registers the HluArcMapExtension assembly with COM
- It registers the assembly in the appropriate ESRI component category MxExtension
- It executes the ESRIRegAsm utility (if ArcGIS 10 onwards is installed) to register the assembly information for use by ArcGIS.
There are three distinct guides that accompany the HLU Tool. All the guides are written using reStructuredText, their sources are stored on GitHub and they are available to view and download on ReadTheDocs.

**User Guide**

The user guide, available to view or download on ReadTheDocs at HLUTool-UserGuide, is for those who will be regular users of the HLU Tool but are not concerned with how to install or configure the tool or how to perform database administration.

**Technical Guide**

The technical guide, available to view or download on ReadTheDocs at HLUTool-TechnicalGuide, is for those of a more ‘techie’ nature. It contains details of how to install, configure, maintain and upgrade the HLU Tool and use and maintain the associated relational database.

**Developer’s Guide**

The developer’s guide (this guide), available to view or download on ReadTheDocs at HLUTool-DevelopersGuide, explains a little about how the HLU Tool source code is structured, what you may need to develop and support the tool and how to build and distribute it.
9.1 reStructuredText

9.1.1 What is it?

reStructuredText is an easy-to-read, what-you-see-is-what-you-get plain text markup syntax. It is often used for in-line program documentation (such as Python docstrings), for quickly creating simple web pages, and for standalone documents (such as the HLU Tool guides).

reStructuredText can be written in any text editor, but some editors provide syntax highlighting and shortcuts to assist authors (see Tools & Extensions (page 9) for some examples).

9.1.2 Further reading

For those unfamiliar with reStructuredText the following websites will provide some useful background reading material and references to help in authoring documents:

- **Sphinx reStructuredText** - an introduction to reStructuredText concepts and syntax.
- **Docutils** - a reference guide for reStructuredText markup syntax.
9.2 GitHub

9.2.1 Source Control

The source code for the documentation is stored on GitHub with each guide in a separate repository. Source control for the guides works in the same way as for the tool itself using Git and GitHub for Windows (see *GitHub Source Control* (page 15)).

**Tip:** See HabitatFramework for a list of all the repositories on GitHub relating to the tool.

One of the benefits of reStructuredText is that documents written using it are readable in their ‘raw’ markup format. So the source code files for the guides can be viewed and downloaded directly using GitHub. However, the intended purpose of the markup is the conversion of reStructuredText documents into more structured data formats; that’s where ReadTheDocs comes in (see *ReadTheDocs* (page 38) below).

9.2.2 Webhooks

The really clever trick is that once a change to a guide has been committed to GitHub, a **Webhook** notifies ReadTheDocs of the change. ReadTheDocs will then rebuild the documentation using the latest source of the documents.

**Note:** Webhooks allow external services such as ReadTheDocs to be notified when certain events happen on GitHub. When the specified events happen, such as a commit, GitHub sends a *POST* request to each of the specified URLs. The target system can then pull in the latest source and perform an action, such as rebuilding the documentation.
9.3 ReadTheDocs

9.3.1 What is it?

ReadTheDocs is an online documentation repository for the open source community. It supports Sphinx docs written with reStructuredText. Sphinx is a documentation generator which converts reStructuredText files into HTML websites and other formats including PDF. ReadTheDocs automates the process of building and uploading Sphinx documentation.

9.3.2 Building

By using a GitHub Webhook, ReadTheDocs will be ‘pinged’ when the source has been updated. ReadTheDocs will then rebuild the documentation using the latest source documents.

When each ReadTheDocs project (each guide is a separate project) is built it automatically builds separate HTML and PDF formats of the documentation. This provides users with alternative methods of viewing the guides, each with its own strengths and weaknesses.

9.3.3 Versions

ReadTheDocs supports multiple versions for each project, so for each release of the tool it can host a parallel release of each of the guides. To do this each guide would need to be updated (where appropriate) and then ‘tagged’ in GitHub (see GitHub Source Control (page 15)). ReadTheDocs will then build HTML and PDF formats of the guide for the new version and continue to host this latest version together with all previous versions.

Tip: Which versions are available to users on ReadTheDocs can be configured on the Versions page in the Admin section for each project (guide).

9.3.4 Further reading

The ReadTheDocs documentation provides an introduction to those unfamiliar with ReadTheDocs features and explains the build process.
10.1 ALERC Forum

The ALERC forum is available for open discussions by users about the tool. Users should post items on the forum in the following situations:

- When they experience any new issues unrelated to testing a specific release of the tool.
- If they want to recommend any new changes or enhancements to the tool.
- If they have any general queries relating to the use or functionality of the tool.

Posting new issues and proposed enhancements to the tool on the forum allows all users to consider and discuss the proposal. Once a consensus has been reached on a change or solution to an issue it can be added to the HLUTool issues log on GitHub (see GitHub Issues Log (page 39) for more details). It can then be prioritised and scheduled to be addressed when time and funding allows.

The ALERC forum is not intended for use during any scheduled development or maintenance phases. This is because planned changes or fixes are easier to find and track using GitHub.

10.2 GitHub Issues Log

The GitHub HLUTool issues log is where changes and fixes to the tool should be logged once they have been agreed by the users. When time and funding then allows they can be prioritised and scheduled within an upcoming project. They can then be assigned to a project milestone (and if necessary a specific developer).

Once the project is under way the issues log should be used in the following situations:

- If users have any queries about a planned change or fix scheduled to be addressed in an upcoming release of the tool.
- If users experience any problems testing any of the changes or fixes addressed in a new release of the tool.
- If users have successfully tested a scheduled change or fix.

Queries, problems and test results should all be added as comments to the relevant change/fix in the issues log. Users will have to register on GitHub and log in to be able to post comments.
In order to apply structural and data changes to the HLU tool database you will need to use the HLU Tool Database Updater `HLUDbUpdater.exe`. The HLU Tool Database Updater provides an automated mechanism of applying changes to a target HLU Tool relational database. It will process one or more script files and execute all the SQL commands in the files.

### 11.1 Components

Within the Visual Studio solution for the HLU Database Updater there is a single project containing a single .NET assembly.

**HLUDbUpdater** This is the main assembly that contains all of the user interfaces, the ‘business logic’ and also handles the data connection with the chosen relational database. There are a few ad-hoc classes in the parent folder but the majority of the source code is structured in the same sub-folders as the main HLU GIS Tool project (see *HLUGISTool* (page 11) for more details).

The majority of the user interfaces and supporting classes relate to connecting the updater to the required relational database. Most of these are exact copies of the same components used in the main tool assembly but a few have been adapted to provide additional functionality specific to the database updater. There is no interaction with GIS or the spatial data so the components relating to the ArcGIS and MapInfo applications used by the main tool assembly are not included.
11.2 Source Control

11.2.1 GitHub

Like the main tool the source code for the database updater is open source and hosted by GitHub. It can be downloaded from HLUDbUpdater repository.

- **Branches**

  There are two main branches in the repository:
  
  - **master** contains the source code for the database updater
  
  - **scripts** contains all of the SQL scripts to be applied by the database updater

- **Tags**

  The source code for every version of the database updater source code from v1.0.0 to v1.0.1 is Tagged on GitHub. The HLUDbUpdater Tags point to a specific ‘commit’ in a branch to indicate that the commit relates to a released version of the tool.

- **Releases**

  In addition to the source code tags each release of the database updater is also listed under HLUDbUpdater Releases. Each release relates to one of the above tags but in addition contains a set of Release Notes together with a download link to a Zip copy of the source code and the executable HLUDbUpdater.exe for that version.

- **Scripts**

  All of the latest scripts for the database updater can be downloaded from [GitHub Archive](https://github.com/HabitatFramework/HLUTool-DatabaseUpdater/archive/).
11.3 Building Releases

Building the database updater for a new version/release is more straightforward than building the main tool. The database updater does not need to be installed in order to be executed and hence it does not require an installer. There are just a few steps to consider.

11.3.1 Version Number

The HLUDbUpdater assembly version, using the format Major.Minor.Patch.Build, should be incremented following semantic versioning rules. So whether the increment relates to a major change, minor update or just a patch will depend on what is contained in the new release.

- Major version numbers change whenever there is significant change to the look or functionality or for large or potentially backward-incompatible changes.
- Minor version numbers change when a new minor feature is introduced, or when a set of smaller features are rolled out together.
- Patch numbers change when a new build of the software is released containing small bug fixes.
- Build numbers typically don’t change as a new version is not usually released just for a new build.

Note: The database updater version number appears in the user interface title bar.

11.3.2 ReadMe File

The ReadMe.txt file must be amended to reflect the version number and copyright details of the new release, as well as any new features or changes to system requirements. The ReadMe file is a simple text (.txt) file which is distributed with the database updater executable HLUDbUpdater.exe.
11.4 Distribution

Like the main tool, the database updater is currently distributed via GitHub. There are a number of stages involved in distributing a new release:

11.4.1 GitHub Tags

Once the final commit has been applied for a new version then a new tag should be created in the local Git repository for the master branch. It is common practice to use tag names by prefixing the version number with the letter v. The tag descriptions also follow a set pattern by explicitly stating if it is a major, minor or patch release.

- **Name**: version number prefixed by ‘v’ (e.g. ‘v1.0.1’)
- **Description**: Major/Minor/Patch release version number (e.g. *Minor release v1.0.1*)

**Note:** To create the above tag example enter the following in a Git shell whilst the master branch is active:

```sh
git tag -a v1.0.1 -m 'Minor release v1.0.1'
```

Once the tags have been created in the local repository they should be pushed to the remote GitHub repository.

**Note:** To push new tags to GitHub enter the following in a Git shell window:

```sh
git push --tags
```

**Note:** The database updater script branch does not require tags because scripts do not necessarily relate to specific versions of the database updater or the main tool.

**Tip:** Existing tags for the database updater can be viewed on GitHub under HLUTool Tags.

11.4.2 Release Notes

Each new release of the database updater should be accompanied by its own set of release notes. Release notes are written using GitHub Flavored Markdown and should contain the following information as a minimum:

- Version
- Release date
- System requirements
- Execution Instructions
- Additions
• Removals
• Changes
• Fixes

Once the new tag for a release has been pushed to the GitHub repository then release notes can be added. To add release notes go to the list of HLUDbUpdater Tags and click Add release notes against the required tag.

Tip: Existing release for the database updater can be viewed on GitHub under HLUTool Releases.

11.4.3 Executables

Finally, once a new release has been created on GitHub the HLUDbUpdater.exe executable and associated files (e.g. ReadMe.txt, Licence.txt and .dlls) can be uploaded. This provides an effective way of distributing the database updater and ensures that it is stored alongside the relevant release notes and source code for each release.

Note: To attach the executable and associated files to a release combine them all into a single .zip file, edit the release on GitHub and then ‘drag and drop’ the .zip file on the Attach binaries by dropping them here area.
11.5 Scripts

The scripts processed by the database updater contain one or more SQL statements designed to update the structure and/or contents of an HLU Tool relational database. Each script file must adopt the following rules in order to be valid and be processed by the database updater program.

11.5.1 File Names

Script files (e.g. ‘0000B.sql’) must be named sequentially using Base36 (e.g. 0 to 9 then A to Z, 10 to 19 then 1A to 1Z, etc.)

If a script file is found that has already been processed then it will be skipped and moved to the Archive sub-folder. If a script file is missing from the Base36 sequence then an error will appear and processing will stop.

11.5.2 SQL Commands

Each SQL command must meet the following rules:

- Each SQL command must fit on a single line - multi-line commands will be split at line ends
- Comments are delimited using the prefix/suffix /* and */, e.g.
  /* Delete the existing exports_fields row. */
- String values are delimited by single quotes '', e.g.
  INSERT INTO [exports] (export_id, export_name) VALUES (1, 'All attribute fields')
- Database table names are delimited by square brackets [], e.g.
  DELETE * FROM [exports]
- INSERT commands must explicitly include the INTO keyword, e.g.
  INSERT INTO [lut_user] ...  

Note:

- Single quotes within strings are not currently supported (e.g. ‘White’s House’)
- Double quotes within strings are not currently supported (e.g. ‘White House “North”’)

11.5.3 Connection Type Directives

Specific connection types or databases can be targeted by specifying the required connection types/database in a comma-delimited list within square brackets [] on a separate line, e.g.

[Access,SqlServer,PostGreSql,Oracle]

Connection type directives are required when the structure or keywords of a SQL command are different between connection types or databases - for example Access uses the function ‘UCASE’ to convert strings to upper case whereas SQLServer, Oracle and PostgreSQL use the function ‘UPPER’.
Once a connection type directive has been specified in a script all subsequent SQL commands in the script will only be applied if the actual connection type or database established by the user is found in the comma-delimited list until either:

- Another specific connection type directive is encountered, or
- The connection type is reset using the [All] or [Any] directive

### 11.5.4 Special Commands

Scripts can contain a number of special commands unique to the database updater:

**Set Ignore_Errors**

- Set **On** to ignore any errors in subsequent SQL commands, i.e.
  
  Set Ignore_Errors On

- Set **Off** to immediately stop a script if any errors occur processing subsequent SQL commands, i.e.
  
  Set Ignore_Errors Off

**Set Timeout**

- To override the default timeout specify the number of seconds before a database timeout will occur when processing a single SQL command, e.g.
  
  Set timeout 120

- To reset the default timeout specify:
  
  Set timeout default or Set timeout

**Set Display_Results**

- Set **On** to display the results of any subsequent SQL commands, i.e.
  
  Set display_results on

- Set **Off** to hide the results of all subsequent SQL commands, i.e.
  
  Set display_results off

**Set Skip_Version_Update**

- Set **On** to skip updating the database version in the lut_version table, i.e.
  
  Set skip_version_update on

- Set **Off** to ensure the database version in the lut_version table is updated (as default), i.e.
  
  Set skip_version_update off
12.1 GNU Free Documentation License

GNU Free Documentation License
Version 1.3, 3 November 2008

<http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below,
refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not Transparent is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.
The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit
legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one.
stating the title, year, authors, and publisher of the Document as
given on its Title Page, then add an item describing the Modified
Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for
public access to a Transparent copy of the Document, and likewise
the network locations given in the Document for previous versions
it was based on. These may be placed in the "History" section.
You may omit a network location for a work that was published at
least four years before the Document itself, or if the original
publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications",
Preserve the Title of the section, and preserve in the section all
the substance and tone of each of the contributor acknowledgements
and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document,
unaltered in their text and in their titles. Section numbers
or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section
may not be included in the Modified Version.

N. Do not retile any existing section to be Entitled "Endorsements"
or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or
appendices that qualify as Secondary Sections and contain no material
copied from the Document, you may at your option designate some or all
of these sections as invariant. To do this, add their titles to the
list of Invariant Sections in the Modified Version's license notice.
These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains
nothing but endorsements of your Modified Version by various
parties--for example, statements of peer review or that the text has
been approved by an organization as the authoritative definition of a
standard.

You may add a passage of up to five words as a Front-Cover Text, and a
passage of up to 25 words as a Back-Cover Text, to the end of the list
of Cover Texts in the Modified Version. Only one passage of
Front-Cover Text and one of Back-Cover Text may be added by (or
through arrangements made by) any one entity. If the Document already
includes a cover text for the same cover, previously added by you or
by arrangement made by the same entity you are acting on behalf of,
you may not add another; but you may replace the old one, on explicit
permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License
give permission to use their names for publicity for or to assert or
imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this
License, under the terms defined in section 4 above for modified
versions, provided that you include in the combination all of the
Invariant Sections of all of the original documents, unmodified, and
list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION
Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this license or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this
License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive MultiAuthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive MultiAuthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDITIONAL: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (c) YEAR YOUR NAME.
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:
If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.
Index

A
Acknowledgements, 4
ArcGIS Registration, 30
Assemblies, 11
  ArcObjectsInstaller, 13
  HluArcMapExtension, 12
  HLUDbUpdater, 41
  HLUGISTool, 11
  InterProcessComm, 12
  Named Pipes, 12

C
Coding Standards
  Comment Tasks, 21
  General Comments, 20
  General Guidelines, 19
  XML Comments, 20
Components
  Installer, 13
  Named Pipes, 14

D
Database Updater, 41
  New Releases, 43
  Scripts, 46
  Source Control, 42
Documentation, 33
  GitHub, 37
  ReadTheDocs, 38
  reStructuredText, 36

L
Licensing, 3

N
New Releases
  Database Updater, 43
  Executables, 30
  Product Code, 27
  ReadMe Files, 28

  Release Notes, 29
  Tags, 29
  Upgrading, 28
  Version Numbers, 27

Q
Quick Links, 4

R
Releasing, 29
Requirements
  Applications, 8
  Skills, 7
  Tools & Extensions, 9
  Websites, 9

S
Source Control
  Database Updater, 42
  GitHub, 15
  Helpful Links, 17

W
Windows Installer, 22