

LimsLink EI™

Version 4.2

(includes LL EI v4.2 Installation and Configuration Manual, LL EI v4.2 User Manual and CDS and LIMS Module Inserts)

LimsLink EI v4.2

Copyright 1993-2011 Labtronics Inc.

Printed in Canada.

Windows® is a registered trademark of Microsoft® Corporation.

Microsoft SQL Server® and Microsoft SQL Server Desktop Engine® are registered trademarks of Microsoft® Corporation.

Oracle® is a registered trademark of Oracle Corporation.

The Nexxis iLAB product uses WinWrap Basic®, Copyright 1993-2009 Polar Engineering and Consulting,
<http://www.winwrap.com/>.

Publication History

FIRST EDITION March 2011

Customer Support:

North America

Phone: (519) 767-1061

Fax: (519) 836-4431

E-mail: support@labtronics.com

LimsLink EI™

Version 4.2

Installation and Configuration Manual





Master Software License Agreement

This Master Software License Agreement ("Agreement") is effective as soon as you (LICENSEE) install a Licensed Software supplied by Labtronics Inc (LICENSOR) either directly or through one of their agents. By installing a product from the LICENSOR, the LICENSEE agrees to the conditions of this Agreement.

This Agreement supersedes all prior Master Software License Agreements from Labtronics. Purchase orders previously entered into by and between the parties, which incorporate prior software license agreements, shall henceforth be interpreted as incorporating this Agreement. Where there is a conflict between such Agreements, the Agreement containing signatures will take precedence.

Date of this Agreement is December 1, 2009

ARTICLE I – DEFINITIONS, LICENSE GRANT

The term "Licensed Software" shall mean any and all computer programs, in whatever form, and any Upgrades, (defined below) and Licensed Material (defined below) related to the computer programs.

The term "Upgrades" shall mean modified computer programs, or Licensed Material as furnished by LICENSOR from time to time. Modifications include new releases, enhancements, error corrections and field fixes.

The term "Licensed Material" shall mean documentation in hard copy and or electronic format that is supplied in support of the Licensed Software.

- 1.1 **License Grant.** Subject to the terms and conditions of this Agreement LICENSOR grants to LICENSEE a paid-up, perpetual, nonexclusive, irrevocable license to use, copy and transport Licensed Software. LICENSEE's right to copy Licensed Software for CPU and Server licenses shall be limited to backup/recovery and archival purposes. LICENSEE's right to use Licensed Software shall include the right to use multiple versions or releases of the Licensed Software, if available. LICENSEE's rights to use and copy include the rights to load the Licensed Software into memory for maintenance, testing, and execution.

The specific license definition for each Licensed Software is defined in Exhibit A.

- 1.2 **Test and Help Desk Facilities.** The grant in Section 1.1 includes, for no extra fees, the right to install, access, and use the Licensed Software at any of LICENSEE's computing and Help Desk facilities for the purpose of testing, supporting, and distributing the Licensed Software. This includes the right to install and test the Licensed Software on an extra CPU for up to ninety (90) days while LICENSEE is testing hardware and/or software upgrades. No production use of the Licensed Software is granted in this Section.
- 1.3 **Support and Maintenance Optional.** The grant in Section 1.1 shall survive any election by LICENSEE to terminate maintenance services from LICENSOR.
- 1.4 **Disaster Recovery.** The right to use Licensed Software shall include the right to use the Licensed Software on one or more CPUs at any of LICENSEE's or third party operated disaster recovery facilities for temporary disaster/recovery testing and/or processing. LICENSEE's rights to use and copy include the rights to install the Licensed Software in a 'fail-over' mode for disaster recovery purposes at no additional cost.
- 1.5 **Audit Rights.** Pursuant to any applicable governmental requirements and/or regulations imposed on LICENSEE in the normal course of its business or any LICENSEE internal quality control procedures, LICENSOR shall cooperate with LICENSEE during the term of this Agreement by allowing LICENSEE and/or any governmental

agency having jurisdiction, upon thirty (30) days' prior written request, access to audit the software development and maintenance processes of LICENSOR and other information reasonably related to the licenses granted hereunder, for purposes of ensuring compliance with all applicable governmental laws and regulations and to ensure supplier quality. Audits are chargeable at current list prices.

ARTICLE II - ACCEPTANCE, TERMS

- 1.6 **Acceptance Testing.** Licensee shall have 30 days from invoice date to perform Acceptance Testing of the Licensed Software; LICENSEE shall test the Licensed Software to verify that they meet LICENSEE's requirements. . Within this period, LICENSEE shall notify LICENSOR in writing if LICENSEE rejects the Licensed Software. If LICENSEE rejects the Licensed Software, LICENSEE shall notify the LICENSOR of the nature, extent, and identity of any Errors, Defects, or Omissions in the Licensed Software, which cause LICENSEE to reject acceptance of the Licensed Software. If LICENSEE fails to notify LICENSOR during this time, then all such products are deemed accepted.
- 1.7 **Taxes.** LICENSEE shall be responsible for all State, Local or Federal taxes imposed including Duty and/or Brokerage Fees as a result of this agreement and LICENSEE agrees to remit directly to the applicable Government Agency as required. LICENSEE shall be responsible for any penalties or interest on any such taxes.
- 1.8 **Replacement of Lost/Damaged Media.** In the event that LICENSEE loses or damages the media that contains the Licensed Software, then, upon written notice from LICENSEE of such an event, LICENSOR shall provide a replacement copy of the Licensed Software at no charge other than production, shipping and handling costs, providing LICENSEE has a paid up SMP (defined in 3.0).

ARTICLE III - PROPRIETARY RIGHTS

- 2.0 **No Title in LICENSEE.** No title or ownership of Licensed Software is transferred to LICENSEE by way of this Agreement. The Licensed software is copyright protected by the LICENSOR with all rights reserved.
- 2.1 **Rights in Derivative Works.** LICENSOR retains all rights to derivative works related to the Licensed Software resulting from work performed by LICENSOR on behalf of the LICENSEE.

LICENSEE will own work that is unrelated to the Licensed Software and which is identified as 'Custom Work' in the Requirements Document prior to any such work being performed.

- 2.2 **Reverse-Engineering.** LICENSEE agrees not to reverse-engineer the object code form of Licensed Software in any manner.
- 2.3 **Confidentiality, Non-Disclosure.** LICENSEE agrees not to disclose the Licensed Software to any third party, except to parties under contract to LICENSEE who have agreed to abide by the confidentiality restrictions set forth in this Agreement.
- 2.4 **Exceptions to Confidentiality.** Nothing contained herein shall in any way restrict or impair LICENSEE's rights to use, disclose, or otherwise deal with any portion of Licensed Software which:
- is or becomes generally available to the public through no wrongful act of LICENSEE;
 - was in LICENSEE's possession prior to the time it was acquired from LICENSOR and which was not directly or indirectly acquired from LICENSOR;
 - is independently made available as a matter of right to LICENSEE by a third party;
 - is required, in the opinion of LICENSEE's legal counsel, to be disclosed by court order or operation of law, provided that LICENSOR is given notice of any court proceeding and an opportunity to contest disclosure; or
 - is independently developed for LICENSEE by persons not having exposure to those portions of Licensed Software or Licensed Material excepted above.
- 2.5 **Period of Confidentiality.** LICENSEE's obligations of confidentiality and nondisclosure regarding any Licensed Software shall terminate Two (2) Years after termination of use of Licensed Software by LICENSEE.
- 2.6 **Copyright Notices.** LICENSEE agrees not to remove any copyright notices and other proprietary legends appearing on Licensed Software.
- 2.7 **Assignment Of/By LICENSOR.** LICENSOR may assign this License and Agreement to a third party successor which agrees to be bound by all terms and conditions of this Agreement. In the event that LICENSOR itself or its ownership rights to any Licensed Software is transferred or assigned to an unaffiliated organization, then LICENSOR shall notify LICENSEE in writing within thirty (30) days of such event.
- **LICENSOR's Confidentiality, Nondisclosure Obligations.** In the event that LICENSEE discloses to LICENSOR any information about LICENSEE's

business which is marked or designated as confidential or proprietary, LICENSOR shall keep the same confidential and not disclose it to any third party without LICENSEE's prior written consent. All data or information in LICENSEE's computer systems and any type of report based on this data or information to which LICENSOR has access or which are given to LICENSOR are hereby deemed LICENSEE Confidential Information. LICENSOR shall treat such information in the same manner as it treats its own confidential information of the same type. LICENSOR shall return to LICENSEE or destroy all such confidential information and certify in writing within thirty (30) days as to its return or destruction, upon the LICENSEE's written request,

LICENSOR's obligations of confidentiality and nondisclosure hereunder shall terminate two (2) years following LICENSOR's certification of return or destruction of LICENSEE's confidential information or one (1) year following termination of the applicable Schedule A to this Agreement, whichever is later.

2.8 **LICENSOR's Exceptions to Confidentiality.** Nothing contained herein shall in any way restrict or impair LICENSOR's rights to use, disclose, or otherwise deal with any portion of LICENSEE's confidential information which

- is or becomes generally available to the public through no wrongful act of LICENSOR;
- was in LICENSOR's possession prior to the time it was acquired from LICENSEE and which was not directly or indirectly acquired from LICENSEE;
- is independently made available as a matter of right to LICENSOR by a third party;
- is required, in the opinion of LICENSOR's legal counsel, to be disclosed by court order or operation of law, provided that LICENSEE is given notice of any court proceeding and an opportunity to contest disclosure; or
- is independently developed for LICENSOR by persons not having exposure to those portions of LICENSEE's confidential information.

2.9 **Data Privacy.** LICENSOR represents and warrants that it will comply with all applicable privacy laws and regulations with respect to any data collected (or to which LICENSOR is given access) from LICENSEE or its employees, agents, consultants or contractors including any Personal Information as a result of LICENSOR providing Licensed Software or maintenance services under this Agreement. All Personal Information

provided to, or to which LICENSOR is given access too, shall be deemed LICENSEE's confidential information and shall not be used by LICENSOR for any purpose other than that of providing the Licensed Software and maintenance services to LICENSEE, nor shall such data or any part of such data be disclosed, sold, assigned, leased or otherwise disposed of to third parties or commercially exploited by or on behalf of LICENSOR:

ARTICLE IV – SUPPORT AND MAINTENANCE

3.0 **Support and Maintenance.** A Support and Maintenance Plan ("SMP") fee is required for the first year of the Agreement. For second and subsequent years, the LICENSEE may elect to purchase additional SMP's. An SMP will provide the following services from the LICENSOR for a period of 1 year:

- correct defects and publish them as Upgrades. Defects will be considered for correction, at the discretion of the LICENSOR, for a period of 3 years from the release date of a full point or a 1/10 point version release.
- promptly provide LICENSEE with all Upgrades, upon request;
- furnish telephone, email and support portal support with qualified personnel knowledgeable in the Licensed Software. Such support is provided for 5 years following release. Support includes providing technical help and information, but it does not include items which are provided as billable Services by the LICENSOR. Without limitation, examples of excluded items include method development, method trouble shooting, system analysis, and design work. Support is also not provided for software that has been installed on non-approved configurations.
- respond to support requests within four (4) hours, during supported hours
- provide access to LICENSOR'S Support Portal
- Provide access to LICENSOR'S User's Forum

If an SMP is not purchased in a given year, LICENSEE will be required to pay missed SMP fees to activate Support and Maintenance.

3.1 **Maintenance Standards.** LICENSOR represents and warrants that it shall perform all software maintenance services at least according to the same quality standards as are contained in the Licensed Software at the time initially licensed by LICENSEE. LICENSOR will maintain the current release and one release prior of the Licensed Software.

3.2 **Termination, Reinstatement.** LICENSEE may choose not to renew their SMP and thereby not to

receive support and maintenance offered by LICENSOR. Subsequent to any such termination, LICENSEE may, at its option, reinstate maintenance services for the upcoming maintenance period by providing notice to LICENSOR and making payment of the maintenance fees agreed to. In the case of reinstatement maintenance services, LICENSEE shall be obligated to pay for the upcoming maintenance period as well as for each year during which provision of Upgrades was not purchased.

ARTICLE V - WARRANTIES

4.0 **Ownership.** LICENSOR hereby represents and warrants that for Licensed Software that is marked with 'Copyright Labtronics' (i) it has full and complete ownership in the Licensed Software and all copyrights and other intellectual property rights existing therein; (ii) it has the right to license the Licensed Software to LICENSEE hereunder; and (iii) to the best of LICENSOR'S knowledge the Licensed Software does not infringe upon any third party's patent, trademark, copyright or other intellectual property rights or trade secrets.

4.1 **Conformance to Licensed Materials.** LICENSOR does not warrant that the Licensed Software will operate error-free. LICENSOR represents and warrants that Licensed Software will perform in accordance with applicable Licensed Material for a period of 90 days from the original date of purchase. In the event that the Licensed Software shall prove defective, the LICENSEE's sole remedy shall be the replacement of this product or the return of the Purchase price.

4.2 The above is the only warranty of any kind, expressed or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose that is made by the LICENSOR. Under no circumstances shall LICENSOR *nor* its agents be liable for any loss or damage, direct or indirect, incidental or consequential, arising out of the use of, or inability to use, this product.

4.3 **Century Date compliant:** LICENSOR warrants that Licensed Software "Century Date Compliant". Any such Licensed Software is deemed to be Century Date Compliant if:

- (i) the functions, calculations, and other computing processes of the product (collectively "Processes") perform in a consistent manner regardless of the date in time on which the Processes are actually performed and regardless of the date on which

data was input into the Licensed Software, whether before, on, or after January 1, 2000 and whether or not the dates are affected by leap years;

- (ii) the Licensed Software accepts, calculates, compares, sorts, extracts, sequences, and otherwise processes date inputs and date values, and returns and displays date values in a consistent manner regardless of the dates used, whether before, on , or after January 1, 2000;
- (iii) the Licensed Software will function without interruptions caused by the date in time on which the Processes are actually performed or by the date input to the system, whether before, on or after January 1, 2000;
- (iv) the Licensed Software accepts and responds to year input in a manner that resolves any ambiguities as to century in a defined and predetermined and appropriate manner; and
- (v) the Licensed Software stores and displays date information in ways that are unambiguous as to the determination of the century.

4.4 **Consequential Damage.** NEITHER PARTY WILL BE LIABLE TO THE OTHER FOR ANY LOST REVENUE OR INDIRECT, PUNITIVE, EXEMPLARY, SPECIAL, OR CONSEQUENTIAL DAMAGES, EVEN IF SUCH PARTY HAS BEEN ADVISED AS TO THE POSSIBILITY OF SUCH DAMAGES. FOR PURPOSES OF THIS AGREEMENT, ANY DAMAGES, PENALTIES, FINES, OR EQUITABLE REMEDIES PAYABLE TO THIRD PARTIES SHALL BE CONSTRUED AS DIRECT DAMAGES.

4.5 **Contracts With Employees.** LICENSOR warrants that it will be responsible for the conduct of its employees and agents and will obtain from its employees and agents any contracts which are needed to enforce its obligations under this Agreement.

4.6 **FDA 21 CFR Part 11 Compliance:** LICENSOR represents that the Licensed Software, when properly installed and utilized in accordance with the Licensed Material, complies in all material respects with FDA regulation 21 CFR Part 11 as it may be relevant to the intended use of the Licensed Software. Both parties agree that the business processes that are deployed by LICENSEE in conjunction with the Licensed Software may be relevant factors regarding compliance with 21 CFR Part 11. In the event that the 21CFR Part 11 requirements as they relate to the intended use of the Licensed Software are modified by the FDA, LICENSOR agrees to furnish compliant changes through the standard maintenance program set forth in Section 4.1. LICENSOR agrees that if the FDA shall request information from LICENSEE regarding the compliance of the Licensed Software to

21 CFR Part 11 requirements, that LICENSOR will furnish such information to LICENSEE in a timely manner.

ARTICLE VI - GENERAL PROVISIONS

- 5.0 **Assignment of/by LICENSEE.** LICENSEE may assign its rights or obligations hereunder, with LICENSOR's prior written consent, to an Affiliate, to a successor to the business or operation of LICENSEE or any LICENSEE Affiliate or to any other party which agrees to be bound by all terms and conditions of this Agreement. In the event that LICENSEE divests a LICENSEE Business Entity which has been using the Licensed Software in the ordinary course of its business operations under LICENSEE's license pursuant to this Agreement, LICENSEE may continue to use the Licensed Software on behalf of the former LICENSEE Business Entity during a transition period of up to two (2) years at no added charge
- 5.1 **Entire Agreement, Partial Invalidity.** This Agreement embodies the entire understanding of the parties hereto on the subject matter hereof and supersedes any previous agreements or understandings, written or oral, in effect between the parties relating to the subject matter hereof. If any part, term, or provision of this Agreement shall be held illegal, unenforceable, or in conflict with any law of a federal, state, Provincial or local government having jurisdiction over this Agreement, the validity of the remaining portion or portions shall not be affected thereby.
- 5.2 **Changes.** LICENSOR reserves the right to modify this Agreement at any time. Copies of modified Agreements will be made available to LICENSEE.
- 5.3 **Waiver of Breach.** The waiver of a breach of this Agreement or the failure of a party to exercise any right under this Agreement shall in no event constitute a waiver as to any other breach, whether

similar or dissimilar in nature, or prevent the exercise of any right under this Agreement.

- 5.4 **No Other Relationship/Obligations.** Neither party shall have any right, power, or authority to assume, create, or incur any expense, liability, or obligation, express or implied, on behalf of the other party, except as expressly provided herein. This Agreement is not intended to be nor shall it be construed as a joint venture, association, partnership, or other form of a business organization or agency relationship.
- 5.5 **GOVERNING LAW, FORUM.** THIS AGREEMENT SHALL BE CONSTRUED AND THE LEGAL RELATIONS BETWEEN THE PARTIES DETERMINED IN ACCORDANCE WITH THE LAWS OF THE PROVINCE OF ONTARIO, CANADA.
- 5.5.1 Both parties hereby consent to the exclusive jurisdiction of the courts of the Province of Ontario and expressly waive any objections or defense based upon lack of personal jurisdiction or venue.
- 5.6 **Force Majeure.** Neither party shall be responsible for any failure to perform or delay in performing any of its obligations under this Agreement where and to the extent that such failure or delay results from causes outside the reasonable control of the party. Such causes shall include, without limitation, Acts of God or of the public enemy, acts of the government in either its sovereign or contractual capacity, fires, floods, epidemics, quarantine restrictions, freight embargoes, civil commotion's, or the like. Notwithstanding the above, strikes and labor disputes shall not constitute an excusable delay for either party under this Agreement.
- 5.7 **Headings, Counterparts.** Headings used in this Agreement are for reference purposes only and shall not be used to modify the meaning of the terms and conditions of this Agreement. This Agreement may be executed in two or more counterparts each of which shall be deemed an original, but all of which together shall constitute one and the same instrument.

By: Robert pavlis, President



SCHEDULE A

The following is a definition for the license for each Licensed Software.

Licensed Software	License Definition
Collect	Is licensed for use on a single machine installation. A single machine installation is defined as any computer or electronic device that has the software installed locally or is used to access the software through a remote terminal client.
PipetteTracker, PipetteTracker Pro, PipetteTracker Pro+	Is licensed for use on a single machine installation. A single machine installation is defined as any computer or electronic device that has the software installed locally or is used to access the software through a remote terminal client.
NAP	<p>Is licensed for use on a single machine installation. A single machine installation is defined as any computer or electronic device that has the software installed locally or is used to access the software through a remote terminal client.</p> <p>In addition to this, NAP is also licensed for a certain number of channels which can be either a 2 channels or 6 channels. Adding additional channels requires additional channels.</p>
LimsLink, LimsLink CDS	<p>Is licensed on an instrument basis. Each instrument that sends data to LimsLink requires a license. A bar code reader does not require a license when used in conjunction with an instrument that has a valid license.</p> <p>For multiplexed instruments, such as chromatography instruments connected to a central CDS, each instrument requires a license.</p>
Nexxis	Nexxis is licensed on a concurrent user basis.
Nexxis ELN	Nexxis ELN is licensed on a concurrent user basis.
Nexxis iLAB	Nexxis iLAB is licensed on a concurrent user basis.
Nexxis SDMS	<p>Nexxis SDMS consists of 2 parts. One part is a Labtronics module and it is licensed on a concurrent user basis.</p> <p>Nexxis SDMS also includes a license for KnowledgeTree document management system as described below.</p>
KnowledgeTree	<p>This product is available in 2 formats:</p> <p>Open Source format – the product is licensed by Open Source GNU Ver. 3.0 License Agreement. See web site http://www.knowledgetree.com for full license details.</p> <p>Commercial format – the license has two parts. It is licensed per site on an annual subscription basis as well as on an Instrument license and/or a concurrent user license basis. The Labtronics Master Software License Agreement applies for the Commercial format.</p>
VTK (validation tool kit) for all products	Is licensed on a per site basis. Every site that uses the VTK for validation purposes, even if modified by the LICENSEE, requires to purchase a valid license

IMPORTANT INFORMATION

Policy Regarding the Installation Of Labtronics Products

The purpose of this notice is to inform you that all Labtronics products should be installed using the automated InstallShield method supplied with the product. The process has been qualified and is the process that will be supported.

Labtronics is limited in the support that can be provided for any system that has been installed by another means. When asked to research a possible defect, the customer will be required to install using the automated method and verify the event before an investigation by Labtronics is initiated.

Policy Regarding Labtronics Field Fixes

In certain situations, there may be field fixes included with the software. If you decide to apply a field fix before running the software, some files may have to be updated. This information can be found in the Information Data Sheet that accompanies the field fix.

A listing of available field fixes for a particular product can be obtained through a User Forum. Prior to performing any installation of this product, it is recommended that our clients review the list of field fixes available and determine whether they are appropriate for their intended deployment.

Upon registration, the User Forum can be accessed from the support area of the Labtronics website (www.labtronics.com).

Installation

Contents of the LimsLink EI Package

- One CD-ROM, containing **LimsLink** EI v4.2 program files. Install files may also be located on the Nexxis iLAB installation DVD.
- One **LimsLink** EI v4.2 Installation and Configuration Manual
- One **LimsLink** EI v4.2 User Manual including CDS Module and LIMS Module inserts
- One ReportLink v2.0 User Manual

Hardware Requirements

To run LimsLink EI effectively, we recommend that your computer have the following minimum specifications:

- P4 1.0 GHz CPU
- 256 MB RAM
- 100 MB hard disk space
- CD-ROM drive for installation.
- SVGA graphics with a resolution of at least 1024 x 768.

Software Requirements

LimsLink EI

LimsLink EI is supported under the following operating systems:

- Windows Server 2003 R2 SP2 (32 bit)
- Windows XP Professional SP2
- Windows XP Professional SP3

LimsLink EI Databases

The LimsLink EI Databases support the following database management systems:

- SQL Server 2005 SP2
- SQL Server 2005 Express Edition SP2
- Oracle 10g Release 2 (10.2)

Installing LimsLink EI

Installation Prerequisites

Certain prerequisites are required for the installation of LimsLink EI. They are as follows:

- The installation for the LimsLink EI application assumes that the LIMS and CDS that you are using have been previously installed. You will be asked to select the install targeted for a specific CDS module. Also during the installation process, you will need to select the LIMS module that you will be using with LimsLink EI. Refer to the specific CDS and LIMS module documentation for installation requirements specific to a particular CDS or LIMS module.

Installing LimsLink EI

1. Ensure your system meets the minimum hardware and software requirements and the prerequisites for LimsLink EI.
2. Insert the LimsLink EI CD into your computer's CD-ROM drive.
3. The program will attempt to autostart the setup routine. If the setup does not begin automatically, you can manually start it by running the setup.exe file in the root folder of the CD.
4. Select LimsLink EI Installation to begin the installation procedure.
5. Select the LimsLink EI Installation item for the CDS Module you are using (e.g., LimsLink EI Installation for Empower used in this exemplary case).
6. Select Next at the Welcome to the InstallShield Wizard for LimsLink EI v4.2
7. Read the license agreement and select 'I accept the terms of the license agreement'. Select Next.

8. Select the LIMS for this instance of the install. Select Next.
9. At the Choose Destination Location window, select the folder where Setup will install files. To install to the default folder, select Next. To install to a different folder, click Change, select another folder, click OK, and then click Next.
10. Certain modules may require individual setup at installation. If so, the setup windows will appear next in the installation process. Specific instructions for the module will appear onscreen. Once complete, click the 'Next' button. Refer to the specific CDS module User Manual for more information.
11. Select 'Install' to begin the installation. At this point, the software will verify that the CDS module is installed. If not, the installation process will abort.

Installing the LimsLink EI Databases

If using SQL Server 2005 Express Edition, a menu item is available on the installation CD to automate the SQL Server 2005 Express Edition installation and database creation process for you. Refer to the following Installation Prerequisites section to ensure you meet the minimum prerequisites before starting the installation.

If using another database management system, the tables for each database can be created using scripts contained on the installation CD in the LimsLink EI\Databases\<database type> folder where the database type is one of SQL_Server or Oracle.

The following databases are used by LimsLink EI:

1. LimsLink EI Main database

This database contains setup information about LimsLink EI and its methods.

2. LimsLink EI Archive database

This database contains archived copies of methods that have been validated.

Each database has a unique table structure, so they can be located in the same owner/schema within the same database if desired. However, because each database can grow at different rates, it may be advisable to store them in separate filegroups/tablespaces or separate databases.

Installing the LimsLink EI Databases on a SQL Server 2005 Express Edition Platform

Installation Prerequisites

Windows must have the latest service packs installed. In addition, the following components must be installed:

Microsoft .NET Framework 2.0.

Windows Installer 3.1

Note: Windows may NOT have these components installed by default.

Microsoft .NET Framework 2.0, MDAC 2.8 SP1, and Windows Installer 3.1 are included as separate

installs under the Support folder on the installation CD.

To install Microsoft .NET Framework 2.0, do the following:

1. Execute the dotnetfx.exe executable located in the Support\Dot_Net_2.0 folder of the installation CD.
2. Follow the prompts to complete the installation.

To install Windows Installer 3.1, do the following:

1. Execute the WindowsInstaller-KB893803-v2-x86.exe executable located in the Support\Windows_Installer_3.1 folder of the installation CD.
2. Follow the prompts to complete the installation.

Installation

1. A menu item is available on the installation CD to automate the SQL Server Express installation and the database creation process.
2. Ensure your system meets the minimum hardware and software requirements and prerequisites for SQL Server 2005 Express Edition.
3. Insert the installation CD into your computer's CD-ROM drive.
4. The program will attempt to auto-start the setup routine. If the setup does not begin automatically, you can manually start it by running the setup.exe file in the root directory of the CD.
5. From the installation menu, select the LimsLink EI Installation menu item.
6. From the installation sub-menu, select the LimsLink EI Database Installation with SQL Server Express menu item.
7. The automated installation will proceed.

Installing LimsLink EI Databases on a SQL Server 2005 Platform

The scripts for generating the **LimsLink EI** databases under SQL Server 2005 can be found on the installation CD in the Databases\SQL_Server folder. These scripts can be used to create the databases using a default configuration or for other configurations, based on your environment and operating requirements.

The database scripts can be executed using the SQL Server Command Line Tool, SQLCMD, from the Windows Command Prompt if the SQL Server Client components are installed on the system. Executing the scripts from the SQL Server itself will ensure the necessary components are present.

Creating the LimsLink EI Databases for the Default Configuration

1. Ensure you are logged onto Windows as a user with administrative rights to SQL Server. If necessary, log onto the SQL Server machine itself as a local administrator.
2. Open the 'Databases\SQL_Server' folder on the installation CD.
3. Copy the database scripts to a folder on your hard drive. This must be done on a system that has the SQL Server Command Line Tool, SQLCMD, and can connect to the desired SQL Server.

4. Open the Command Prompt (Start > Programs > Accessories > Command Prompt).
5. Navigate to the folder where you saved the database scripts. For example,

```
cd C:\temp\LLEIDBs\SQL_Server
```

6. At the command prompt, execute the following command to create the databases for LimsLink EI using a default configuration. It creates databases called "LimsLinkEI" and "LimsLinkEIArchive" and runs the creation scripts against them. To run the script type the following command:

```
sqlcmd -E -S ServerName[InstanceName] -i LLEI_CreateDatabasesWithDefaults.sql
```

where,

- *ServerName* is the computer name of the machine running SQL Server
- *InstanceName* is the name of the SQL Server instance. If the default instance is used, then this can be omitted

Creating the LimsLink EI Databases for Other Configurations

1. Ensure you are logged onto Windows as a user with administrative rights to SQL Server. If necessary, log onto the SQL Server machine itself as a local administrator.
2. Open the 'Databases\SQL_Server' folder on the installation CD.
3. Copy the database scripts to a folder on your hard drive. This must be done on a system that has the SQL Server Command Line Tool, SQLCMD, and can connect to the desired SQL Server.
4. Open the Command Prompt (Start > Programs > Accessories > Command Prompt).
5. Navigate to the folder where you saved the database scripts. For example,

```
cd C:\temp\LLEIDBs\SQL_Server
```

6. At the command prompt, execute the following command to create a database for the LimsLink EI Main database.

```
sqlcmd -E -S ServerName[InstanceName] -Q "CREATE DATABASE DatabaseName"
```

where,

- **ServerName** is the computer name of the machine running SQL Server
- **InstanceName** is the name of the SQL Server instance. If the default instance is used, then this can be omitted.
- **DatabaseName** is the name of the database to create

7. When you are returned to the command prompt, run the script to create the database tables by typing in the following command:

```
sqlcmd -E -S ServerName[InstanceName] -d DatabaseName -i LimsLinkEI_SQL.sql
```

where,

- *ServerName* is the computer name of the machine running SQL Server
- *InstanceName* is the name of the SQL Server instance. If the default instance is used, then this can be omitted.
- *DatabaseName* is the name of the LimsLink EI main database created above.

8. Create the LimsLink EI Archive database by typing in the following command:

```
sqlcmd -E -S ServerName[InstanceName] -Q "CREATE DATABASE DatabaseName"
```

where,

- *ServerName* is the computer name of the machine running SQL Server
- *InstanceName* is the name of the SQL Server instance. If the default instance is used, then this can be omitted.
- *DatabaseName* is the name of the database to create

9. Run the script to create the database tables by typing in the following command:

```
sqlcmd -E -S ServerName[InstanceName] -d DatabaseName -i LimsLinkEIArchive_SQL.sql
```

where,

- *ServerName* is the computer name of the machine running SQL Server
- *InstanceName* is the name of the SQL Server instance. If the default instance is used, then this can be omitted.
- *DatabaseName* is the name of the LimsLink EI main database

Creating Users and Roles to Access the LimsLink EI Databases on a SQL Server 2005 Platform

Database roles can be created to control access to the LimsLink EI databases. This is often preferred to accessing the databases as the database owner. The scripts for generating the database roles and granting permissions to them are contained in the installation CD under the

\Databases\SQL_Server\Roles folder. Scripts are available for use with the default configuration or for other configurations.

Creating Roles for LimsLink EI Using the Default Configuration

If you are using the default configuration for the LimsLink EI databases, there is a script available to create the necessary login, users, and roles automatically. A new database role named LLEI_User will be added to the LimsLinkEI and LimsLinkEIArchive databases and be granted to a new LimsLinkEI user, with a password of Labtr0nics (with a zero).

1. Ensure you are logged onto Windows as a user with administrative rights to SQL Server. If necessary, log onto the SQL Server machine itself as a local administrator.
2. Open the '\Databases\SQL_Server\Roles' folder on the installation CD.
3. Copy the database scripts to a folder on your hard drive. This must be done on a system that has the SQL Server Command Line Tool, SQLCMD, and can connect to the desired SQL Server.
4. Open the Command Prompt (Start > Programs > Accessories > Command Prompt).
5. Navigate to the folder where you saved the database scripts. For example,

```
cd C:\temp\LLEIDBs\SQL_Server\Roles
```

6. At the command prompt, execute the following command to create the new login, users, and roles, granting the necessary permissions.

```
sqlcmd -E -S ServerName[InstanceName] -i LLEI_CreateLoginUsersAndRolesWithDefaults.sql
```

ServerName is the computer name of the machine running SQL Server

- *InstanceName* is the name of the SQL Server instance. If the default instance is used, then this can be omitted.

Creating Roles for LimsLink EI Using Other Configurations

1. Ensure you are logged onto Windows as a user with administrative rights to SQL Server. If necessary, log onto the SQL Server machine itself as a local administrator.
2. Open the '\Databases\SQL_Server\Roles' folder on the installation CD.
3. Copy the database scripts to a folder on your hard drive. This must be done on a system that has the SQL Server Command Line Tool, SQLCMD, and can connect to the desired SQL Server.
4. Open the Command Prompt (Start > Programs > Accessories > Command Prompt).

5. Navigate to the folder where you saved the database scripts. For example,

```
cd C:\temp\LLEIDBs\SQL_Server\Roles
```

6. At the command prompt, execute the following command to create a new “LimsLinkEI” login with a password of “Labtr0nics”.

```
sqlcmd -E -S ServerName[InstanceName] -i LLEI_CreateLogin.sql
```

where,

- *ServerName* is the computer name of the machine running SQL Server
- *InstanceName* is the name of the SQL Server instance. If the default instance is used, then this can be omitted.

7. When you are returned to the command prompt, run the script to create the main database user and role by typing in the following command:

```
sqlcmd -E -S ServerName[InstanceName] -d DatabaseName -i LLEI_CreateRoleForMain.sql
```

where,

- *ServerName* is the computer name of the machine running SQL Server
- *InstanceName* is the name of the SQL Server instance. If the default instance is used, then this can be omitted.
- *DatabaseName* is the name of the LimsLink EI main database.

8. When you are returned to the command prompt, run the script to create the archive database user and role by typing in the following command:

```
sqlcmd -E -S ServerName[InstanceName] -d DatabaseName -i LLEI_CreateRoleForArchive.sql
```

where,

- *ServerName* is the computer name of the machine running SQL Server
- *InstanceName* is the name of the SQL Server instance. If the default instance is used, then this can be omitted.
- *DatabaseName* is the name of the LimsLink EI archive database.

Assigning the Role to a Specific User

By default the above scripts create a LimsLinkEI login, with a password of Labtr0nics (with a zero), and user, assigning the roles to that user. If you wish to assign the roles to other users, the following

command will do so:

```
sqlcmd -E -S ServerName[\iInstanceName] -d DatabaseName -Q "execute sp_addrolemember  
'LLEI_User', 'UserName'"
```

where,

- *ServerName* is the computer name of the machine running SQL Server
- *InstanceName* is the name of the SQL Server instance. If the default instance is used, then this can be omitted.
- *DatabaseName* is the name of the LimsLink EI database being used.
- *UserName* is the name of the user to which the role should be granted

This command will need to be executed for each of the main and archive databases.

Installing LimsLink EI Databases on an Oracle Database Platform

The scripts for generating the **LimsLink EI** schemas under Oracle Database can be found on the installation CD in the Databases\Oracle folder. These scripts can be used to create the database objects using a default configuration or for other configurations, based on your environment and operating requirements.

The database scripts must be executed using SQL*Plus, which is generally installed as part of the Oracle Database client.

Updating the Tablespace Creation Scripts

The tablespace creation scripts should be updated to include the path you wish to use for your environment. If these steps are not performed, the data files for the tablespaces will be created in a default location, which may not be the preferred location for your server.

1. Open the '\Databases\Oracle' folder on the installation CD.
2. Copy the scripts to a folder on your hard drive. This must be done on a system that has the Oracle Database client installed and configured with SQL*Plus.
3. From the new folder on your hard drive, open the 'LIMSLINK_DATA.sql' file in Notepad. Update the 'LIMSLINK_DATA.DBF' reference to include the path for your database files. For example,

'C:\oracle\product\10.2.0\oradata\orc\LIMSLINK_DATA.DBF'

4. Open the 'LIMSLINK_IX.sql' file in Notepad. Update the 'LIMSLINK_IX.DBF' reference to include the path for your database files. For example,

'C:\oracle\product\10.2.0\oradata\orc\LIMSLINK_IX.DBF'

Creating the LimsLink EI Databases for the Default Configuration

This procedure can be used to create the LimsLink EI Databases using the default user and schema names. It will create both LimsLink EI databases under a 'LLEI_Owner' user and schema.

1. Open the Windows Command Prompt (Start > Programs > Accessories > Command Prompt) and navigate to the folder where you saved the database scripts. For example,

```
cd C:\temp\LLEIDBs\Oracle
```

2. Using SQL*Plus, connect to the Oracle Database server where the databases are to be created. For example,

```
sqlplus SystemUser/SystemPassword[@OracleServer]
```

where:

- **SystemUser** is the name of a DBA user
 - **SystemPassword** is the password for **SystemUser**
 - **OracleServer** is the identifier of the database instance that will contain the data. Often this is the Service Name for the database or the computer name of the server itself. If working directly on the database server, the '@OracleServer' portion can generally be omitted.
3. At the SQL> prompt, enter the command '@LLEI_RunAllScriptsWithDefaults.sql' and press Enter. The script will call other scripts to create database objects, using default values for the various prompts. The username and password for the LimsLink EI and LimsLink EI Archive databases are LLEI_Owner/LabtrOnics (with a zero).

Creating the LimsLink EI Databases for Other Configurations

This procedure can be used to tailor the creation of the LimsLink EI Databases to use the user names you specify.

1. Open the Windows Command Prompt (Start > Programs > Accessories > Command Prompt) and navigate to the folder where you saved the database scripts. For example,

```
cd C:\temp\LLEIDBs\Oracle
```

2. Using SQL*Plus, connect to the Oracle Database server where the databases are to be created. For example,

```
sqlplus SystemUser/SystemPassword[@OracleServer]
```

where:

- SystemUser is the name of a DBA user
 - SystemPassword is the password for SystemUser
 - OracleServer is the identifier of the database instance that will contain the data. Often this is the Service Name for the database or the computer name of the server itself. If working directly on the database server, the '@OracleServer' portion can generally be omitted.
3. At the SQL> prompt, enter the command '**@LLEI_RunMainScripts.sql**' and press Enter. The script will call other scripts to create database objects, prompting for values as required.
 4. When prompted for a user name and password, enter the values you would like to use for the main database. The script will proceed with creating the database objects.
 5. When you are returned to the SQL> prompt, enter the command '**@LLEI_RunArchiveScripts.sql**' and press Enter. This script will call other scripts to create database objects, prompting for values as required.
 6. When prompted for a user name and password, enter the values you would like to use for the archive database. The script will proceed with creating the database objects. The username and password for the LimsLink EI and LimsLink EI Archive databases are LLEI_Owner/LabtrOnics (with a zero).

Creating Users and Roles to Access the LimsLink EI Databases on an Oracle Database Platform

Database roles can be created to control access to the LimsLink EI databases. This is often preferred to accessing the databases as the database owner. The scripts for generating the database roles and granting permissions to them are contained in the installation CD under the \Databases\Oracle\Roles folder. Scripts are available for use with the default configuration or for other configurations.

Creating Roles for LimsLink EI Using the Default Configuration

If you are using the default configuration for the LimsLink EI databases, there is a script available to create the necessary user and roles automatically. New database roles named LIMSLINK_EI_USER and LIMSLINK_EI_ARCHIVE_USER will be added a new LIMSLINK_EI user.

1. Open the '\Databases\Oracle\Roles' folder on the installation CD.
2. Copy the scripts to a folder on your hard drive. This must be done on a system that has the Oracle Database client installed and configured with SQL*Plus.
3. Open the Windows Command Prompt (Start > Programs > Accessories > Command Prompt) and navigate to the folder where you saved the database scripts. For example,

cd C:\temp\LLEIDBs\Oracle\Roles

4. Using SQL*Plus, connect to the Oracle Database server where the databases are to be created. For example,

sqlplus SystemUser/SystemPassword[@OracleServer]

where:

- SystemUser is the name of a DBA user
 - SystemPassword is the password for SystemUser
 - OracleServer is the identifier of the database instance that will contain the data. Often this is the Service Name for the database or the computer name of the server itself. If working directly on the database server, the '@OracleServer' portion can generally be omitted.
5. At the SQL> prompt, enter the command '**@LLEI_CreateRolesWithDefaults.sql**' and press Enter. The script will call other scripts to create the user and roles, using default values for the various prompts. The username and password for the LimsLink EI and LimsLink EI Archive databases are LimsLink_EI/Labtr0nics (with a zero).

Creating Roles for LimsLink EI Using Other Configurations

This procedure can be used to tailor the creation of the LimsLink EI users and roles to use the names you specify.

1. Open the Windows Command Prompt (Start > Programs > Accessories > Command Prompt) and navigate to the folder where you saved the database scripts. For example,

cd C:\temp\LLEIDBs\Oracle\Roles

2. Using SQL*Plus, connect to the Oracle Database server where the databases are to be created. For example,

sqlplus SystemUser/SystemPassword[@OracleServer]

where:

- SystemUser is the name of a DBA user
 - SystemPassword is the password for SystemUser
 - OracleServer is the identifier of the database instance that will contain the data. Often this is the Service Name for the database or the computer name of the server itself. If working directly on the database server, the '@OracleServer' portion can generally be omitted.
3. At the SQL> prompt, enter the command '**@LLEI_CreateRoles.sql**' and press Enter. The script will call other scripts to create the users and roles, prompting for values as required.
 4. When prompted for a value for LLEI_Main_Role, enter the name that will be used for the role that will govern access to the main database.
 5. When prompted for a value for LLEI_Main_Owner, enter the name of the user that owns the main database schema. This will be the user name you entered when creating the main database.

6. When prompted for a value for LLEI_Main_User, enter the name of the user that will be created to access the main database. This user will be granted the role that was created above.
7. When prompted for a value for LLEI_Main_User_Password, enter the password to use for the above user.
8. When prompted for a value for LLEI_Archive_Role, enter the name that will be used for the role that will govern access to the archive database.
9. When prompted for a value for LLEI_Archive_Owner, enter the name of the user that owns the archive database schema. This will be the user name you entered when creating the archive database.
10. When prompted for a value for LLEI_Archive_User, enter the name of the user that will be created to access the archive database. This user will be granted the role that was created above.
11. When prompted for a value for LLEI_Archive_User_Password, enter the password to use for the above user.

Upgrading from LimsLink EI v2.0 to LimsLink EI v4.2

The database structure has not changed between LimsLink EI v2.0 and LimsLink EI v4.2. As a result, LimsLink EI v4.2 can be used with a database from LimsLink EI v2.0. It is important to note that once a LimsLink EI v2.0 database is used for LimsLink EI v4.2, it can no longer be used again for LimsLink EI v2.0. If the database will continue to be used with LimsLink EI v2.0, a new database must be created for LimsLink EI v4.2.

Due to changes in the CDS and LIMS modules in LimsLink EI, each Method may need to be edited to ensure that configuration changes are applied correctly.

Upgrading from LimsLink EI v4.1.x to LimsLink EI v4.2

The database structure has not changed between LimsLink EI v4.1.x and LimsLink EI v4.2. As a result, LimsLink EI v4.2 can be used with a database from LimsLink EI v4.1.x. It is important to note that once a LimsLink EI v4.1.x database is used for LimsLink EI v4.2, it can no longer be used again for LimsLink EI v4.1.x. If the database will continue to be used with LimsLink EI v4.1.x, a new database must be created for LimsLink EI v4.2.

Due to changes in the CDS and LIMS modules in LimsLink EI, each Method may need to be edited to ensure that configuration changes are applied correctly.

LimsLink EI™

Version 4.2

User Manual



Table of Contents

Chapter 1 – Introduction	Page 1
Chapter 2 – System Setup	Page 3
Chapter 3 – Configuring a Method	Page 9
Appendix A – Programming Reference	Page 21

Introduction

Overview

LimsLink EI, a component of the Labtronics LimsLink CDS product, provides an embedded link between your CDS and LIMS. Designed to be a Plug & Play system, you can substitute in various CDS and LIMS modules to work with the interface, and set up the individual modules to suit your requirements.

As part of the LimsLink EI install, you are required to run the install targeted at your CDS, and then select the LIMS module during the installation procedure.

The LIMS module provides the functionality to retrieve the sample list from the LIMS. Modules have been defined to accommodate both queried worklists and file worklists.

The CDS module provides the functionality required to view sample lists retrieved from the LIMS, edit and expand the sample list to include control samples, and then convert this list to a Sequence, now ready for analysis by the CDS application. Once analysis is complete, the results can be reported back to the LIMS. Modules have been defined to work with many of the CDS applications on the market today.

LimsLink EI is based on the concept of Methods. A Method may be configured to retrieve information from the LIMS; modify the retrieved information to create an expanded sample list; and upload the data to the CDS. Each Method contains a script that dictates when the various components of the Method execute. The script can be edited to allow further control of the processes within a Method.

With the LimsLink EI application, LIMS integration is embedded directly into your CDS client providing users with a direct and transparent connection to their LIMS.

Manual References

The LimsLink EI User Manual describes the setup and runtime behavior of the application as a whole, but does not explain the specific setup or runtime events for the different CDS and LIMS modules.

For this reason, the LimsLink EI User Manual is supplemented with individual documents for each CDS and LIMS module. You will want to reference this material when you are configuring your system.

A LimsLink EI Installation and Configuration Manual is also available outlining the installation procedure for the LimsLink EI software.

Terminology

CDS – Chromatography Data System

DS – Data System

EI – Embedded Interface

IDS – Instrument Data System

LIMS – Laboratory Information Management System

The following terminology used throughout the software dialogs reflects the specific LIMS and CDS being used. For the purpose of this manual, certain terms may be used interchangeably.

Sequence, Sample Set Method - Refers to a list of samples to be analyzed by a CDS.

Sample List - Refers to a list of samples, pending analysis that has been queried from a LIMS.

Expanded Sample List - Refers to a sample list that has been expanded to include standards, controls and other reference samples.

System Setup

Introduction

Via the CDS Client, a series of setup dialogs can be accessed through which you can establish various links between the required modules (e.g., LIMS, CDS) as well as set up a Method to control the information retrieval and subsequent upload.

Note: All of the LimsLink EI displays are consistent with the CDS application's User Interface, allowing the user to work in a comfortable and familiar environment. The terminology used for the various dialogs and subsequent menu items is dependent on the CDS being employed, and may vary from what is depicted here.

To access these dialogs, select the **LimsLink EI - Method Setup** sub-menu item or toolbar button, from within the CDS module, to open the LimsLink Embedded Interface Setup window.

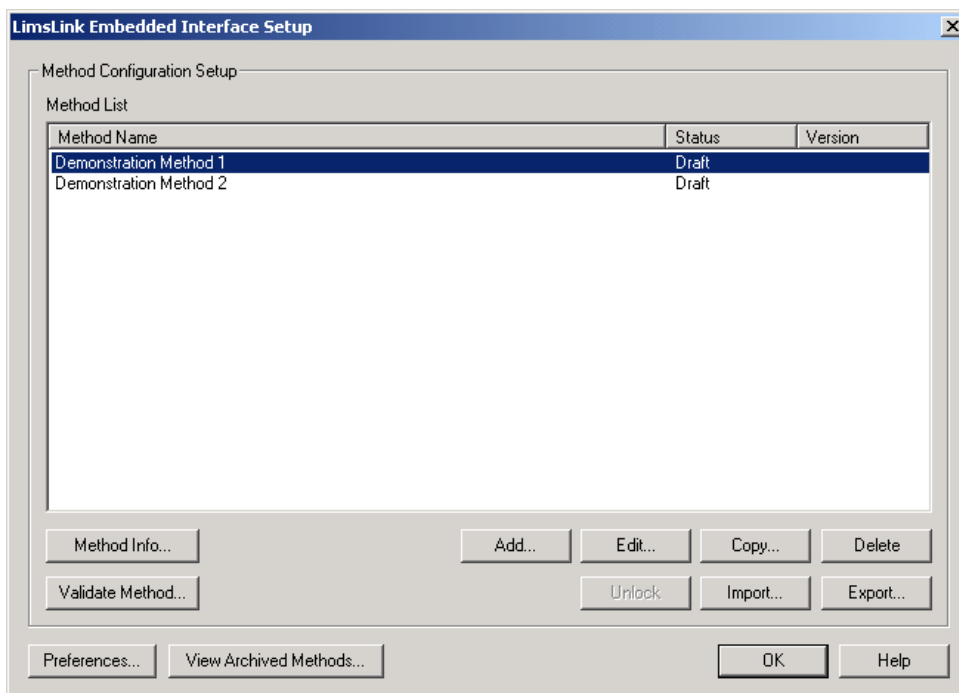


Figure 2.1: LimsLink Embedded Interface Setup

The System Setup is explained within this chapter. Refer to Chapter 3 for information on setting up a Method.

Preferences

A set of parameters governing the transfer of information between the LIMS and the CDS must first be established before information retrieval and analysis can occur. Clicking on the **'Preferences'** button on the LimsLink Embedded Interface Setup window will open the Preferences window, similar to the example shown here. Through this window, you are also able to determine the location of data repositories such as the Method Database and the Archive Method Database.

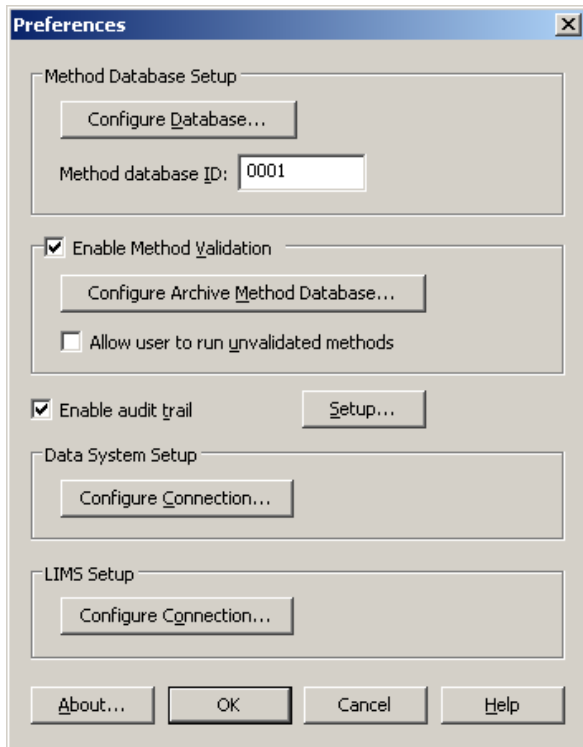


Figure 2.2: Preferences

Method Database Setup

All LimsLink EI Methods are saved to a database. To define the database, click on the '**Configure Database**' button. This will open a Data Link Properties window. From here, you can define a new database to store the Methods.

To connect to the Method Database:

1. From the Data Links Properties window, select the 'Provider' tab. From here, select the database you want to connect to. If you are using SQL Server or SQL Server Express Edition, choose Microsoft OLE DB Provider for SQL Server. If you are using Oracle Database, you will need to use the Oracle Provider for OLE DB installed with the Oracle Database Client.

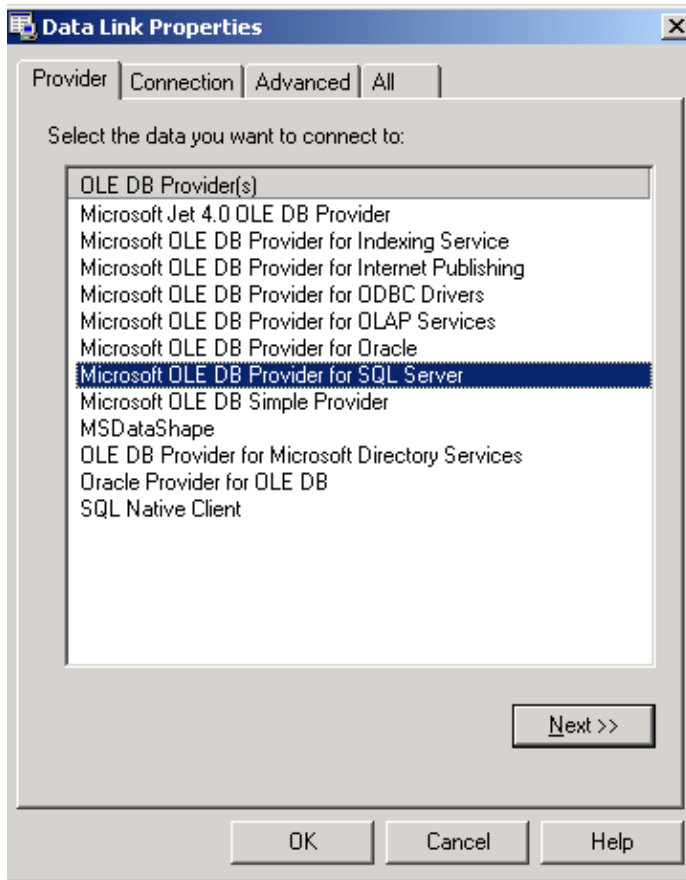


Figure 2.3: Data Link Properties – Provider tab

Note: If using the Oracle client you CANNOT use the Microsoft OLE DB Provider for Oracle.

2. Once the Provider has been established, click '**Next**' to advance to the Connection tab. From here, you can establish the connection information.

Note: If you are modifying existing connection information, the Data Link Properties window will contain the current information when opened.

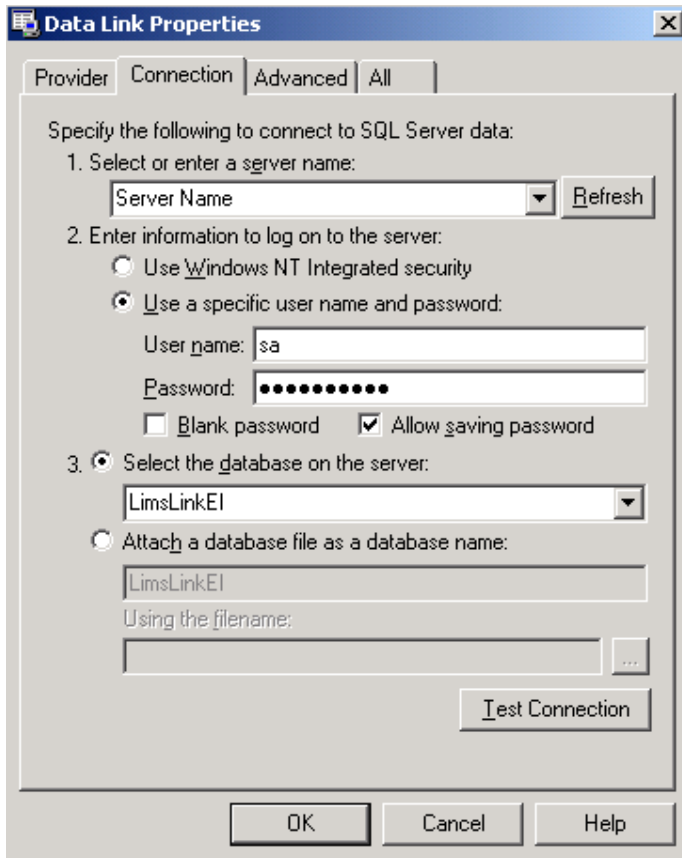


Figure 2.4: Data Link Properties – Connection tab

3. Select or enter the Server Name.

4. Enter the information to log on to the Server:

If you are using Windows NT Integrated Security, make that that option is selected. Otherwise, select the 'Use a specific user name and password' option. Enter the User name and Password, and check the 'Allow saving password' option.

5. Select the LimsLink EI Method database in the 'Select the Database on the Server' combo box.

6. Click on the '**Test Connection**' button to confirm that the settings are correct, and a connection is successful.

Method Database ID – The Method database ID allows you to assign an ID value for the Method database. This, in turn, will be used to identify the origin of a validated Method when it is exported. Ideally this value should be a unique user-assigned value for each Method database used within an organization. The default Method database ID is 0001.

Archive Method Database Setup

An automated Method archiving option facilitates a record of the changes made from one Method version to the next. When a validated Method is imported, or a Method is validated within the system, the program will automatically archive the Method, and store it in an archive database.

To define the archive database, enable the Method validation option, then click on the **‘Configure Archive Method Database’** button. This will open a Data Link Properties window. From here, you can define a new database to store the archived Methods. You have the option to point to an existing database, or create a new database. Refer back to the Method Database Setup section for details on how to define the database.

Note: If the Method validation option is disabled at any time at which the system contains validated Methods, all validated Methods will become invalidated. Before this occurs, you will be informed of the consequences, and have the option to cancel or proceed with the disabling of the Method validation option.

When Method Validation is enabled, only validated Methods may be run unless the user checks the ‘Allow user to run unvalidated methods’ option via the Preference window.

Making Entries to the CDS Audit Trail

The Audit Trail for the CDS (if applicable and supported by the specific CDS module) can be set up such that the software will audit changes made at both setup and runtime. To do this, you must first enable the audit trail in the Preferences window by activating the check box, and then clicking on the accompanying **‘Setup’** button. For details relating to the audit trail for a specific CDS, refer to the respective CDS Module documentation.

CDS Setup

Through the CDS Configuration Setup window, you are able to establish the connection to the CDS. Access this window by clicking on the **‘Configure Connection’** button within the CDS Setup group box on the Preferences window. The Configuration Setup window is CDS-dependent, and will vary depending on the CDS you are using. The dialog title will reflect the type of CDS selected for the installation. Certain CDS’s require no additional configuration, and therefore the window will simply display an information box that details the version of the CDS found by the LimsLink EI application. For details relating to a specific CDS Configuration, refer to the respective CDS Module manual.

LIMS Setup

Through the LIMS Configuration Setup window, you are able to establish the connection to the LIMS. Access this window by clicking on the **‘Configure Connection’** button within the LIMS Setup group box on the Preference window. The Configuration Setup window is LIMS dependent, and will vary depending on the LIMS you are using. The dialog title will reflect the type of LIMS selected for the installation. Certain LIMS require no additional configuration, and therefore the window will simply display an information box that details the version of the LIMS found by the LimsLink EI application. Other LIMS will require further setup. For details relating to a specific LIMS Configuration, refer to the respective LIMS Module documentation.

Configuring a Method

Introduction

A LimsLink EI Method consists of three main parts: retrieving information from the LIMS; modifying the retrieved information to create an Expanded Sample List; and uploading the information to the CDS.

Typically, the information is retrieved from the LIMS using the settings in the LIMS Parameter Setup. The information retrieved is stored in a sample repository called the 'Sample List', which is created based on the defined CDS Parameters. The Sample List exists from the time the information is retrieved from the LIMS until the information is uploaded to the CDS. LIMS fields can be associated with CDS fields in the Sample List based on selections made in the LIMS/DS Field Name Cross-Reference Setup. Once information has been added to the Sample List, it can be manipulated, either by directly adding/deleting/editing records in the list, or by making use of the Expansion Setup. When the Sample List contains all the desired information, its contents are uploaded to the CDS to create a sequence.

Each Method contains a template script that dictates when the various components of the Method execute. For example, the script controls when the LIMS is queried, when and if the Sample and Expanded Sample Lists are displayed, and when the Expanded Sample List is uploaded to the CDS. The script, itself, may be edited to allow finer control of the process. Extra functionality may be easily added to the default script. As an example, you may add code to the scripts that attaches a prefix whenever a sample name is edited, or automatically fills in one field when another is edited.

A Method consists of the following attributes:

- Method Name
- Method Description
- LIMS Parameters
- Data System Parameters
- LIMS/DS Field Name Cross-Reference
- Expansion Setup, if applicable.

- Ability to specify the parameters that can be edited at runtime.

Method Configuration Setup

Via the CDS Client, a series of setup dialogs can be accessed through which you can set up a Method to control the information retrieval and subsequent upload.

Note: Any displays within the LimLink EI application that are related to a CDS are consistent with the particular CDS application's User Interface, allowing the user to work in a comfortable and familiar environment. The terminology used for the various dialogs and subsequent menu items is dependent on the CDS being employed, and may vary from what is depicted here.

To access these dialogs, select the **LimsLink EI - Method Setup** sub-menu item or toolbar button from within the CDS module to open the LimsLink Embedded Interface Setup window.

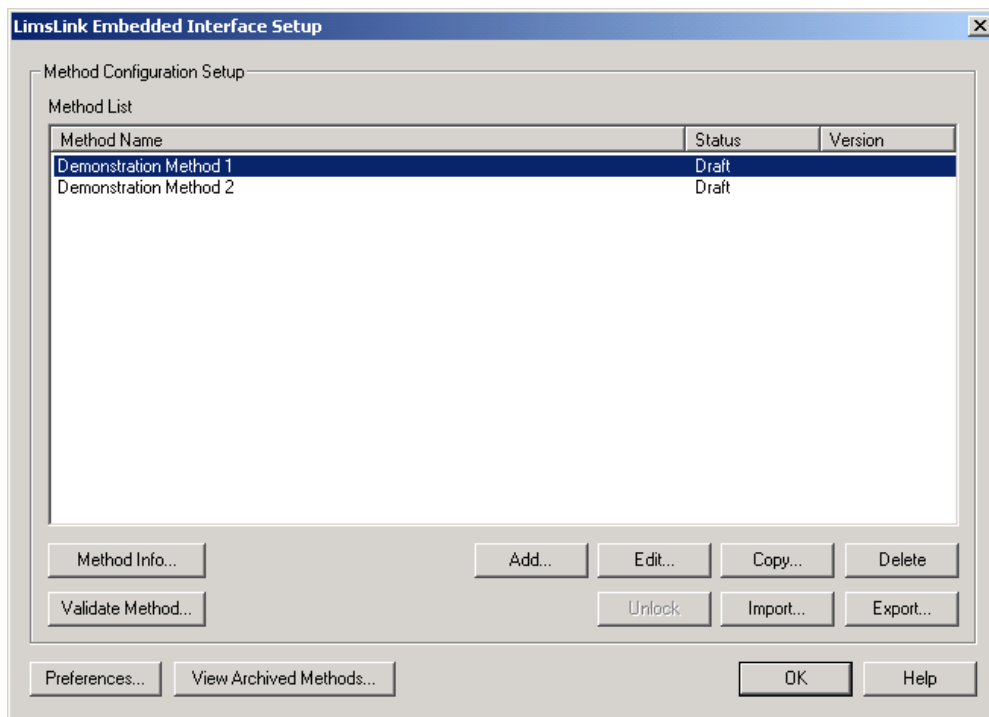


Figure 3.1: LimsLink Embedded Interface Setup

The Method List displays the name, status and version number of each Method currently configured for the system.

A Method's status can be:

Draft - a Method has not been validated. This is the default status for all Methods.

Validated – a Method has been validated. This is only applicable if Method Validation has been activated.

Locked – a Method is locked if it is being edited by another user, or is currently being run. A Method cannot be modified if it is currently 'Locked'.

Note: The Version column is only displayed if Method Validation has been activated via the Preferences window.

From the Method List, new Methods can be created, and current Methods can be modified or copied by selecting the **'Add'**, **'Edit'** or **'Copy'** buttons, respectively, to open the Method Setup window (see Figure 3.3). Methods that are no longer required can be deleted from the Method List.

The **'Import'** and **'Export'** buttons allow you to move Methods between different systems as well as back up a Method to a secure location. Selecting the **'Export'** button will allow the export of a selected Method to a file. The Export program will automatically append *.mth as the file name extension. Selection of the **'Import'** button allows the import of a previously exported Method into the system.

Note: In order to import a Method, the Method must have been created using the same CDS and LIMS modules as what currently exists in the importing system. If this is not the case, the software will not permit the importing process to proceed.

The Method List is organized in ascending alphabetical order. This order may be rearranged by selecting the appropriate column header. Subsequent clicking on the same column will toggle between ascending and descending order.

Method Info

To review information about a specific Method, highlight the Method Name from the Method List and click on the **'Method Info'** button. This will open a Method Info window, similar to the example shown in Figure 3.2.

The screenshot shows a window titled "Method Info" with a close button (X) in the top right corner. The window contains the following fields:

- Name:** A text box containing "Demonstration Method 1".
- Version:** A text box containing "0001-0001".
- Status:** A text box containing "Validated".
- Method Validation Information:** A section containing:
 - Date:** A text box containing "11/7/2007".
 - Time:** A text box containing "11:55:46 AM".
 - Validator:** A text box containing "ADMIN".
 - Comment:** A large text area with a vertical scrollbar, currently empty.
- OK:** A button at the bottom right of the window.

Figure 3.2: Method Info

The Method name, version and status are displayed. All controls are non-editable. If the selected Method is validated, a status of 'Validated' will be displayed; if the selected Method is not validated, a status of

'Draft' will be displayed. If applicable, any Method Validation Information will also be made available.

Unlocking a Method

The **'Unlock'** button is primarily to handle the case when your computer terminates unexpectedly while a Method is being edited or run. In this situation, the Method will maintain its status of 'Locked' when the program is re-started. Clicking on the **'Unlock'** button will unlock the Method and allow you to continue with your work.

A Method will also display a 'Locked' status if the Method is currently being edited at another workstation, or currently being run. It is recommended that you do not attempt to 'Unlock' the Method in this situation as this may cause the Method to be placed in an unstable state. You will be asked to confirm the unlocking process and will be warned that the Method could be in the process of being edited.

Defining the Method Setup Parameters

The Method Setup window, accessed via the **'Add'**, **'Edit'** or **'Copy'** button, allows you to specify a unique Method name, and an optional description.

Method Setup

Method name:

Method description:

☒ Use expansion:

Runtime Editing		
	Sample List	Expanded Sample List
Enable sample deletions	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Enable sample insertions	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Enable sample editing	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Enable sample sorting	<input checked="" type="checkbox"/>	

Figure 3.3: Method Setup

Methods are specific to the LIMS and CDS for which they are defined, and therefore the required parameters are dependent on the LIMS and CDS being employed. Modifying any LIMS or CDS settings will invalidate the Method.

LIMS Parameters Setup

The LIMS Setup allows you to define parameters that will dictate the Sample List that is retrieved from the respective LIMS. Refer to the individual LIMS module documentation for specific setup information related to your LIMS.

Note: Methods are specific to the LIMS for which they are defined, and therefore the required parameters are dependent on the LIMS being employed. Modifying any LIMS settings will invalidate the Method.

Data System Parameters Setup

The Data System Setup allows you to define parameters that will dictate the Sample List uploaded to the respective data system (CDS). Refer to the individual CDS module documentation for specific setup information related to your CDS.

Note: Methods are specific to the CDS for which they are defined, and therefore the required parameters are dependent on the CDS being employed. Modifying any CDS components will invalidate the Method.

LIMS/DS Field Name Cross-Reference Setup

Through the LIMS Parameters Setup and the Data System Parameters Setup windows, fields were specified to retrieve the Sample List from the LIMS, and to subsequently upload the sequence to the CDS. In most cases the LIMS field names will not directly match those requested by the CDS therefore before the actual runtime events can take place, it is necessary to map the field names of samples originating from the LIMS to the field names required by the CDS.

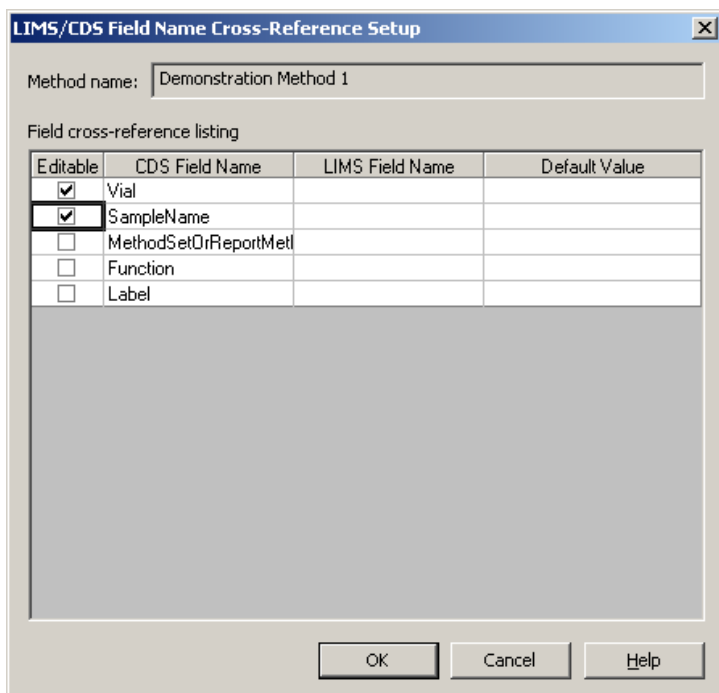


Figure 3.4: LIMS/CDS Field Name Cross-Reference Setup (Empower CDS Module used in the example)

The first column allows you to specify whether the parameter will be editable at runtime in either the Sample or Expanded Sample List.

The second column (CDS Field Name) lists all CDS fields required for the sequence. This includes any mandatory field names, as well as the optional sequence field names specified in the CDS Parameters Setup window.

The third (LIMS Field Name) column allows you to map a LIMS field name to the CDS field name. All field names specified in the LIMS Parameters Setup window are available for selection. If the LIMS field name is left blank here, a default value or the Method script may be used to provide the contents of the particular field. The user may also manually add these values at runtime if the necessary editing options are selected.

The fourth column (Default Value) allows you to define default values for each field. The default value will be used if no value is retrieved from the LIMS for the field.

Expanding the Sample List

The Sample List, acquired from the LIMS may be expanded at runtime to include replicates, as well as reference samples such as standards, controls and blanks. Once the Sample List is expanded, it is referred to as the 'Expanded Sample List'. The actual setup for the expansion is done once during Method Setup using the Expansion '**Setup**' button, and is available at runtime each time the Method is run. This eliminates the tedious task of continually entering such information at runtime, and ensures that the sequence always contains the required controls to meet your specific protocols.

Expansion Setup

Start Block

Vial	SampleName	Label	Function	Method Set / Report Method
1				
2				
3				
4				

Intermediate Block

Vial	SampleName	Label	Function	Method Set / Report Method
1				
2				
3				
4				

End Block

Vial	SampleName	Label	Function	Method Set / Report Method
1				
2				
3				
4				

Number of replicates for each sample:

Number of samples between intermediate blocks:

☐ Use random interval option

Figure 3.5: Expansion Setup (Empower CDS module used in the example)

Through the Expansion Setup window, you can add reference samples (e.g., standards or controls) at the beginning of the Sample List (Start Block), throughout the list at standard intervals (Intermediate Block) and at the end of the Sample List (End Block).

When including an intermediate block, you have the option to incorporate the block at fixed or random intervals.

Using the **'Insert Sample'** and **'Delete Sample'** buttons, you can insert a new sample, or delete the selected sample respectively. In the case of inserting a sample, the new sample will be inserted above the selected sample. The newly inserted sample will assume the default settings assigned in the CDS Setup window for the specific CDS module. Refer to your specific CDS Module manual.

The column headings are extracted from the parameters defined with the CDS Setup window for the specific CDS module.

At run-time, the software will carry out the expansion based on the setup. The expansion operations are always carried out in a particular order, and each is completely independent of the other. The expansion is executed as follows:

- 1) Replicates
- 2) Add Start Block
- 3) Add Intermediate Block
- 4) Add End Block

For example, the retrieved Sample List from the LIMS is A, B, C with number of replicates set to 3, a start block of X, Y, an intermediate block of S (fixed interval of 4), and an end block of K, L, the Expanded

Sample List would be X, Y, A, A, A, B, S, B, B, C, C, S, C, K, L. If the random interval option is used, the number of samples between each intermediate block would be a random number less than or equal to the specified interval.

Runtime Editing of Samples

During setup, you can specify what editing capabilities the user will have at runtime for both the Sample List (samples retrieved from the LIMS) and the Expanded Sample List. If you allow editing in either the Sample or Expanded Sample List, you can further define the fields that may be edited in the LIMS/CDS Field Name Cross-Reference Setup. Both the runtime editing and the individual CDS field must be enabled for editing before the field becomes editable at runtime.

For example, you can select if sample deletions, insertions, editing and sorting will be permitted at runtime. Activating the associated checkbox will permit you the capability to edit at runtime. Sorting is not permitted for the Expanded Sample List.

Editing the Script

Each time a new Method is created, a template script is generated. This script dictates when the various components of the Method will execute. For example, the script controls when the LIMS is queried, when the Sample and Expanded Sample Lists are displayed, and when the sequence is uploaded to the LIMS.

Typically, you will not change the template script; however, in special cases this may be necessary. The script gives access to the information retrieved from the LIMS as well as events when the user edits the Sample List or Expanded Sample Lists. By using the script, you have the ability to take greater control over the information before it is uploaded to the CDS.

The template script contains the predefined entry points for the runtime functions. Refer to the Programming Reference Manual in Appendix A outlining the exposed functionality within the script.

Clicking on the **'Edit Script'** button will open the script editor for the selected Method. The script editor is an Integrated Development Environment (IDE) for developing and debugging scripts. The scripting language is not explained within this manual and it is therefore recommended that you refer to the help file accessed from within the script editor for information on the programming language.

Runtime debugging

You have the option to debug the script at runtime. From the script editor window, select the **File | Run-Time Debugging** sub-menu item.

At runtime, the IDE of the script engine will open in the background allowing you to debug the script. You will have to set breakpoints in the script to stop it at various strategic intervals in order for you to gain access to the debug window.

Locking the script

You have the option to lock the script from further editing. From the script editor window, select the **File | Lock Script** sub-menu item.

If selected, a message will display stating the once the script is locked, a password will be required to unlock it. You will need to enter and confirm the password information. Clicking the **'OK'** button will save and lock the script. At this point, the script can only be accessed with the correct password.

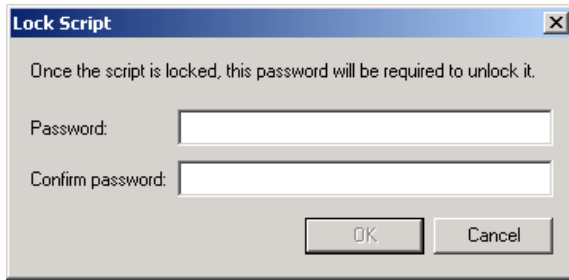


Figure 3.6: Lock Script

Method Validation and Versioning

Method Validation provides a means for the system administrator, in conjunction with in-house procedures, to ensure that each Method is properly validated and signed off by authorized personnel before use.

Method versioning provides a means for the user to track and identify the revision level of a Method. Each time a Method is validated, its version number is incremented. This allows the user to easily identify the version and suitability of a Method. The version number of any Method is made up of two individual components, the serial number of the validating system, and the Method version number. For example, a typical version number would take the form of 0123-0001, where 0123 is the Method Database ID for the validating system, and 0001 is the Method version number.

Validating a Method

There is a main check box on the Preferences window to activate the Validation system. Disabling the Validation system at any time causes all Methods in the system to be invalidated. If the system contains any validated Methods, you will be warned that all Methods will lose their validated status, and then prompted to confirm your actions. An option allowing the user to run Unvalidated Methods is also available. On the Preferences window, enable the 'Allow user to run unvalidated methods' checkbox. This facilitates the testing of new and edited Methods prior to validation.

From the Method List, located on the LimsLink Embedded Interface Setup window (see Figure 3.1), you can select a Method to validate from the list of currently invalidated Methods in the system. Once a Method is highlighted, select the **'Validate Method'** button to open the Validate Method window.

The screenshot shows a 'Validate Method' dialog box. It has a title bar with the text 'Validate Method' and a close button (X). Below the title bar, there are four input fields: 'Method name:' containing 'Demonstration Method 1', 'Version:' containing '0001-0001', 'Current user:' containing 'ADMIN', and 'Method validation comments:' which is a large empty text area with a vertical scrollbar. At the bottom of the dialog, there are four buttons: 'Password...', 'OK', 'Cancel', and 'Help'.

Figure 3.7: Validate Method

Method Name – displays the name of the Method to be validated. The Method name is non-editable.

Version Number – displays the version number of the Method to be validated. The version number is non-editable.

Current User – displays the User Name of the current user. The current user is non-editable.

Method Validation Comments – allows you to enter a comment describing the Method Validation. Here, you may want to document the procedures used to validate the Method.

Password – allows you to assign a password to the method. This will prevent other users from editing or deleting the method. A password is not required when validating a method.

Clicking on the '**OK**' button will initiate the validation process. The Method List will now display a status of 'Validated' for the selected Method.

Unarchiving Methods

An automated Method archiving option facilitates a record of the changes made from one Method version to the next. When a validated Method is imported, or a Method is validated within the system, the program will automatically archive the Method, and store it in an archive database. The specifics of the Method will be documented allowing you to recall older Methods and compare them to a current Method. Through a manual comparison of parameters, you can determine changes that have occurred from one version of a Method to the next.

From the LimsLink Embedded Interface Setup window (see Figure 3.1), access the Unarchive Method window via the '**View Archived Methods**' button.

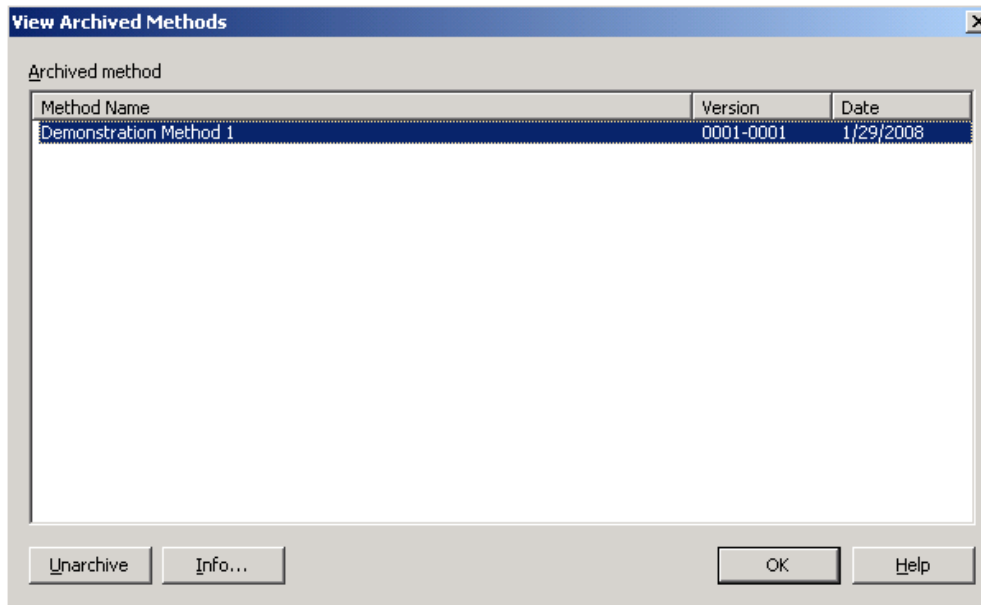


Figure 3.8: View Archived Method

Displayed is a list of all archived or previously validated Methods. From here, you can extract an archived Method from the Method Archive database into the current system. To do this, highlight the archived Method, and select the **'Unarchive'** button. Once selected, the following will occur:

If the Method is not the latest archived version of the Method, the status of the unarchived Method will be changed to 'Draft' (if the Method was validated prior to archiving).

If a Method of the same name exists within the current system, you will be prompted to rename the extracted Method.

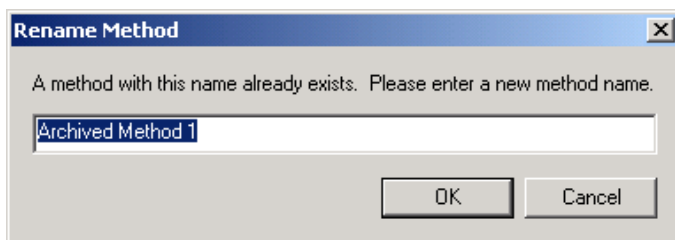


Figure 3.9: Rename Method

If the Method is renamed, the Method version number will be reset and the status will be set to 'Draft'.

To obtain details on an archived Method, highlight the Method, and select the **'Info'** button. This will open a Method Info window (see Figure 3.2) detailing the Method Name, Method Version and Validation Information for the Method.

Programming Reference

Script Entry Point Function

Syntax:

```
Public Function Execute(ByVal LIMSMModule As CDSLINKLib.ILIMSMModule, ByVal CDSModule  
As CDSLINKLib.ICDSModule, ByVal FieldMap As CDSLINKLib.FieldNameMap, ByVal  
SampleCont As CDSLINKLib.Samples, ByVal Expand As Boolean) As Boolean
```

LIMSMModule

Data Type: ILIMSMModule interface in a specific LIMS module object

Used to retrieve samples from a LIMS.

CDSModule

Data Type: ICDSModule interface in a specific CDS module object

Used to upload a sequence to a CDS.

FieldMap

Data Type: FieldNameMap object

Used to cross-reference CDS field names with LIMS field names.

SampleCont

Data Type: Samples object

Used as the sample container.

Expand

Data Type: Boolean

Indicates whether samples are to be expanded.

Returns:

A Boolean value that indicates whether the function completed successfully.

Description:

The script receives this function call when the user runs a method. This function controls the sample retrieval and sequence generation and allows the user to customize the data.

CDS Module Event Handler Methods

Description:

The following CDS module event handlers may be implemented in the script to perform custom actions when the user interacts with the Sample List and Sequence List displays at runtime.

OnSampleChange

Syntax:

```
Sub CDSModuleEvents_OnSampleChange(ByVal Source As CDSLINKLib.SampleChange,
ByVal ChangeType As CDSLINKLib.SampleChangeType, ByVal SampleIndex As Long, ByVal
SampleCont As CDSLINKLib.Samples, ByRef Changed As Boolean)
```

Source

Data Type: SampleChange enumeration

Indicates the source of the change. May be one of: SampleChange_SampleList, SampleChange_SequenceList.

ChangeType

Data Type: ICDSModule ChangeType enumeration

Indicates if a sample has been added or deleted. May be one of: ChangeType_Add, ChangeType_Delete.

SampleIndex

Data Type: Long integer

The 1-based index of the sample.

SampleCont

Data Type: Samples object

Used as the sample container.

Changed

Data Type: Boolean

An out parameter used to indicate whether the script changed anything in the sample container.

Description:

The script receives this method call when the CDS module fires an OnSampleChange event. This occurs when the user adds or deletes a sample in the Sample List or Sequence List display.

OnSampleEdit

Syntax:

```
Sub CDSModuleEvents_OnSampleEdit(ByVal Source As CDSLINKLib.SampleChange, ByVal
SampleIndex As Long, ByVal FieldIndex As Long, ByVal SampleCont As CDSLINKLib.Samples,
ByRef Changed As Boolean)
```

Source

Data Type: SampleChange enumeration

Indicates the source of the change. May be one of: SampleChange_SampleList, SampleChange_SequenceList.

SampleIndex

Data Type: Long integer

The 1-based index of the sample.

FieldIndex

Data Type: Long integer

The 1-based index of the sample field.

SampleCont

Data Type: Samples object

Used as the sample container.

Changed

Data Type: Boolean

An out parameter used to indicate whether the script changed anything in the sample container.

Description:

The script receives this method call when the CDS module fires an OnSampleEdit event. This occurs when the user edits a sample field in the Sample List or Sequence List display.

OnSampleSort*Syntax:*

```
Sub CDSModuleEvents_OnSampleSort(ByVal FieldIndex As Long, ByVal SampleCont As
CDSLINKLib.Samples, ByRef Changed As Boolean)
```

FieldIndex

Data Type: Long integer

The 1-based index of the sort field.

SampleCont

Data Type: Samples object

Used as the sample container.

Changed

Data Type: Boolean

An out parameter used to indicate whether the script changed anything in the sample container.

Description:

The script receives this method call when the CDS module fires an OnSampleSort event. This occurs when the user sorts the samples by a field in the Sample List display.

CDSLINKLib Library Objects and Interfaces

LIMS Module Object*Description:*

The common interfaces that a LIMS module object can implement are defined in the CDSLINKLib type library. Each LIMS module object is implemented in a separate DLL and has its own type library that defines interfaces and objects specific to it.

ILIMSModule Interface*Description:*

This is the required interface that all LIMS module objects implement.

DisplayName Property*Syntax:*

interface.DisplayName

interface

Data Type: ILIMSModule interface in a LIMS module object

Any object variable returning an ILIMSModule interface.

Description:

A read-only property that returns a string value that is the name of the LIMS module object and may be used for display purposes.

Configuration Property*Syntax:*

interface.Configuration

interface

Data Type: ILIMSModule interface in a LIMS module object

Any object variable returning an ILIMSModule interface.

Description:

A read/write property that sets or returns an ILIMSConfig interface to the LIMS module object's Configuration object.

FieldNames Property*Syntax:*

interface.FieldNames

interface

Data Type: ILIMSModule interface in a LIMS module object

Any object variable returning an ILIMSModule interface.

Description:

A read-only property that returns a Variant value containing an array of strings that are the field names defined in the LIMS module object.

Logon Method*Syntax:*

```
interface.Logon [userid, password]
```

interface

Data Type: ILIMSModule interface in a LIMS module object

Any object variable returning an ILIMSModule interface.

userid

Data Type: String

The user ID. This parameter is optional and defaults to an empty string.

password

Data Type: String

The user password. This parameter is optional and defaults to an empty string.

Description:

Performs a logon to the LIMS module if a user is not logged on. A script error is thrown if the user ID is blank and the password is not, or if the logon fails.

Logoff Method*Syntax:*

```
object.Logoff
```

interface

Data Type: ILIMSModule interface in a LIMS module object

Any object variable returning an ILIMSModule interface.

Description:

Performs a logoff from the LIMS module if a user is currently logged on.

GetSamples Function*Syntax:*

```
interface.GetSamples( samplecont )
```

interface

Data Type: ILIMSModule interface in a LIMS module object

Any object variable returning an ILIMSModule interface.

samplecont

Data Type: Samples object

Any object variable returning a Samples object.

Return Value:

A long integer value that indicates the status of the operation. If the return value is vbOK, the operation was successful. If the return value is vbCancel, the operation was canceled by the user.

Description:

Retrieves samples from the LIMS according to the LIMS module object's setup. A runtime display will not be shown if no options have been configured to be editable. A script error is thrown if the `samplecont` parameter is Nothing or the operation fails.

SequenceCompleted Method*Syntax:*

```
interface.SequenceCompleted status
```

interface

Data Type: ILIMSModule interface in a LIMS module object

Any object variable returning an ILIMSModule interface.

status

Data Type: Boolean

The sequence upload completion status. True if successfully completed, False if not.

Description:

Notifies the LIMS module object of the sequence upload completion status.

ICDSModule Interface*Description:*

This is the required interface that all CDS module objects implement.

DisplayName Property*Syntax:*

interface.DisplayName

interface

Data Type: ICDSModule interface in a CDS module object.

Any object variable returning an ICDSModule interface.

Description:

A read-only property that returns a string value that is the name of the CDS module object and may be used for display purposes.

Configuration Property*Syntax:*

interface.Configuration

interface

Data Type: ICDSModule interface in a CDS module object.

Any object variable returning an ICDSModule interface.

Description:

A read/write property that sets or returns an ICDSConfig interface to the CDS module object's Configuration object.

DefaultScript Property*Syntax:*

interface.DefaultScript

interface

Data Type: ICDSModule interface in a CDS module object.

Any object variable returning an ICDSModule interface.

Description:

A read-only property that returns a string value that is the default script for the CDS module object.

UserName Property*Syntax:*

interface.UserName

interface

Data Type: ICDSModule interface in a CDS module object.

Any object variable returning an ICDSModule interface.

Description:

A read-only property that returns a string value that is the name of the currently logged on user.

UserPassword Property*Syntax:*

interface.UserPassword

interface

Data Type: ICDSModule interface in a CDS module object.

Any object variable returning an ICDSModule interface.

Description:

A read-only property that returns a string value that is the password of the currently logged on user.

UserType Property*Syntax:*

interface.UserType

interface

Data Type: ICDSModule interface in a CDS module object.

Any object variable returning an ICDSModule interface.

Description:

A read-only property that returns a UserType enumeration value to indicate the type of user currently logged on. May be one of: UserType_Unknown, UserType_Admin, UserType_Normal.

GlobalFieldNames Property*Syntax:*

interface.GlobalFieldNames

interface

Data Type: ICDSModule interface in a CDS module object.

Any object variable returning an ICDSModule interface.

Description:

A read-only property that returns a Variant containing an array of strings that are the global field names for this CDS module object.

Logon Method*Syntax:*

interface.Logon [username, password]

interface

Data Type: ICDSModule interface in a CDS module object.

Any object variable returning an ICDSModule interface.

username

Data Type: String

The user ID. This parameter is optional and defaults to an empty string.

password

Data Type: String

The user password. This parameter is optional and defaults to an empty string.

Description:

Performs a logon to the CDS system if a user is not logged on. A script error is thrown if the logon fails.

Logoff Method

Syntax:

interface.Logoff

interface

Data Type: ICDSModule interface in a CDS module object.

Any object variable returning an ICDSModule interface.

sqltext

Description:

Performs a logoff from the CDS system if a user is logged on.

InitializeFields Method

Syntax:

interface.InitializeFields samplecont

interface

Data Type: ICDSModule interface in a CDS module object.

Any object variable returning an ICDSModule interface.

samplecont

Data Type: Samples object

Any object variable returning a Samples object.

Description:

Adds the required fields to the Sample template object and adds the global fields to the global Fields object in the sample container. A script error is thrown if the `samplecont` parameter is Nothing.

UploadSequence Function

Syntax:

```
interface.UploadSequence( samplecont )
```

interface

Data Type: ICDSModule interface in a CDS module object.

Any object variable returning an ICDSModule interface.

samplecont

Data Type: Sample object

Any object variable returning a Samples object.

Return Value:

A Boolean value that indicates if the sequence upload was successful.

Description:

Uploads a sequence to the CDS. A script error is thrown if the `samplecont` parameter is Nothing.

MessageToClient Method

Syntax:

```
interface.MessageToClient message
```

interface

Data Type: ICDSModule interface in a CDS module object.

Any object variable returning an ICDSModule interface.

message

Data Type: String

The message text.

Description:

Displays a user message on the CDS client.

ICDSRuntimeDisplay Interface

Description:

This is an optional interface that a CDS module object implements if it supports a runtime display.

DoRuntimeDisplay Method

Syntax:

interface.DoRuntimeDisplay

interface

Data Type: ICDSRuntimeDisplay interface in a CDS module object

Any object variable returning an ICDSRuntimeDisplay interface.

Description:

Shows the CDS module object's runtime display that allows the user to select runtime options. The runtime display will not be shown if no options have been configured to be editable.

ICDSExpansion Interface

Description:

This is an optional interface that a CDS module object implements if it supports sample expansion.

ExpandSamples Method

Syntax:

interface.ExpandSamples samplecont

interface

Data Type: ICDSExpansion interface in a CDS module object

Any object variable returning an ICDSExpansion interface.

samplecont

Data Type: Samples object

Any object variable returning a Samples object.

Description:

Expands the samples in the sample container according to the configured expansion options. A script error is thrown if the `samplecont` parameter is `Nothing` or an expansion error occurs.

ICDSExpansionSetup Interface*Description:*

This is an optional interface that a CDS module object implements if it supports sample expansion.

DoExpansionSetup Method*Syntax:*

```
interface.DoExpansionSetup retVal
```

interface

Data Type: ICDSExpansionSetup interface in a CDS module object

Any object variable returning an ICDSExpansionSetup interface.

retVal

Data Type: Integer

Description:

Displays the CDS module's expansion setup dialog. If the user presses OK `retVal` will be 1, if the user presses Cancel `retVal` will be 2.

ICDSExpansionSetup2 Interface*Description:*

This is an optional interface that a CDS module object implements if it supports the standard

sample expansion.

StartBlock Property

Syntax:

interface.StartBlock

interface

Data Type: ICDSExpansionSetup2 interface in a CDS module object

Any object variable returning an ICDSExpansionSetup2 interface.

Description:

A read/write property that sets or returns the expansion Start block sample container.

IntermediateBlock Property

Syntax:

interface.IntermediateBlock

interface

Data Type: ICDSExpansionSetup2 interface in a CDS module object

Any object variable returning an ICDSExpansionSetup2 interface.

Description:

A read/write property that sets or returns the expansion Intermediate block sample container.

EndBlock Property

Syntax:

interface.EndBlock

interface

Data Type: ICDSExpansionSetup2 interface in a CDS module object

Any object variable returning an ICDSExpansionSetup2 interface.

Description:

A read/write property that sets or returns the expansion End block sample container.

NumberOfReplicates Property

Syntax:

interface.NumberofReplicates

interface

Data Type: ICDSExpansionSetup2 interface in a CDS module object

Any object variable returning an ICDSExpansionSetup2 interface.

Description:

A read/write property that sets or returns the number of replicates to create for each sample.

NumberOfSamplesBtwnIntermediateBlocks Property

Syntax:

interface.NumberOfSamplesBtwnIntermediateBlocks

interface

Data Type: ICDSExpansionSetup2 interface in a CDS module object

Any object variable returning an ICDSExpansionSetup2 interface.

Description:

A read/write property that sets or returns the number of samples between intermediate blocks.

UseRandomInterval Property

Syntax:

interface.UseRandomInterval

interface

Data Type: ICDSExpansionSetup2 interface in a CDS module object

Any object variable returning an ICDSExpansionSetup2 interface.

Description:

A read/write property that sets or returns the random interval option.

ICDSSampleDisplay Interface*Description:*

This is an optional interface that a CDS module object implements if it supports a sample display.

AllowDeleteSampleList Property*Syntax:*

interface.AllowDeleteSampleList

interface

Data Type: ICDSSampleDisplay interface in a CDS module object

Any object variable returning an ICDSSampleDisplay interface.

Description:

A read/write property that sets or returns a Boolean value that determines if the user is allowed to delete samples in the Sample List display.

AllowInsertSampleList Property*Syntax:*

interface.AllowInsertSampleList

interface

Data Type: ICDSSampleDisplay interface in a CDS module object

Any object variable returning an ICDSSampleDisplay interface.

Description:

A read/write property that sets or returns a Boolean value that determines if the user is allowed to insert samples in the Sample List display..

AllowEditSampleList Property*Syntax:*

```
interface.AllowEditSampleList
```

interface

Data Type: ICDSSampleDisplay interface in a CDS module object

Any object variable returning an ICDSSampleDisplay interface.

Description:

A read/write property that sets or returns a Boolean value that determines if the user is allowed to edit samples in the Sample List display..

AllowSortSampleList Property*Syntax:*

```
interface.AllowSortSampleList
```

interface

Data Type: ICDSSampleDisplay interface in a CDS module object

Any object variable returning an ICDSSampleDisplay interface.

Description:

A read/write property that sets or returns a Boolean value that determines if the user is allowed to sort samples in the Sample List display..

DoSampleDisplay Method*Syntax:*

```
interface.DoSampleDisplay samplecont
```

interface

Data Type: ICDSSampleDisplay interface in a CDS module object

Any object variable returning an ICDSSampleDisplay interface.

samplecont

Data Type: Samples object

Any object variable returning a Samples object.

Description:

Shows the Sample List display. A script error is thrown if the `samplecont` parameter is Nothing.

ICDSSequenceDisplay Interface

Description:

This is an optional interface that a CDS module object implements if it supports a sequence display.

AllowDeleteSequenceList Property

Syntax:

`interface.AllowDeleteSequenceList`

interface

Data Type: ICDSSequenceDisplay interface in a CDS module object

Any object variable returning an ICDSSequenceDisplay interface.

Description:

A read/write property that sets or returns a Boolean value that determines if the user is allowed to delete samples in the Sequence List display..

AllowInsertSequenceList Property

Syntax:

`interface.AllowInsertSequenceList`

interface

Data Type: ICDSSequenceDisplay interface in a CDS module object

Any object variable returning an ICDSSequenceDisplay interface.

Description:

A read/write property that sets or returns a Boolean value that determines if the user is allowed to insert samples in the Sequence List display.

AllowEditSequenceList Property*Syntax:*

interface.AllowEditSequenceList

interface

Data Type: ICDSSequenceDisplay interface in a CDS module object

Any object variable returning an ICDSSequenceDisplay interface.

Description:

A read/write property that sets or returns a Boolean value that determines if the user is allowed to edit samples in the Sequence List display.

DoSequenceDisplay Method*Syntax:*

interface.DoSequenceDisplay samplecont

interface

Data Type: ICDSSequenceDisplay interface in a CDS module object

Any object variable returning an ICDSSequenceDisplay interface.

samplecont

Data Type: Samples object

Any object variable returning a Samples object.

Description:

Shows the Sequence List display. A script error is thrown if the samplecont parameter is Nothing.

Samples Object*Description:*

This is the sample container collection used to retrieve samples from the LIMS module and upload a sequence to the CDS module. The samples in the collection may be enumerated using the For Each...Next syntax.

Item Property*Syntax:*

```
object.Item( key, keytype )
```

```
object( key, keytype )
```

object

Data Type: Samples object

Any object variable returning a Samples object.

key

Data Type: Long integer

The 1-based index or 1-based record number of the sample.

keytype

Data Type: SampleKeyType enumeration

Determines the type of the key value. May be one of: SampleKeyType_Index or SampleKeyType_RecordNum.

Description:

A read-only property that returns a Sample object from the collection. A script error is thrown if the sample index or record number is out of range.

Count Property*Syntax:*

```
object.Count
```

object

Data Type: Samples object

Any object variable returning a Samples object.

Description:

A read-only property that returns a long integer value that is the number of samples in the collection.

GlobalFields Property

Syntax:

object.GlobalFields

object

Data Type: Samples object

Any object variable returning a Samples object.

Description:

A read-only property that returns a Fields object that holds a collection of global fields.

SampleTemplate Property

Syntax:

object.SampleTemplate

object

Data Type: Samples object

Any object variable returning a Samples object.

Description:

A read-only property that returns a Sample object that may be used as a template for adding or inserting new samples to the collection.

AddSample Function

Syntax:

object.AddSample()

object

Data Type: Samples object

Any object variable returning a Samples object.

Return Value:

The new Sample object.

Description:

Adds a new empty Sample object to the end of the collection. The Sample object does not contain any fields. A script error is thrown if the operation fails.

AddSampleFromTemplate Function*Syntax:*

```
object.AddSampleFromTemplate()
```

object

Data Type: Samples object

Any object variable returning a Samples object.

Return Value:

The new Sample object.

Description:

Adds a new Sample object to the collection. The Sample object is a duplicate of the template Sample object. A script error is thrown if the operation fails.

InsertSample Function*Syntax:*

```
object.InsertSample( key, keytype )
```

object

Data Type: Samples object

Any object variable returning a Samples object.

key

Data Type: Long integer

The 1-based index or 1-based record number of the sample.

keytype

Data Type: SampleKeyType enumeration

Determines the type of the key value. May be one of: SampleKeyType_Index or SampleKeyType_RecordNum.

Return Value:

The new Sample object.

Description:

Inserts a new empty Sample object into the collection before the sample key position. The Sample object does not contain any fields. A script error is thrown if the index or record number is out of range, or if the operation fails.

InsertSampleFromTemplate Function*Syntax:*

```
object.InsertSampleFromTemplate( key, keytype )
```

object

Data Type: Samples object

Any object variable returning a Samples object.

key

Data Type: Long integer

The 1-based index or 1-based record number of the sample.

keytype

Data Type: SampleKeyType enumeration

Determines the type of the key value. May be one of: SampleKeyType_Index or SampleKeyType_RecordNum.

Return Value:

The new Sample object.

Description:

Inserts a new Sample object into the collection before the sample key position. The Sample object is a duplicate of the template Sample object. A script error is thrown if the index or record number is out of range, or if the operation fails.

RemoveSample Method*Syntax:*

```
object.RemoveSample
```

object

Data Type: Samples object

Any object variable returning a Samples object.

key

Data Type: Long integer

The 1-based index or 1-based record number of the sample.

keytype

Data Type: SampleKeyType enumeration

Determines the type of the key value. May be one of: SampleKeyType_Index or SampleKeyType_RecordNum.

Description:

Removes a sample from the collection at the sample key position. A script error is thrown if the index or record number is out of range.

RemoveAllSamples Method*Syntax:*

```
object.RemoveAllSamples
```

object

Data Type: Samples object

Any object variable returning a Samples object.

Description:

Removes all samples from the collection.

GetNextRecord Function*Syntax:*

```
object.GetNextRecord( recnum )
```

object

Data Type: Samples object

Any object variable returning a Samples object.

recnum

Data Type: Long integer

The record number of a reference sample.

Return Value:

A Sample object

Description:

Retrieves the sample with the next record number relative to the reference sample. If the `recnum` parameter is zero, the first sample in the collection will be returned. If there is no next sample or the collection is empty, a value of Nothing is returned.

GetPrevRecord Function*Syntax:*

```
object.GetPrevRecord()
```

object

Data Type: Samples object

Any object variable returning a Samples object.

recnum

Data Type: Long integer

The record number of a reference sample.

Return Value:

A Sample object.

Description:

Retrieves the sample with the previous record number relative to the reference sample. If the `recnum` parameter is zero, the last sample in the collection will be returned. If there is no previous sample or the collection is empty, a value of `Nothing` is returned.

AutoNumberField Method

Syntax:

```
object.AutoNumberField fieldkey, keytype, initialval, increment
```

object

Data Type: Samples object

Any object variable returning a Samples object.

fieldkey

Data Type: Variant

The 1-based index or CDS name of a sample field.

keytype

Data Type: FieldKeyType enumeration

Determines the type of the field key value. May be one of: `FieldKeyType_Index`, `FieldKeyType_Name`.

initialval

Data Type: Long integer

The initial value.

increment

Data Type: Long integer

The increment value.

Description:

Auto numbers a field in the all samples starting with the initial value and incrementing by the increment value. A script error is thrown if a sample does not contain the specified field.

AddField Method*Syntax:*

```
object.AddField cdsname [, permanent]
```

object

Data Type: Samples object

Any object variable returning a Samples object.

cdsname

Data Type: String

The CDS name of the new sample field.

permanent

Data Type: Boolean

Determines if the sample field is permanent. This parameter is optional and defaults to False.

Description:

Adds a new field to every sample in the collection and to the sample template. A script error is thrown if the operation fails.

RemoveField Method*Syntax:*

```
object.RemoveField fieldkey, keytype
```

object

Data Type: Samples object

Any object variable returning a Samples object.

fieldkey

Data Type: Variant

The 1-based index or CDS name of a sample field.

keytype

Data Type: FieldKeyType enumeration

Determines the type of the field key value. May be one of: FieldKeyType_Index, FieldKeyType_Name.

Description:

Removes a field from every sample in the collection and from the sample template. A script error is thrown if a sample or sample template does not contain the field or if the field is marked as permanent.

SortField Method*Syntax:*

object.SortField fieldkey, keytype, order

object

Data Type: Samples object

Any object variable returning a Samples object.

fieldkey

Data Type: Variant

The 1-based index or CDS name of a sample field.

keytype

Data Type: FieldKeyType enumeration

Determines the type of the field key value. May be one of: FieldKeyType_Index, FieldKeyType_Name.

order

Data Type: SortOrder enumeration

Determines the sorting order. May be one of: SortOrder_Ascending, SortOrder_Descending.

Description:

Sorts the samples in the collection by the specified field and ordering. A script error is thrown if a sample does not contain the field.

Sample Object*Description:*

This is the sample container collection used to retrieve samples from the LIMS module and upload a sequence to the CDS module.

RecordNum Property*Syntax:*

object.RecordNum

object

Data Type: Sample object

Any object variable returning a Sample object.

Description:

A read-only property that returns the sample's record number. This value is generated by the Samples object when a sample is added or inserted to the collection and is unique for a given instance of a Samples object.

Fields Property*Syntax:*

object.Fields

object

Data Type: Sample object

Any object variable returning a Sample object.

Description:

A read-only property that returns the Fields object.

Editable Property*Syntax:*

object.Editable

object

Data Type: Sample object

Any object variable returning a Sample object.

Description:

A read/write property that sets or returns Boolean value that determines if the sample may be edited in the Sample List and Sequence List displays.

Fields Object*Description:*

This is the collection of fields for a single sample. The fields in the collection may be enumerated using the For Each...Next syntax.

Item Property*Syntax:*

```
object.Item( key, keytype )
```

```
object( key, keytype )
```

object

Data Type: Fields object

Any object variable returning a Fields object.

key

Data Type: Variant

The 1-based index or CDS name of the field.

keytype

Data Type: FieldKeyType enumeration

Determines the type of the key value. May be one of FieldKeyType_Index or FieldKeyType_Name.

Description:

A read-only property that returns a Field object from the collection. A script error is thrown if the field index is out of range or the CDS name is invalid.

Count Property

Syntax:

object.Count

object

Data Type: Fields object

Any object variable returning a Fields object.

Description:

A read-only property that returns a long integer value that is the number of fields in the collection.

AddField Function

Syntax:

object.AddField(cdsname [,permanent])

object

Data Type: Fields object

Any object variable returning a Fields object.

cdsname

Data Type: String

The CDS name of the new field.

permanent

Data Type: Boolean

Determines if the field is permanent. This parameter is optional and defaults to False.

Return Value:

The new Field object.

Description:

Adds a new field to the end of the collection. A script error is thrown if a field with the specified CDS name already exists in the collection or if the operation fails.

RemoveField Method*Syntax:*

object.RemoveField key, keytype

object

Data Type: Fields object

Any object variable returning a Fields object.

key

Data Type: Variant

The 1-based index or CDS name of the field.

keytype

Data Type: FieldKeyType enumeration

Determines the type of the key value. May be one of FieldKeyType_Index or FieldKeyType_Name.

Description:

Removes a field from the collection. A script error is thrown if the field index is out of range or CDS name is invalid, or if the field is marked as permanent.

RemoveAllFields Method*Syntax:*

object.RemoveAllFields

object

Data Type: Fields object

Any object variable returning a Fields object.

Description:

Removes all fields from the collection.

ExistsField Function

Syntax:

```
object.ExistsField( key, keytype )
```

object

Data Type: Fields object

Any object variable returning a Fields object.

key

Data Type: Variant

The 1-based index or CDS name of the field.

keytype

Data Type: FieldKeyType enumeration

Determines the type of the key value. May be one of FieldKeyType_Index or FieldKeyType_Name.

Return Value:

A Boolean value that indicates if the field exists in the collection. True if it exists, False if not.

Description:

Determines if a field exists in the collection.

Field Object

Description:

This is a single field in a sample.

Value Property

Syntax:

```
object.Value
```

```
object
```

object

Data Type: Field object

Any object variable returning a Field object.

Description:

A read-only property that returns a string that is the value of the field.

CDSName Property*Syntax:*

object.CDSName

object

Data Type: Field object

Any object variable returning a Field object.

Description:

A read-only property that returns a string that is the CDS name of the field.

Permanent Property*Syntax:*

object.Permanent

object

Data Type: Field object

Any object variable returning a Field object.

Description:

A read-only property that returns a Boolean value that indicates if the field is permanent. A permanent field cannot be removed from the fields collection.

Hidden Property*Syntax:*

object.Hidden

object

Data Type: Field object

Any object variable returning a Field object.

Description:

A read-write property that returns a Boolean value that determines if the field is shown in the Sample List and Sequence List displays at runtime.

FieldNameMap Object*Description:*

This is the collection of items that is used to cross-reference CDS field names with LIMS field names. The items in the collection may be enumerated using the For Each...Next syntax.

Item Property*Syntax:*

```
object.Item( key, keytype )
```

```
object( key, keytype )
```

object

Data Type: FieldNameMap object

Any object variable returning a FieldNameMap object.

key

Data Type: Variant

The 1-based index, CDS field name or LIMS field name of the map item.

keytype

Data Type: FieldMapKeyType enumeration

Determines the type of the key value. May be one of FieldNameKeyType_Index, FieldNameKeyType_CDSName or FieldNameKeyType_LIMSName.

Description:

A read-only property that returns a FieldNameMapItem object from the collection. A script error is thrown if the item index is out of range or if the CDS field name or LIMS field name is invalid.

Count Property

Syntax:

object.Count

object

Data Type: FieldNameMap object

Any object variable returning a FieldNameMap object.

Description:

A read-only property that returns a long integer value that is the number of items in the collection.

AddField Function

Syntax:

object.AddField(cdsname, limsname)

object

Data Type: FieldNameMap object

Any object variable returning a FieldNameMap object.

cdsname

Data Type: String

The CDS name of the new map item.

limsname

Data Type: String

The LIMS name of the new map item.

Return Value:

The new FieldNameMapItem object.

Description:

Adds a new field name map item to the end of the collection. Multiple CDS field names may be mapped to the same LIMS field name. A script error is thrown if an item with the CDS field name

already exists in the collection or if the operation fails.

RemoveField Method

Syntax:

```
object.RemoveField key, keytype
```

object

Data Type: FieldNameMap object

Any object variable returning a FieldNameMap object.

key

Data Type: Variant

The 1-based index, CDS field name or LIMS field name of the map item.

keytype

Data Type: FieldMapKeyType enumeration

Determines the type of the key value. May be one of FieldNameKeyType_Index, FieldNameKeyType_CDSName or FieldNameKeyType_LIMSName.

Description:

Removes one or more field name map items from the collection. A script error is thrown if the item index is out of range or if the CDS field name or LIMS field name is invalid.

RemoveAllFields Method

Syntax:

```
object.RemoveAllFields
```

object

Data Type: FieldNameMap object

Any object variable returning a FieldNameMap object.

Description:

Removes all field name map items from the collection.

FieldNameMapItem Object*Description:*

This item maps a single CDS field name to a LIMS field name.

CDSFieldName Property*Syntax:*

object.CDSFieldName

object

Data Type: FieldNameMapItem object

Any object variable returning a FieldNameMapItem object.

Description:

A read/write property that sets or returns a string that is the CDS field name.

LIMSFieldName Property*Syntax:*

object.LIMSFieldName

object

Data Type: FieldNameMapItem object

Any object variable returning a FieldNameMapItem object.

Description:

A read/write property that sets or returns a string that is the LIMS field name.

AllowEdit Property*Syntax:*

object.AllowEdit

object

Data Type: FieldNameMapItem object

Any object variable returning a FieldNameMapItem object.

Description:

A read/write property that sets or returns a Boolean value that determines if the user is allowed to edit this CDS field in the Sample List and Sequence List displays.

LIMSParams Object*Description:*

This is the collection of setup parameters for a the generic SQL Query LIMS module. The parameters in the collection may be enumerated using the For Each...Next syntax.

Item Property*Syntax:*

object.Item(key, keytype)

object(key, keytype)

object

Data Type: LIMSParams object

Any object variable returning a LIMSParams object.

key

Data Type: Variant

The 1-based index of the parameter or the parameter name.

Description:

A read-only property that returns a LIMSParam object from the collection. A script error is thrown if the parameter index is out of range or if the parameter name is invalid.

Count Property*Syntax:*

object.Count

object

Data Type: LIMSParams object

Any object variable returning a LIMSParams object.

Description:

A read-only property that returns a long integer value that is the number of parameters in the collection.

Add Function*Syntax:*

```
object.Add( name )
```

object

Data Type: LIMSParams object

Any object variable returning a LIMSParams object.

name

Data Type: String

The parameter name.

Return Value:

The new LIMSParam object.

Description:

Adds a new parameter to the end of the collection. A script error is thrown if a parameter with the name already exists in the collection or if the operation fails.

Remove Method*Syntax:*

```
object.Remove name
```

object

Data Type: LIMSParams object

Any object variable returning a LIMSParams object.

name

Data Type: String

The parameter name.

Description:

Removes a parameter from the collection. An script error is thrown if the parameter name is invalid.

RemoveAll Method

Syntax:

object.RemoveAll

object

Data Type: LIMSParams object

Any object variable returning a LIMSParams object.

Description:

Removes all parameters from the collection.

Exists Function

Syntax:

object.Exists(name)

object

Data Type: LIMSParams object

Any object variable returning a LIMSParams object.

name

Data Type: String

The parameter name.

Return Value:

A Boolean value that indicates if a parameter with the specified named exists in the collection.
True if it exists, False if not.

Description:

Determines if a parameter exists in the collection.

LIMSParam Object*Description:*

This is a setup parameter for the generic SQL Query LIMS module.

Value Property*Syntax:*

object.Value

object

object

Data Type: LIMSParam object

Any object variable returning a LIMSParam object.

Description:

A read/write property that sets or returns a string that is the SQL value of the parameter.

Name Property*Syntax:*

object.Name

object

Data Type: LIMSParam object

Any object variable returning a LIMSParam object.

Description:

A read-only property that returns a string that is the name of the parameter.

DefaultValue Property*Syntax:*

object.DefaultValue

object

Data Type: LIMSPARAM object

Any object variable returning a LIMSPARAM object.

Description:

A read/write property that sets or returns the default list value of the parameter. This property is ignored at runtime if the ParamListFromQuery property is set to True.

ParameterList Property*Syntax:*

object.ParameterList

object

Data Type: LIMSPARAM object

Any object variable returning a LIMSPARAM object.

Description:

A read-only property that returns the LIMSPARAMList object. This property is ignored at runtime if the ParamListFromQuery property is set to True.

AllowEdit Property*Syntax:*

object.AllowEdit

object

Data Type: LIMSPARAM object

Any object variable returning a LIMSPARAM object.

Description:

A read/write property that sets or returns a Boolean value that determines if the user is allowed to change this parameter value in the Retrieve Samples display.

ParamListFromQuery Property*Syntax:*

object.ParamListFromQuery

object

Data Type: LIMSParam object

Any object variable returning a LIMSParam object.

Description:

A read/write property that sets or returns a Boolean value that determines if this parameter generates a parameter list from a SQL query.

QueryText Property*Syntax:*

object.QueryText(password)

object

Data Type: LIMSParam object

Any object variable returning a LIMSParam object.

password

Data Type: String

The query text password. An empty string if the query is not locked.

Description:

A read/write property that sets or returns a string that is the SQL query text. This property is ignored at runtime if the ParamListFromQuery property is set to False. A script error is thrown if the query is locked and the password is invalid.

FirstQueryItemAsDefault Property*Syntax:*

object.FirstQueryItemAsDefault

object

Data Type: LIMSPParam object

Any object variable returning a LIMSPParam object.

Description:

A read/write property that sets or returns a Boolean value that determines if the first item returned by a SQL query is used as the default list value for this parameter. This property is ignored at runtime if the ParamListFromQuery property is set to False.

QueryLocked Property*Syntax:*

object.QueryLocked

object

Data Type: LIMSPParam object

Any object variable returning a LIMSPParam object.

Description:

A read-only property that returns a Boolean value that indicates if the query is locked. True if locked, False if not.

LockQuery Method*Syntax:*

object.LockQuery password

object

Data Type: LIMSPParam object

Any object variable returning a LIMSPParam object.

password

Data Type: String

The query password.

Description:

Locks the query text with a password. A script error is thrown if the password is an empty string or if the query is already locked.

UnlockQuery Method*Syntax:*

```
object.UnlockQuery password
```

object

Data Type: LIMSPParam object

Any object variable returning a LIMSPParam object.

password

Data Type: String

The query password.

Description:

Unlocks the query text with a password. A scrip error is thrown if the query text is not locked or if the password is invalid.

ILIMSPParam2 Interface*Description:*

This is an interface that is implemented on the LIMSPParam object.

GetAllowOtherValuesAtRuntime Function*Syntax:*

```
interface.GetAllowOtherValuesAtRuntime
```

interface

Data Type: LIMSPARAM object

Any object variable returning an ILIMSPARAM2 interface e.

Description:

Returns True if the “allow other values at runtime” option is enabled for the parameter, False otherwise.

PutAllowOtherValuesAtRuntime Function

Syntax:

interface.PutAllowOtherValuesAtRuntime

interface

Data Type: LIMSPARAM object

Any object variable returning an ILIMSPARAM2 interface.

Description:

Enables or disables the “allow other values at runtime” option for the parameter.

GetAllowMultipleSelections Function

Syntax:

interface.GetAllowMultipleSelections

interface

Data Type: LIMSPARAM object

Any object variable returning an ILIMSPARAM2 interface.

Description:

Returns True if the “allow multiple selections” option is enabled for the parameter, False otherwise.

PutAllowMultipleSelections Function

Syntax:

```
interface.PutAllowMultipleSelections
```

interface

Data Type: LIMSPParam object

Any object variable returning an ILIMSPParam2 interface.

Description:

Enables or disables the “allow multiple selections” option for the parameter.

LIMSPParamsList Object*Description:*

This is the collection of list value / SQL value pairs for a parameter. The items in the collection may be enumerated using the For Each...Next syntax.

Item Property*Syntax:*

```
object.Item( key, keytype )
```

```
object( key, keytype )
```

object

Data Type: LIMSPParamList object

Any object variable returning a LIMSPParamList object.

key

Data Type: Variant

The 1-based index or name of the parameter list item.

Description:

A read-only property that returns a LIMSParamList. A script error is thrown if the item index is out of range or if the item name is invalid.

Count Property*Syntax:*

object.Count

object

Data Type: LIMSParamList object

Any object variable returning a LIMSParamList object.

Description:

A read-only property that returns a long integer value that is the number of items in the collection.

Add Function*Syntax:*

object.Add(name)

object

Data Type: LIMSParamList object

Any object variable returning a LIMSParamList object.

sqltext

Data Type: String

The parameter list item name.

Return Value:

The new LIMSParamListItem object.

Description:

Add a new parameter list item to the collection. A script error is thrown if an item with the specified name already exists in the collection.

Remove Method

Syntax:

```
object.Remove name
```

object

Data Type: LIMSParamList object

Any object variable returning a LIMSParamList object.

name

Data Type: String

The parameter list item name.

Description:

Removes a parameter list item from the collection. A script error is thrown if the item name is invalid.

RemoveAll Method

Syntax:

```
object.RemoveAll
```

object

Data Type: LIMSParamList object

Any object variable returning a LIMSParamList object.

Description:

Removes all items from the collection.

Exists Function

Syntax:

```
object.Exists
```

object

Data Type: LIMSParamList object

Any object variable returning a LIMSParmList object.

name

Data Type: String

The parameter list item name.

Return Value:

A Boolean value that indicates if an item with the specified name exists in the collection.

Description:

Executes an arbitrary SQL statement.

LIMSParmListItem Object*Description:*

This is a list value / SQL value pair.

Name Property*Syntax:*

object.Name

object

Data Type: LIMSParmListItem object

Any object variable returning a LIMSParmListItem object.

Description:

A read-only property that returns a string that is the name of the parameter list item (i.e. list value).

AltName Property*Syntax:*

object.AltName

object

Data Type: LIMSParmListItem object

Any object variable returning a LIMSParmListItem object.

Description:

A read-only property that returns a string that is the alternate name of the parameter list item (i.e. SQL parameter value).

LimsLink EI™

Version 4.2

CDS Modules



Chromeleon

Introduction

Overview

LimsLink EI, a component of the Labtronics LimsLink CDS product, provides an embedded link between your CDS and LIMS. Designed to be a Plug & Play system, you can substitute in various CDS and LIMS modules to work with the interface, and set up the individual modules to suit your requirements.

The Chromeleon module is suitable for use with Chromeleon v6.8 and is compatible with any LIMS module, selected during installation.

The Chromeleon module provides the functionality required to view sample lists retrieved from the LIMS, edit and expand the sample list to include control samples, and then convert this list to a Sequence for analysis by the Chromeleon application. Once analysis is complete, results can be reported back to the LIMS.

As part of the LimsLink EI installation procedure, you are required to select the CDS and LIMS module for the installation. Upon selecting the Chromeleon CDS, the Chromeleon module is then installed as part of your LimsLink EI installation. Toolbar buttons are embedded in the Chromeleon client to provide access to LimsLink EI.

With the LimsLink EI application, LIMS integration is embedded directly into your Chromeleon application providing users with a direct and transparent connection to their LIMS.

Installation

Software Requirements

The LimsLink EI application requires that the following software be installed, and operational, prior to installation:

- Chromeleon v6.8

- Chromeleon SDK

Note: If LimsLink EI is installed whereby the Chromeleon Clients have been deployed in a 'Distributed User' environment, the PATH variable on the PC will need to be modified.

The PATH environment variable will require that the full Chromeleon\bin directory be present.

For example, if Chromeleon is installed to the "C:\Chromel" directory, then the following would need to be added to the PATH variable of the PC:

"C:\Chromel\bin"

The PC Client will need to be rebooted in order for these changes to take effect.

Security

The **LimsLink EI – Method Setup** and **LimsLink EI – Create Sequence** buttons are controlled by the Chromeleon security system, and are therefore restricted to the appropriate users. This allows only certain users to access Method Setup, and other users access only to the Create Sequence button. If the Chromeleon security system is enabled, all users that are members of the 'LimsLink EI Admins' Access Group in Chromeleon will have access to the Method Setup button. All valid Chromeleon users will have access to the Create Sequence button.

References

The documentation for the Chromeleon module only refers to that information which pertains to the direct setup and runtime of the LimsLink EI application with respect to the Chromeleon application. You will need to refer to the LimsLink EI User Manual for specifics relating to the setup of the application as a whole.

System Setup

A set of parameters governing the transfer of information between the LIMS and the CDS must first be established before information retrieval and analysis can occur. This information would typically be configured by someone with administrative privileges, and would only need to be configured once during the initial setup.

Accessing the System Setup

To access the system parameters, perform the following steps:

1. From the Chromeleon application, click on the LimsLink EI – Method Setup toolbar button to open the LimsLink Embedded Interface Setup window, similar to the example shown here:

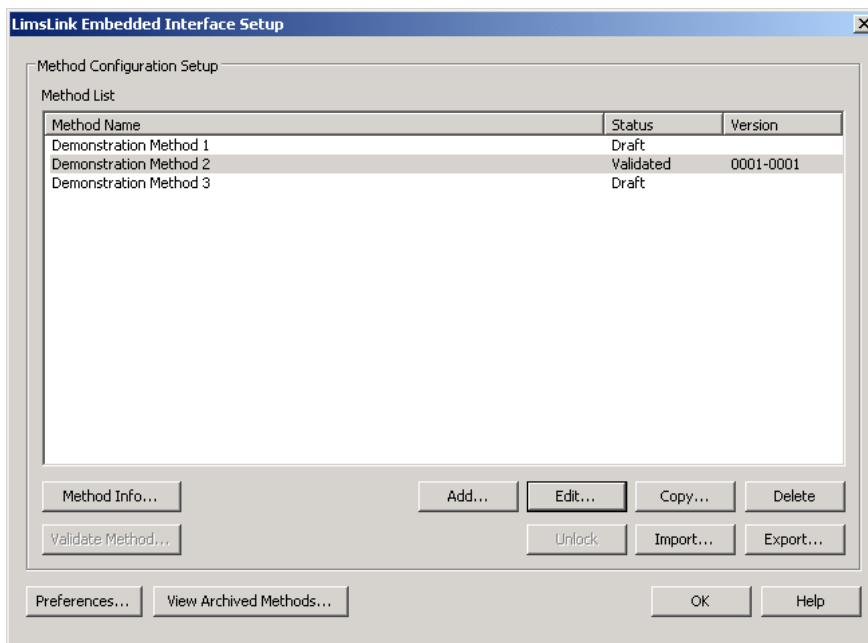


Figure -1: LimsLink Embedded Interface Setup

- Click on the **'Preferences'** button on the LimsLink Embedded Interface Setup window to open the Preference window, similar to the example shown here.

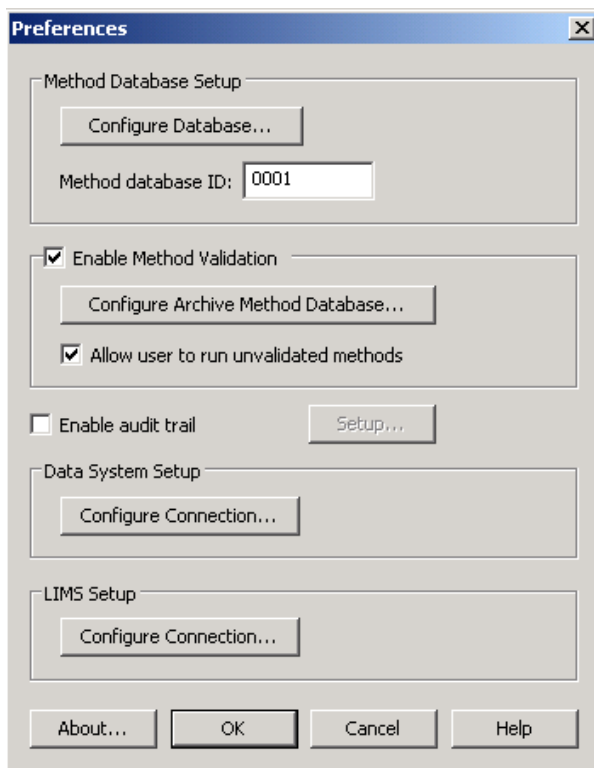


Figure 2: Preferences

System Preferences

Through this window, you are able to determine the location of data repositories such as the Method Database and the Archive Method Database. Other specifics such as Method Validation can be established here. Refer to Chapter 2 - System Setup in the LimsLink EI User Manual for additional details.

Chromeleon Configuration Setup

Click on the '**Configure Connection**' button within the Data System Setup group box to open the Chromeleon Configuration Setup window. The Chromeleon application requires no additional configuration, and therefore the window will simply display an information box.

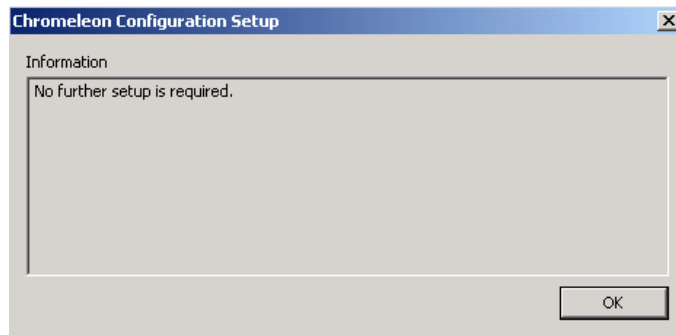


Figure 3: Chromeleon Configuration Setup

Defining the CDS Parameters

Method Setup

Recall that a LimsLink EI Method is responsible for retrieving information from the LIMS, modifying the retrieved information, and subsequently uploading the information to the CDS. As part of the Method setup, the parameters for the respective CDS being used need to be defined in order for the information to be uploaded to the CDS.

Accessing the Chromeleon Setup

1. To access the Method Setup window, click on the '**Add**' button from the LimsLink Embedded Interface Setup window.
2. Click on the '**Data System Parameters Setup**' button to open the Chromeleon Setup window.

Chromeleon Setup

Method name:

Sequence Parameters

Sequence name: ☒ Edit

Data source: ☒ Edit

Time base: ☒ Edit

Report: ☒ Edit

Sequence path: ☒ Edit

☒ Allow edit default Program

☒ Allow edit default Method

☒ Use logon parameters to create sequence

Sequence Fields

Optional Sequence Field Names	
<input type="checkbox"/>	Type
<input type="checkbox"/>	Inj. Vol.
<input type="checkbox"/>	Status
<input type="checkbox"/>	Weight
<input type="checkbox"/>	Dil. Factor
<input type="checkbox"/>	ISTD Amount
<input type="checkbox"/>	Sample ID
<input type="checkbox"/>	Replicate ID

OK Cancel Help

Figure 4: Chromeleon Setup

Setting up Chromeleon

The Chromeleon Setup allows you to define the information that is required to create the Sequence (i.e., the list of samples) to be analyzed by the Chromeleon CDS. The Sequence Parameters section establishes parameters concerning the sequences that will be created. The Sequence Fields section allows you to select the fields that will be uploaded to the CDS.

Specifying the Method Name

Method Name – The name of the current Method is displayed. It is not editable.

Defining the Sequence Parameters

Sequence Name

Here you can enter a name for the sequence.

The sequence name may not contain the following characters:

/ \ " ' < | > ? *

You have the option to include a macro variable that will be replaced by LimsLink EI at runtime, or you can enter a fixed sequence name that will be applicable to all sequences created using the selected Method.

The following macros are available:

{date} – the current system date (YYMMDD)

{time} – the current system time (HHMMSS)

{counter} – a 2-digit incremental counter

{userid} – the currently logged in Chromeleon user. If security is not enabled in Chromeleon, this will be replaced with the Windows user name.

You may permit runtime editing of the sequence name selection by checking the associated Edit check box.

Data source

From here, you can select the Chromeleon data source to which to connect. All data sources, supplied by the Chromeleon application applicable to the current user, are available for selection.

You may permit runtime editing of the database selection by checking the associated Edit check box.

Time base

Here you can enter a time base. The time base name may not contain the following characters:

/ \ " ' . < | > ? *

Alternatively, you can browse for a time base by selecting the associated browse button.

You may permit runtime editing of the time base selection by checking the associated Edit check box.

Report

Here you can specify a report definition file. The report definition file may not contain the following characters:

/ " ' < | > ? *

Alternatively, you can browse for a report definition file by selecting the associated browse button.

You may permit runtime editing of the report selection by checking the associated Edit check box.

Sequence path

Here you can specify the path within the data source to create the sequence. The sequence path may not contain the following characters:

/ " ' < | > ? *

Alternatively, you can browse for a sequence path by selecting the associated browse button.

You may permit runtime editing of the sequence path selection by checking the associated Edit check box.

Allow edit default program

Selecting this checkbox allows the user at run time to perform a selection in order to define the value for "Program" that will be used for the Sequence. This value may be overridden if the LimsLink EI Method is configured to acquire these values either by means of the Query or with the optional Winwrap script.

Allow edit default Method

Selecting this checkbox allows the user at run time to perform a selection in order to define the value for "Method" that will be used for the Sequence. This value may be overridden if the LimsLink EI Method is configured to acquire these values either by means of the Query or with the optional Winwrap script.

Use logon parameters to create sequence

If the 'Use logon parameters to create sequence' option is enabled, the LimsLink EI Method, at runtime, will default to use the parameter combination currently selected in the Chromeleon application. The following setup parameters will be affected when this option is selected: Data Source, Sequence Path and Sequence Name.

A selection at run-time will override the setup parameters as follows:

- The created sequence will automatically use the currently selected data source (or the data source containing the selected sequence path or sequence) in the Chromeleon browser.
- The created sequence will automatically use the currently selected sequence path (or the path containing the selected sequence) in the Chromeleon browser.
- If no path or sequence is currently selected (only the data source is selected), the path specified in the Select CDS Parameters window (see Figure 6) will be used.
- The created sequence will automatically use the name of the currently selected sequence in the Chromeleon browser. The created sample list will be appended to the existing samples in the sequence.
- If no sequence is selected, then the sequence name specified in the Sample List window (see Figure 7) will be used.

Defining the Sequence Fields

Optional Sequence Field Names

This lists a series of optional Sequence field names, acquired from Chromeleon. From here, you can select additional field names to be uploaded to the Sequence. All field names available to the Sequence for the selected data source are displayed. Custom field names entered by the user are also available for selection. Activating the check box for any additional fields will result in the respective field being uploaded to Chromeleon at runtime.

Establishing Variable Method and Program Values in a Sequence

The LLEI Chromeleon module supports the ability to interactively populate the Method and Program fields

in a Chromeleon Sequence.

To populate a Method or Program, you need to know the actual full path as well as the Method or Program name. Typically the path is hardcoded in the SQL query and appended to the Method or Program retrieved from the LIMS

An example of a SQL query that would be used if the LimsLink EI method used the *SQL Query LIMS* module would be:

Select 'server\analyst\january |\'| Program |\'| pgm from table which contains the desired Program in the Program field

The return is "server\analyst\january\ Sugar. pgm " (without the quotes). Where 'server\analyst\january' is the path and 'sugar' is the information retrieved from LIMS which is the program that will be used to run the samples. '.pgm' needs to be appended to the retrieval so that Chromeleon knows this is a Program to be used.

The same logic applies when desiring to populate the Method field in a Chromeleon Sequence except that the Method is found in the Method field and '.qnt' needs to be appended to the retrieval so that Chromeleon knows this is a Method to be used.

Running a Method

All runtime activities are accessible via the Chromeleon application. For example, sample retrieval from the LIMS, modifications to the retrieved sample list, and subsequent uploading of the Sequence to Chromeleon is accomplished through the Chromeleon application.

Note: The runtime dialogs reflect the combination of CDS and LIMS being used. You will want to reference your specific LIMS module documentation contained within the LimsLink EI manual to view the runtime dialogs specific to your LIMS. Refer to the section entitled 'Runtime Behavior'.

Accessing the Methods

To access the available Methods, perform the following steps:

1. From the Chromeleon application, click on the LimsLink EI - Create Sequence button to open the Select Method to Run window. This window will only open if more than one Method exists for the current system. If only one Method exists, the Method will start automatically upon selecting the **Create Sequence** button.

The Method list is retrieved from the Method Database defined via the Preferences window (see Figure 2). If the Method validation option is activated via the Preferences window, only validated Methods will be displayed and available for selection. If the option to 'Allow user to run unvalidated methods' is activated via the Preferences window then all Methods currently configured for the system will be displayed and available for selection.

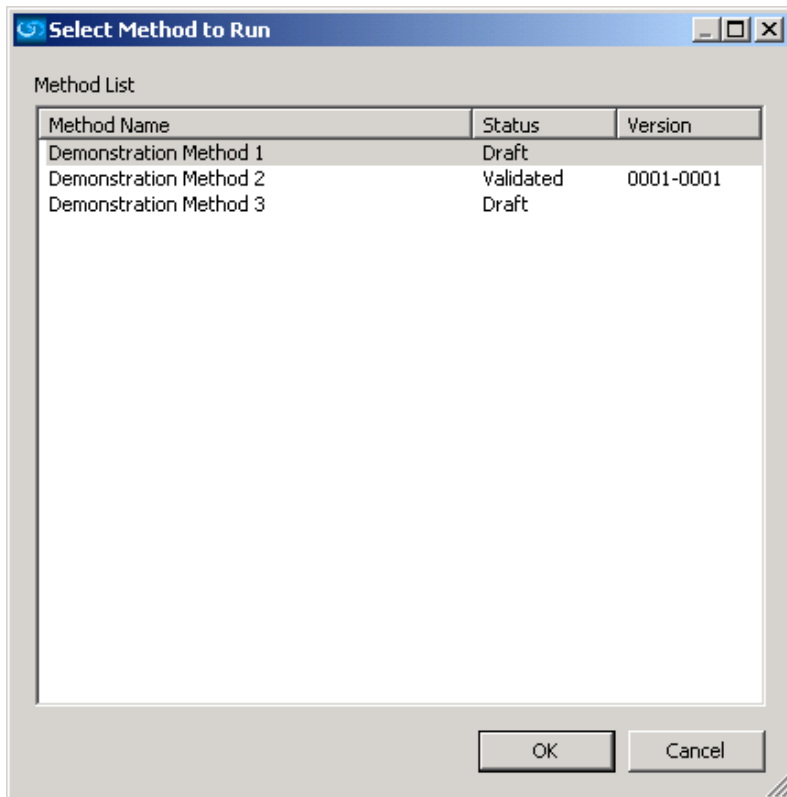


Figure 5: Select Method to Run

2. Highlight the Method to run and click on the **'OK'** button.

Selecting CDS Parameters

The next dialog will be presented only if one or more of its parameters (i.e., Data source, Time base, Program, Method, Report, Sequence Path) can be edited. If through the CDS Parameters Setup, all parameters were defined and non-editable at runtime, you will not see this window. Once you have made your selections, click on the **'OK'** button.

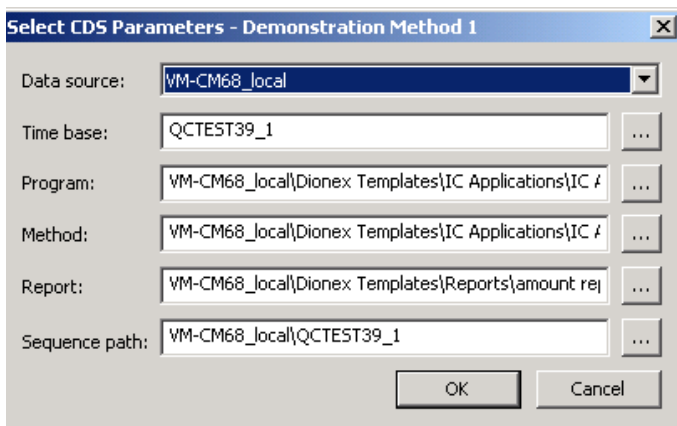


Figure 6: Select CDS Parameters

Selecting LIMS Parameters

The contents of the next dialog(s) will vary depending on the LIMS being used, and the Method Setup. You will want to reference your specific LIMS module documentation to see the actual runtime dialog(s). This information is contained within the LimsLink EI User Manual. Refer to the section entitled 'Runtime Behavior'.

Viewing the Sample List

Once the list of samples to be analyzed has been retrieved from the LIMS, the software will subsequently display the Sample List containing those samples.

Note: The contents of the next dialog will vary depending on the Method Setup. Not all options may be available.

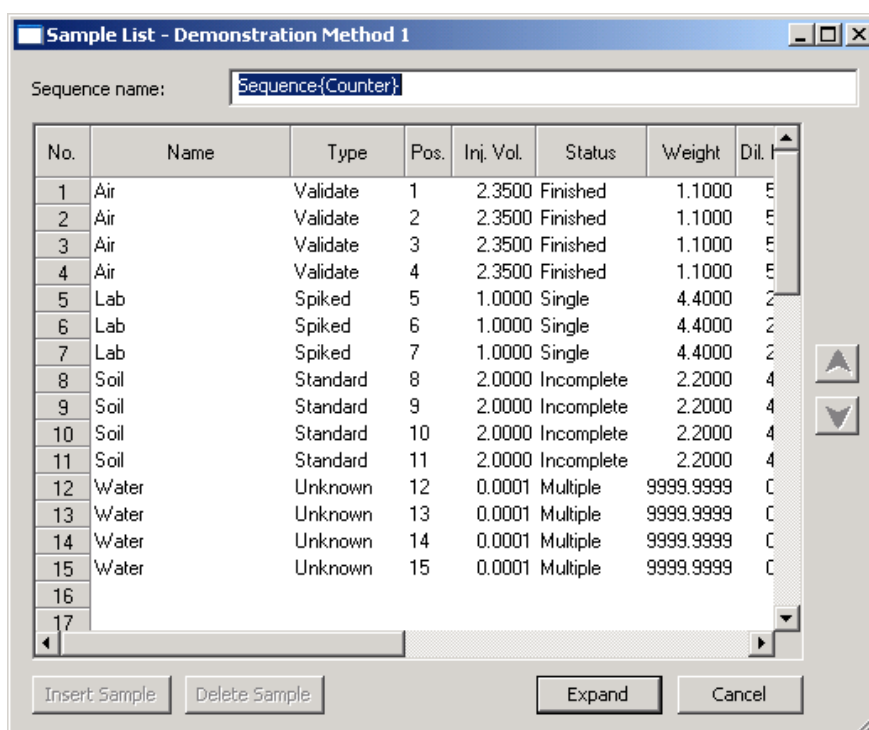


Figure 7: Sample List

Editing the Sample List

At this point you can review the sample list and if, during Method Setup, runtime editing of various capabilities (e.g., sample deletion, sample insertion, sample editing, sample sorting) was enabled, you can revise the sample list. The Sample List will only be displayed if one or more of these options were selected in the Method Setup.

Inserting a Sample

Clicking on the '**Insert Sample**' button will insert a blank sample row directly above the currently selected row. This button will only be available if the option to permit runtime editing for sample insertion was

enabled during the Method Setup. It will take on the default fields established during setup, and the user will need to directly enter the sample information into the grid.

Deleting a Sample

Clicking on the '**Delete Sample**' button will delete the currently selected sample. If multiple samples are selected, all selected samples will be deleted. This button will only be available if the option to permit runtime editing for sample deletion was enabled during the Method Setup.

Sample Editing

The software permits the editing of individual sample fields. The ability to edit existing samples depends on the options selected in the Method Setup and the LIMS/CDS Field Names Cross Reference Setup window. Samples that are added manually at this point can be edited regardless of the editing permissions set.

Sorting the Sample List

You can rearrange the sample list by clicking on the column headers to alternate between ascending and descending sorts based on the selected column. You also can move samples in the list by clicking and dragging a sample from one position in the list to another. For example, you can click and drag the 5th sample up to the 1st sample position. Alternatively, you can use the up and down arrows to move the selected sample up and down in the list, respectively.

Expanding the Sample List

Once the Sample List is correct, clicking on the '**Expand**' button automatically incorporates special samples such as controls and standards, set up via the Expansion Setup window, into the Sequence.

If expansion is not enabled, the button will be labeled '**Create Sequence**'. Clicking this button will upload the information to Chromeleon.

Sequence name:

No.	Name	Type	Pos.	Inj. Vol.	Status	Weight	Dil. Factor
1	Air	Validate	1	2.3500	Finished	1.1000	5.5000
2	Air	Validate	2	2.3500	Finished	1.1000	5.5000
3	Air	Validate	3	2.3500	Finished	1.1000	5.5000
4	Air	Validate	4	2.3500	Finished	1.1000	5.5000
5	Lab	Spiked	5	1.0000	Single	4.4000	2.2000
6	Lab	Spiked	6	1.0000	Single	4.4000	2.2000
7	Lab	Spiked	7	1.0000	Single	4.4000	2.2000
8	Soil	Standard	8	2.0000	Incomplete	2.2000	4.4000
9	Soil	Standard	9	2.0000	Incomplete	2.2000	4.4000
10	Soil	Standard	10	2.0000	Incomplete	2.2000	4.4000
11	Soil	Standard	11	2.0000	Incomplete	2.2000	4.4000
12	Water	Unknown	12	0.0001	Multiple	9999.9999	0.0001
13	Water	Unknown	13	0.0001	Multiple	9999.9999	0.0001
14	Water	Unknown	14	0.0001	Multiple	9999.9999	0.0001
15	Water	Unknown	15	0.0001	Multiple	9999.9999	0.0001
16							
17							

Buttons:

Figure 8: Expanded Sample List

Again, you have the option to insert and delete samples from the Expanded Sample List as well as edit individual samples, provided these runtime editing options were enabled during the Method Setup. There is no option to sort the Expanded Sample List.

Once the Expanded Sample List is correct, clicking on the **'Create Sequence'** button automatically uploads the information to Chromeleon.

Sequence name:

Created by:

Figure 9: Sequence Successfully Created

Reporting Results to the LIMS

Using your LimsLink Method (note this is not the same Method used for the LimsLink EI application) and Labtronics ReportLink, the results acquired by Chromeleon can be reported to your LIMS.

Refer to the LimsLink EI User Manual as well as the ReportLink User Manual for further details.

Template Script

Chromeleon Script

Option Explicit

'-----

' Main runtime function.

'-----

Function Execute(ByVal LIMSMOD As CDSLINKLib.ILIMSModule, ByVal CDSMOD As
CDSLINKLib.ICDSModule, ByVal FieldMap As CDSLINKLib.FieldNameMap, ByVal SampleCont As
CDSLINKLib.Samples, ByVal Expand As Boolean) As Boolean

Execute = False

On Error GoTo ErrHandler

Dim objLIMS As Object

Set objLIMS = LIMSMOD

Dim objCDS As Object

Set objCDS = CDSMOD


```

Dim Ret As Long

Ret = vbOK

' Display the runtime display in the CDS module.
Dim CDSRuntime As CDSLINKLib.ICDSRuntimeDisplay
Set CDSRuntime = objCDS
If Not CDSRuntime Is Nothing Then
    Ret = CDSRuntime.DoRuntimeDisplay
    If Ret = vbCancel Then
        CDSMod.Logoff
        Exit Function
    End If
End If

' Initialize the fields in the sample container
' Any changes to the database or project global fields beyond this point will be ignored
CDSMod.InitializeFields SampleCont

' Get the samples from the LIMS module.
Ret = LIMSMOD.GetSamples( SampleCont )
If Ret = vbCancel Then
    CDSMod.Logoff
    LIMSMOD.Logoff
    Exit Function
End If

SampleCont.AutoNumberField("Pos.", FieldKeyType_Name, 1, 1)

' Display the sample list in the CDS module.
Dim CDSSampList As CDSLINKLib.ICDSSampleDisplay

```

```

Set CDSSampList = objCDS
If Not CDSSampList Is Nothing Then
    Ret = CDSSampList.DoSampleDisplay( SampleCont )
    If Ret = vbCancel Then
        CDSMod.Logoff
        Exit Function
    End If
End If

' If we are using sample expansion...
If Expand Then
    ' Expand the samples if the CDS module supports expansion.
    Dim CDSExp As CDSLINKLib.ICDSExpansion
    Set CDSExp = objCDS
    If Not CDSExp Is Nothing Then
        CDSExp.ExpandSamples SampleCont
        SampleCont.AutoNumberField("Pos.", FieldKeyType_Name, 1, 1)
    End If

    ' Display the sequence list in the CDS module.
    Dim CDSSeqList As CDSLINKLib.ICDSSequenceDisplay
    Set CDSSeqList = objCDS
    If Not CDSSeqList Is Nothing Then
        Ret = CDSSeqList.DoSequenceDisplay( SampleCont )
        If Ret = vbCancel Then
            CDSMod.Logoff
            Exit Function
        End If
    End If
End If

```

```

' Generate the sequence and upload it to the CDS.
Dim Status As Boolean
Status = CDSMod.UploadSequence( SampleCont )

' Notify the LIMS module of the upload sequence status.
LIMSMod.SequenceCompleted Status

LIMSMod.Logoff
CDSMod.Logoff

Execute = True
Exit Function

ErrorHandler:
MsgBox Err.Description & vbCrLf & "The script will be terminated"

LIMSMod.Logoff
CDSMod.Logoff
End Function

'-----
' Sample change event handler.
'-----

Sub CDSModuleEvents_OnSampleChange(ByVal Source As CDSLINKLib.SampleChange, ByVal
ChangeType As CDSLINKLib.SampleChangeType, ByVal SampleIndex As Long, ByVal SampleCont As
CDSLINKLib.Samples, ByRef Changed As Boolean)

' Add code here

SampleCont.AutoNumberField("Pos.", FieldKeyType_Name, 1, 1)

```

```

        Changed = True
End Sub

'-----
' Sample edit event handler.
'-----

Sub CDSModuleEvents_OnSampleEdit(ByVal Source As CDSLINKLib.SampleChange, ByVal
SampleIndex As Long, ByVal FieldIndex As Long, ByVal SampleCont As CDSLINKLib.Samples, ByRef
Changed As Boolean)

    ' Add code here

End Sub

'-----
' Sample sort event handler.
'-----

Sub CDSModuleEvents_OnSampleSort(ByVal FieldIndex As Long, ByVal SampleCont As
CDSLINKLib.Samples, ByRef Changed As Boolean)

    ' Add code here

    SampleCont.AutoNumberField("Pos.", FieldKeyType_Name, 1, 1)

    Changed = True

End Sub

```

Empower

Introduction

Overview

LimsLink EI (Embedded Interface), a component of the Labtronics LimsLink CDS product, provides an embedded link between your CDS and LIMS. Designed to be a Plug & Play system, you can substitute in various CDS and LIMS modules to work with the interface, and set up the individual modules to suit your requirements.

The Empower module is suitable for use with Empower 2 FR5, and is compatible with any LIMS module, selected during installation.

The Empower module provides the functionality required to view sample lists retrieved from the LIMS, edit and expand the sample list to include control samples, and then convert this list to a Sample Set Method for analysis by Empower. Once analysis is complete, the results can be reported back to the LIMS.

The system also allows configuration supporting the exporting of results to a file that can then be collected by a LimsLink Method in order to report the results to a LIMS. As part of the LimsLink EI installation procedure, you are required to select the CDS and LIMS module for the installation. Upon selecting the Empower CDS, the Empower module is then installed as part of your LimsLink EI installation. The LimsLink EI installation provides the user the ability to add the **LimsLink EI - Create Sample Set Method, LimsLink EI - Method Setup and the LimsLink EI –Results to LIMS** sub-menu items to the Empower Client. With the LimsLink EI application, LIMS integration is embedded directly into your Empower application providing users with a direct and transparent connection to their LIMS.

Installation

Installation Prerequisites

The LimsLink EI application requires that the following software be installed, and operational, prior to installation:

- Empower 2 FR5
- A valid Empower user name and password

If Empower is a Client/Server edition, the LimsLink EI application must be installed separately on each of the Clients.

During the LimsLink EI installation, the user will have the option to determine where in Empower the LimsLink EI menu items will be placed. For each menu item, there are three options: None, Project Dialogs, and Run Samples. If 'None' is selected, the corresponding menu item will not appear in Empower. This option should only be used if an Empower Client is intended exclusively for creating or running Methods, but not both. The 'Project Dialogs' option will place the menu item in Empower's Project windows. The 'Run Samples' option will place the menu items in the Run Samples window.

Upgrading Methods from LimsLink EI v2.0

In order to leverage the enhanced autonumbering functionality in LimsLink EI v4.2, some minor updates are required to the Method Script of existing Methods. A simple way to update the script is simply to save the script from a new Method created in LimsLink EI v4.2 and open it into the existing Method. Alternatively, to manually perform the updates, please do the following.

A new variable must be declared at the beginning of the script, after the `Option Explicit` statement.

```
Dim g_CDSEMod2 As CDSLINKLib.ICDSModule2
```

This new variable must be assigned to the `objCDS` object after the statement `Set objCDS = CDSEMod:`

```
Set g_CDSEMod2 = objCDS
```

Also, there are three statements in the code that appear as follows:

```
SampleCont.AutoNumberField "Vial", FieldKeyType_Name, 1, 1
```

These statements should be updated to the following in order to use the enhanced functionality:

```
If Not g_CDSEMod2 Is Nothing Then
```

```
    g_CDSEMod2.AutoNumberField(SampleCont, "Vial", FieldKeyType_Name, 1, 1)
```

```
Else
```

```
    SampleCont.AutoNumberField("Vial", FieldKeyType_Name, 1, 1)
```

```
End If
```

Please refer to the Template Script for more information.

Upgrading Methods from LimsLink EI v4.1.x

LimsLink EI version 4.1.x methods can be imported directly into version v4.2.

LimsLink EI – Empower Menu Utility

As noted, the location of the LimsLink EI menu items for Empower is determined during the installation of LimsLink EI. To alter this location after installation has occurred, a utility (EmpowerMenuUtility.exe) is provided in the Utilities folder on the installation CD.

Running the EmpowerMenuUtility.exe

Running the EmpowerMenuUtility.exe will open a LimsLink EI – Empower Menu Utility window.

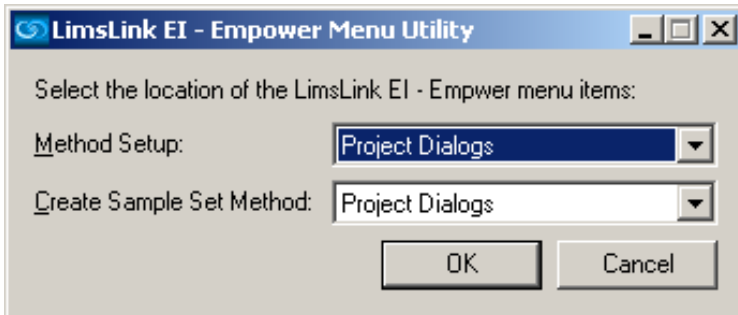


Figure 1: LimsLink EI – Empower Menu Utility

From here, select the location of the LimsLink EI – Empower menu items. You can specify a location for the Method Setup and Create Sample Set Method menu items.

Security and Audit Trail

The **LimsLink EI - Create Sample Set Method** and **LimsLink EI - Method Setup** menu items are controlled by the Empower security system, and are therefore restricted to the appropriate users. All Empower users are able to access the Create Sample Set Method menu item, provided they can access the Empower dialog where it is located. For the Method Setup menu item, only those users whose user type is Administrator, or those users who are members of the group “LimsLinkAdmins”, are able to access Method Setup.

Status of the Sample Set Method upload is automatically entered into the Empower audit trail providing the necessary documentation to meet audit or regulatory requirements.

For each LimsLink EI audit trail entry, the value placed in the Action field of the Empower audit trail is 'Tool Kit'. The other field to note is the 'Details' field. All remaining fields behave exactly as they would in Empower.

The following are examples of Audit Trail Details field entry:

Example 1 – Setup auditing

LimsLink EI v4.2 for Empower, Method: <method name>, Dialog: <dialog name>, <what happened>

This may resolve to:

LimsLink EI v4.2 for Empower, Method: Demonstration Method, Dialog:

LimsLink EI Setup, Method created.

Example 2 – Runtime auditing

LimsLink EI v4.2 for Empower, Method: <method name>, Dialog: <dialog name>, Sample ID: <sample ID>, Field: <field name>, <what happened>

The above would typically evaluate to:

LimsLink EI v4.2 for Empower, Method: Test Method, Dialog: Sample List,

Sample ID: A0004, Field: Label changed from LabelValue1 to LabelValue2

The sample ID value corresponds to the field selected in the Audit Trail

Setup (e.g., The SampleName value for a particular sample).

Example 3 – Runtime auditing

LimsLink EI v4.2 for Empower, Method: <method name>, <what happened>.

In this example, <method name> refers to the name of the Method running,

and <what happened> may be something similar to 'Method run started' or 'Method run incomplete' or 'Method run complete'.

Messaging Center

Throughout the operation of the LimsLink EI application, important runtime messages may be sent to the user via the Empower Message Center. This allows Empower users to receive information in a convenient and familiar way.

For example, suppose you are running a Method that does not allow the user to edit the contents of the Sample/Expanded Sample lists. If the default Method Set for that Method no longer exists within the project, a message center entry will be made indicating the "The method set <method set name> is not available. The set method creation will be aborted."

References

The documentation for the Empower system only refers to that information which pertains to the direct setup and runtime use of LimsLink EI with respect to the Empower application. You will need to refer to the LimsLink EI User Manual for specifics relating to the setup of the application as a whole.

System Setup

A set of parameters governing the transfer of information between the LIMS and the CDS must first be established before information retrieval and analysis can occur. This information would typically be

configured by someone with administrative privileges, and would only need to be configured once during the initial setup.

Accessing the System Setup

To access the system parameters, perform the following steps:

1. From the Empower window where the Method Setup menu item was created, pull down the **Application** or **LimsLink** menu.

Note: The name of this menu may vary depending on the configuration of your Empower Client.

2. Select the **LimsLink EI - Method Setup** menu item to open the LimsLink Embedded Interface Setup window, similar to the example shown here:

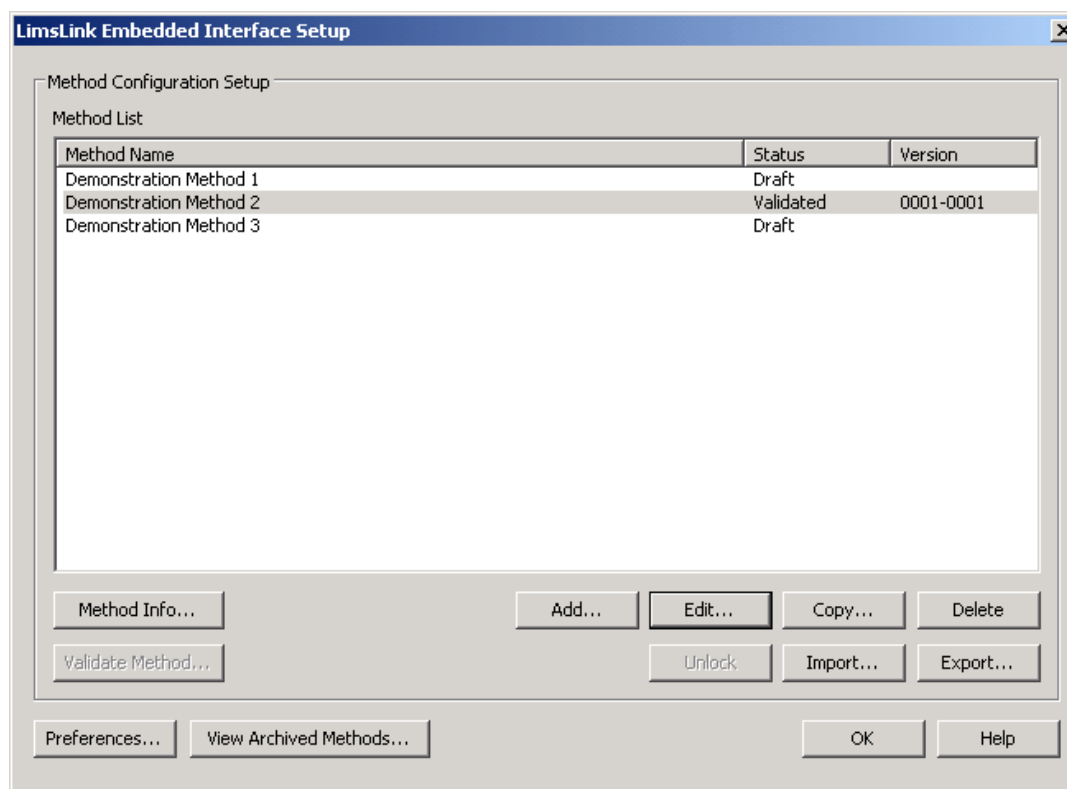


Figure 2: LimsLink Embedded Interface Setup

3. Click on the **'Preferences'** button on the LimsLink Embedded Interface Setup window to open the Preference window, similar to the example shown here.

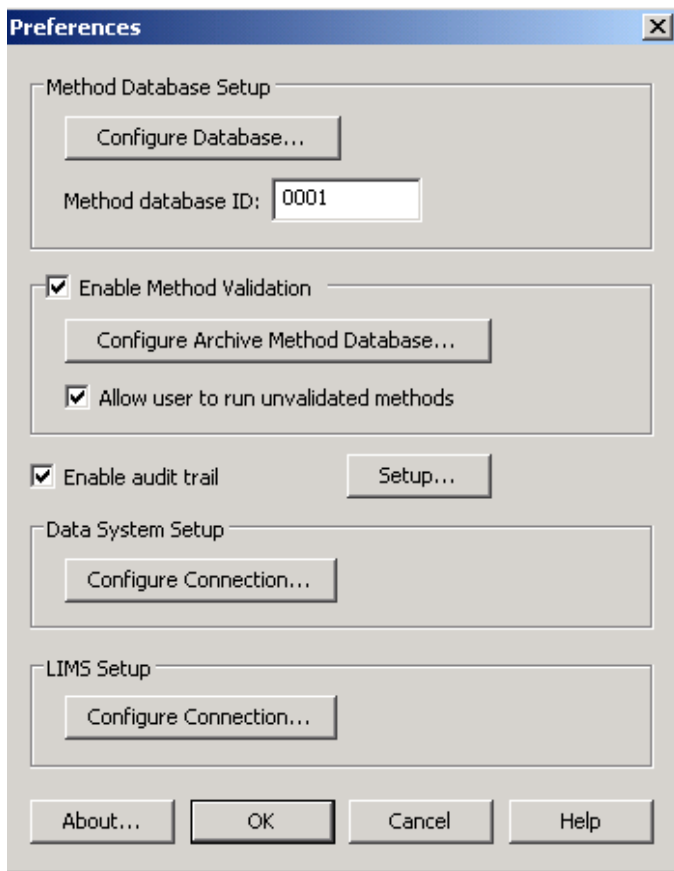


Figure 3: Preferences

System Preferences

Via the Preference window, you are able to determine the location of data repositories such as the Method Database and the Archive Method Database. Other specifics such as Method Validation, and Audit Trail can be established here. Refer to Chapter 2 - System Setup in the LimsLink EI User Manual for additional details.

Making Entries to the Audit Trail

The Audit Trail can be set up such that the software will audit changes made at both setup and runtime. To do this, you must first enable the audit trail in the Preferences window by activating the check box, and then clicking on the accompanying '**Setup**' button. Next you must specify the field name within Empower that will be used to identify any changes made to individual samples within the Sample Lists.

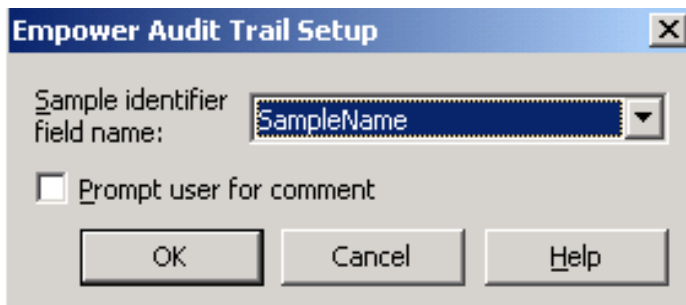


Figure 4: Empower Audit Trail Setup

The Empower Client will supply the available field names. If you change the default selection, the message – “This field must exist in every method or the audit trail will not work correctly. Do you want to change the field?” will be displayed.

Activating the ‘Prompt user for comment’ check box will then prompt the user for a comment every time a change is made to the sample or expanded sample list as well as whenever a database or project is selected at runtime.

Empower Configuration Setup

To verify the Empower version number, service path information plus applied patches detected by the LimsLink EI application, click on the **‘Configure Connection’** button within the Data System Setup group box to open the Empower Configuration Setup window.

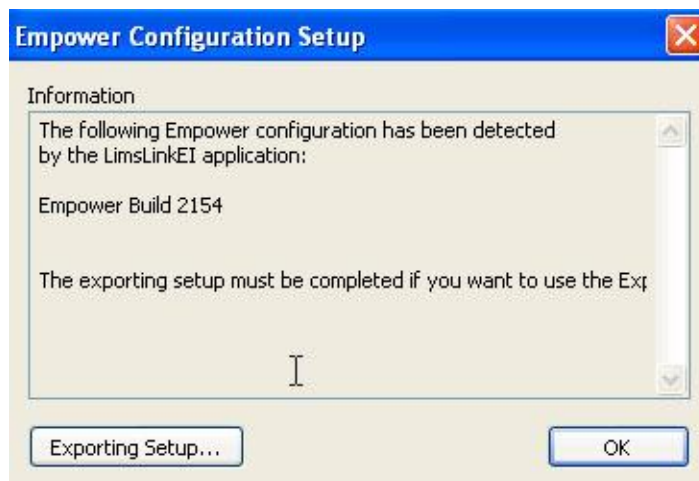


Figure 5: Empower Configuration Setup

Exporting Setup

The Empower module has the capability to export results to a file that can then be collected by a LimsLink Method in order to export the results to a LIMS. There are four export templates that can be used for exporting the results. They have a pre-defined format and structure.

Four result export formats are supported. They are as follows:

Individual Export – used to report individual sample results from Empower2.

Mean Export – used to report results including calculations (Mean, Standard Deviation, and Relative Standard Deviation) for samples and peaks from Empower2.

MOLWT Export – used to report results for MOLWT from Empower2.

Generic Export – used as a default generic output format that can be used to acquire results from Empower2.

The Exporting Setup window is accessed via the '**Export Setup**' button on the Empower Configuration Setup window (see Figure 5). Via the Exporting Setup you can define the output path for the various export types, as well as the delimiter for the exports.

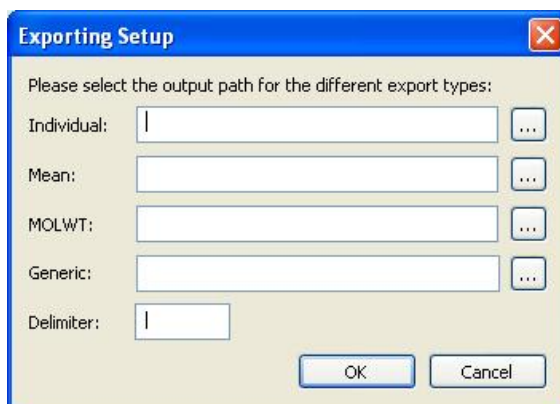


Figure 6: Exporting Setup

Individual – here you can define the path to which Individual exports will be saved. A path can be manually entered, or you can browse to the path.

Mean – here you can define the path to which Mean exports will be saved. A path can be manually entered, or you can browse to the path.

MOLWT – here you can define the path to which MOLWT (molecular weight) exports will be saved. A path can be manually entered, or you can browse to the path.

Generic – here you can define the path to which Generic exports will be saved. A path can be manually entered, or you can browse to the path.

The Delimiter field allows you to define the delimiter character(s) that will be used for ALL exports. The default delimiter is the pipe character (|). It is recommended that you use this character as the delimiter. A delimiter up to 5 characters in length may be entered. It will be used by the export to delimit the fields in the export files. An entry of a least one character in the Delimiter field is mandatory.

Clicking on the '**OK**' button will validate that the fields are either empty or that they contain a valid file path (either UNC or direct drive path). An error will be displayed if an invalid path is encountered.

Empower Custom Fields

The Export to LIMS features interacts with five optional Custom Fields on samples. The Custom Fields are defined within the Empower Project. The fields are optional in that they are not required to be present in the Empower Project if the corresponding export is not required to be generated.

If the Custom field is present in a project, then ALL samples are required to have the same values in the Custom Field.

The Custom field can remain blank meaning that the Export to LIMS will not be run. This is effectively the same as the Custom Field not being present.

The five Empower Result type custom fields are as follows:

- a.) **LL_IND_RPT** – The values in this custom field are a delimited list (using the same delimiter defined in the Exporting Setup dialog) of the variable fields that can be used in the Individual export. The Sample Name and Task ID fields must be the first two fields in the list followed by any two standard or custom Peak fields. If more than two fields are listed, the first two fields are used and any other fields are ignored
- b.) **LL_MEAN_RPT** – The values in this custom field are a delimited list (using the same delimiter defined in the Exporting Setup dialog) of the variable fields that can be used in the Mean export. The Sample Name and Task ID fields must be the first two fields in the list followed by any two standard or custom Peak fields. If more than two fields are listed, the first two fields are used, and any other fields are ignored.
- c.) **LL_MOLWT_RPT** – The values in this custom field are a delimited list (using the same delimiter defined in the Exporting Setup dialog) of the variable fields that can be used in the MOLWT export. The Results fields are: Sample Name, Task ID, Mn, Mw, MP, Retention Time, Total Area. Additionally the MW Markers 3000, 8000 and 10000 values are exported from the Cumulative % field.
- d.) **LL_GEN_RPT** – This Custom Field will not contain any contents as there are no variable components for it. The presence of this Custom Field will result in an export of a file containing all fields in a delimited list
- e.) **LL_NUM_SIGNOFFS**-This field determines the expected number of Sign Offs for an Empower Result. This could be 0, 1 or 2. This is compared against the existing Empower field 'Number of Sign Offs' which designates the number of Sign Offs actually accomplished for any Result. If this Custom field is not present, the program will still continue to execute.
- f.) **LL_REPORTED** - This field determines if the results have already been reported. By default, this field is empty. After reporting once, the field is populated by LimsLink EI with a 'Y'. The software will check this field when the '**LimsLink EI - Results to LIMS**' option is selected. This option is described later in the manual. If this field contains anything other than NULL, the system will display the following message box, 'Results selected have already been reported. Proceed anyway?'

Defining the CDS Parameters

Method Setup

Recall that a Method is responsible for retrieving information from the LIMS, modifying the retrieved information, and subsequently uploading the information to the CDS. As part of the Method setup, the parameters for the respective CDS being used need to be defined in order for the information to be uploaded to the CDS.

Accessing the Empower Setup

1. To access the Method Setup window, click on the **'Add'** button from the LimsLink Embedded Interface Setup window.
2. Click on the **'Data System Parameters Setup'** button to open the Empower Setup window.

Empower Setup

Method name:

CDS Parameters

Database: ☒ Edit

Project: ☒ Edit

☐ Use logon parameters to create sequence

Sample Set Method Based Parameters

☒ Sample set name: ☒ Edit

☒ Method set: ☒ Edit

☒ Report method: ☒ Edit

☐ Use existing sample set: ☐ Edit

Sample addition: ☐ Edit

☐ Update selected sample set method ☐ Edit

Optional Sample Set Field Names	
<input checked="" type="checkbox"/>	Function
<input checked="" type="checkbox"/>	Processing
<input type="checkbox"/>	LabelReference
<input checked="" type="checkbox"/>	Level
<input type="checkbox"/>	Label
<input type="checkbox"/>	Comments

Figure 7: Empower Setup

Setting up Empower

The Empower Setup allows you to define the information that is required to create the Sample Set Method (i.e., the list of samples) to be analyzed by the Empower CDS. The CDS Parameters section determines the database and project to be used; the Sample Set Method Based Parameters determine the contents of the Sample Set Method itself. Here, you can specify what parameters are required by the Sample Set Method.

Method Name – The name of the current Method is displayed. It is not editable.

Defining the CDS Parameters

Database

From here, you can select the CDS database to which to connect to. All databases, supplied by the Empower Client, applicable to the current user, are available for selection.

A database is a user-defined collection of Projects. Since Projects are database dependent, changing the database setting may re-populate the data shown in the dialog.

You may permit runtime editing of the database selection by checking the associated Edit check box.

If you select a new database, a message will appear stating that 'Field settings in other setup dialogs are determined by the selected database, and that you should ensure that those settings are updated to reflect the new database selected'. All settings that depend on the database will revert to their default values. LimsLink EI will attempt to retain any previously selected values that apply to the newly selected database.

If the Empower Client is a standalone workstation, the list box will display 'Local Database'. It will be non-editable.

Project

From here, you can select the project that will be used to store the Sample Set Method.

A project is a user-defined collection of methods, results, peaks, custom fields, and raw data that reside in the database under a single project name. Projects are database dependent, so changing the database setting may re-populate the data shown in the dialog.

You may permit runtime editing of the project selection by checking the associated Edit check box.

If you select a new project, a message will appear asking you if you want to retain the settings from the previously selected project. If 'Yes' is selected, the software will attempt to match the previously selected options with the newly selected project. If 'No' is selected, the software will reset the options to their default values.

Use logon parameters to create sequence

The current Client database and project settings can be used as the default option(s). If the 'Use logon parameters to create sequence' option is enabled, the LimsLink EI Method, at runtime, will default to use the Database/Project combination currently open in the Empower Client. Default Database/Project selections will be ignored. When using this option, it is important to ensure that compatible databases/projects are being used with the Method. If the current project is missing certain fields required by the Method, for instance, those fields will be dropped by the Method at runtime. In this situation, an entry is made to the audit trail.

Defining the Sample Set Based Parameters

The Sample Set Method Based Parameters allow you to select the options required to properly construct the Sample Set Method to be analyzed by Empower. Some parameters (e.g., Sample set name and Method set) are a mandatory part of the Sample Set Method, and therefore the accompanying check box is non-editable.

Sample set name

Here you can enter a name for the Sample Set. You have the option to specify a date and time macro within the Sample Set name to allow a unique naming convention, or you can enter a fixed Sample Set name that will be applicable to all Sample Sets created using the selected Method.

If using the {DATE} macro, the software, at runtime, will substitute a six digit (YYMMDD) value that will represent the current date. If using the {TIME} macro, the software, at runtime, will substitute a six digit (HHMMSS) value that will represent the current time (24 hour format). For example, the Sample Set name with prefix SEQ and the date and time variable would be entered as SEQ{DATE}_{TIME}. The Sample Set Method name would be SEQ080314_210521 if the Sample Set Method were created on March 14, 2009 at 9:05:21 PM.

You may permit runtime editing of the Sample Set name by checking the associated Edit check box.

Method set

You can select the default Method Set that will be used for all samples to be contained within the Sample Set Method. The drop-down list box, supplied by the Empower Client, will contain all Methods available to the current user for the selected database and project.

Method sets are project dependent, so changing the project/database setting may re-populate the Method set drop-down list box.

You may permit runtime editing of the Method Set by checking the associated Edit check box.

Report method

Here you can select the default Report method to be used if the Function is set to 'Report'.

The drop-down list box, supplied by the Empower Client, will contain all Report methods available to the current user for the selected database and project.

Report methods are project dependent, so changing the project/database setting may re-populate the Report method drop-down list box.

You may permit runtime editing of the Report method by checking the associated Edit check box.

Use existing sample set

Here you can select an existing Sample Set to use when saving the new Sample Set. The drop-down list box will contain the current list of Samples Sets available to the current user for the selected database and project.

Sample Sets are project dependent, so changing the project/database setting may re-populate the Existing Sample Set drop-down list box.

You may permit runtime editing of the existing Sample Set by checking the associated Edit check box.

Sample Addition

The Sample addition parameter allows you to define how new samples will be handled by the specified existing sample set. The following options are available:

1. Append samples to the bottom of the specified Sample Set
2. Insert new samples at a specified row number using the associated spin box. A value of 1-99 may be

specified.

3. Replace all existing samples in the specified Sample Set.
4. Substitute unknowns. When this option is selected, a Label Cross-Reference window will open similar to the example shown here.

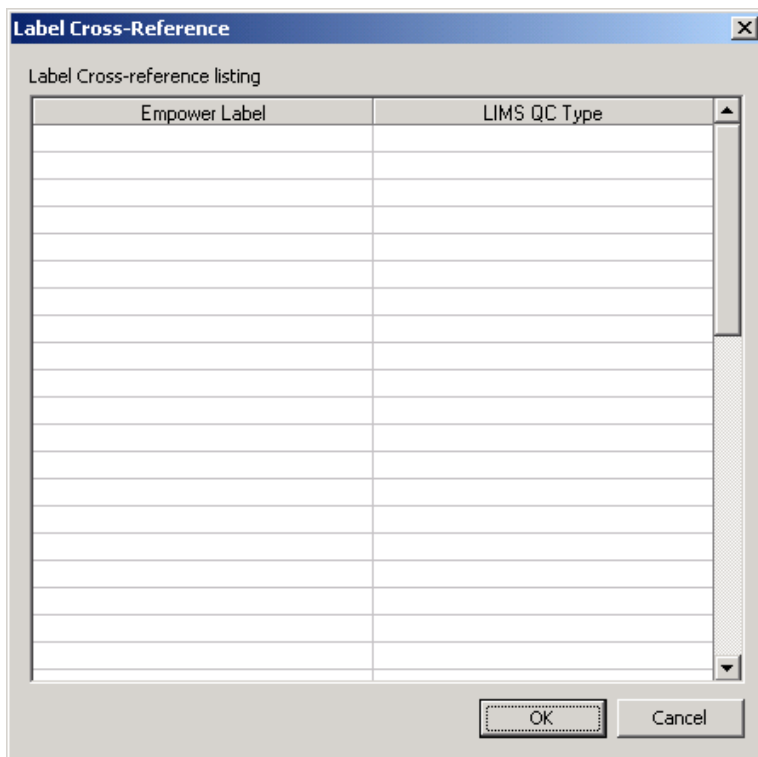


Figure 8: Label-Cross Reference

Via the Label Cross-reference listing grid, you can establish a link by which to transport sample data between field names originating from the LIMS to field names selected for the Sequence to be sent to Empower. Note that it is not always necessary to establish a link between all fields as some are established by default (e.g., function) and other may be established via a customized script.

Update selected sample set method

Here you can select to update the selected Sample Set Method. If this option is enabled, the existing Sample Set Method will be updated with the samples in the Sample List, based on the selected sample addition procedure. If this option is not enabled, then a new Sample Set Method will be created using the specified name.

Optional Sample Set Field Names

A series of optional Sample Set Method field names, acquired from the Empower Client, is presented. From here, you can select additional field names to be uploaded to the Sample Set Method. All field names available to the Sample Set Method for the selected database and project are displayed. Custom field names entered by the user are also available for selection. Activating the check box for any additional fields will result in the respective field being uploaded to Empower at runtime.

Note: The Field names 'Vial number', 'Sample Name', 'Method Set/Report Method' are mandatory fields and are automatically included in the Sample Set Method. These fields do not appear in the list of Optional Sample Set Field Names.

Configuring Autonumbering

You have the option to autonumber the rows (for specified Functions) of the Sample List at runtime. Click on the '**Configure Autonumbering**' button to open the Autonumbering Setup window.

By enabling the checkbox to the left of the Function, any samples associated with the selected Functions will be given a vial number at runtime.

The following Functions will be enabled by default:

Inject Standards

Inject Broad Standards

Inject Narrow Standards

Inject Samples

Inject Broad Samples

Inject Narrow Samples

Inject Controls

Inject RF Internal Standards

Inject Immediate Standards

Inject Immediate Samples

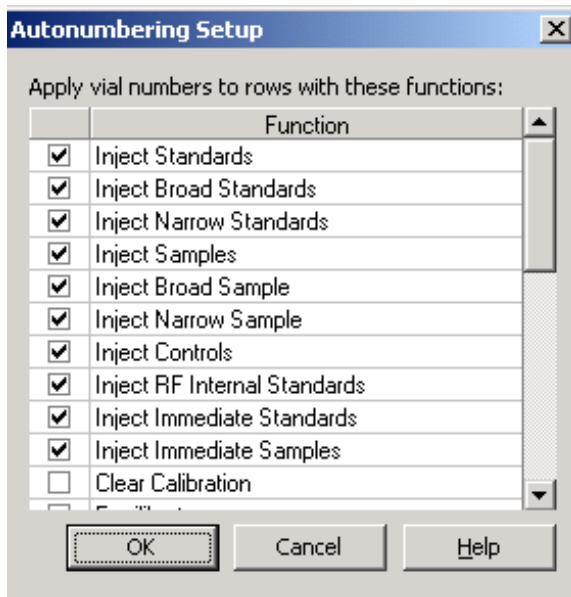


Figure 9: Autonumbering Setup

Running a Method

All runtime activities are accessible via the Empower Client. For example, sample retrieval from the LIMS, modifications to the retrieved Sample List, and subsequent uploading of the Sample Set Method to Empower for analysis, is accomplished through the Client. Exporting results to LIMS is also accessible via the Empower Client.

Note: The runtime dialogs reflect the combination of CDS and LIMS being used. You will want to reference your specific LIMS module documentation contained within the LimsLink EI manual to view the runtime dialogs specific to your LIMS. Refer to the section entitled 'Runtime Behavior'.

Accessing the Methods

To access the available Methods, perform the following steps:

1. From the Empower window where the Method Setup menu item was created, pull down the **Application or LimsLink** menu.
2. From the Empower where the Method Setup menu item was created, pull down the **Application or LimsLink** menu.

Note: The name of this menu may vary depending on the configuration of your Empower Client.

3. Select the **LimsLink EI - Create Sample Set Method** menu item to open the Select Method to Run window. This window will only open if more than one Method exists. If only one Method exists, the Method will start automatically upon selecting the **LimsLink EI - Create Sample Set Method** menu item.

The Method list is retrieved from the Method Database. If the Method validation option is activated via the Preferences window, only validated Methods will be displayed and available for selection. If the option to 'Allow user to run unvalidated methods' is activated via the Preferences window then all Methods currently configured for the system will be displayed and available for selection.

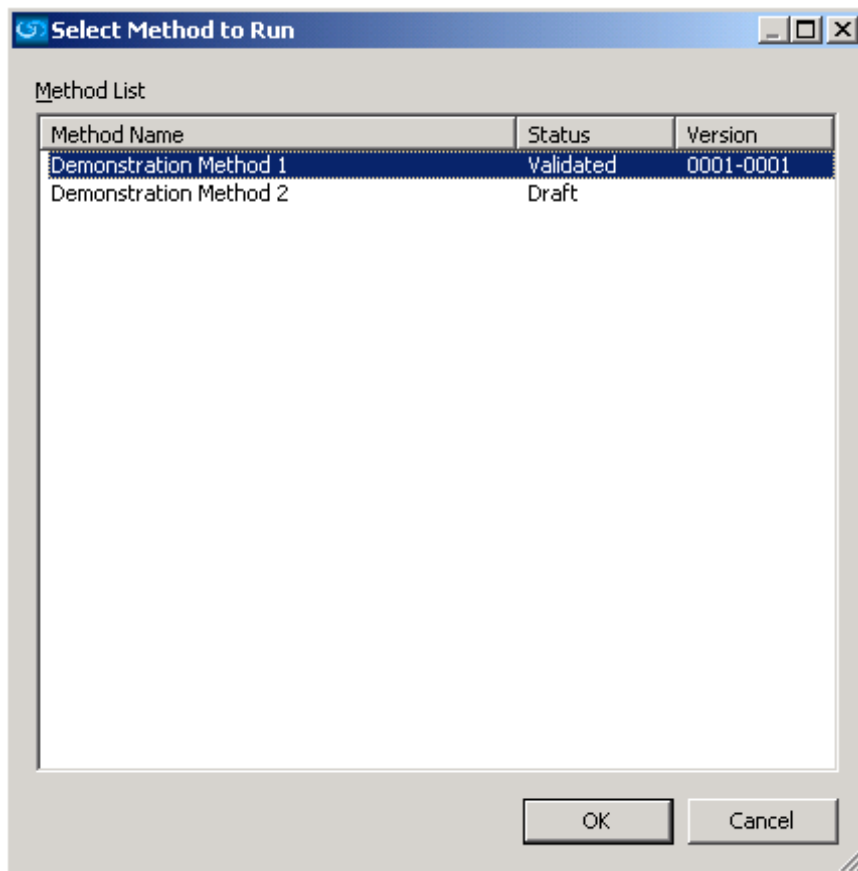


Figure 10: Select Method to Run

4. Highlight the Method to run and click on the '**OK**' button.

Selecting CDS Parameters

The next dialog will be presented only if one or more of its parameters can be edited (i.e., the Edit check box was checked in Method Setup). Once you have made your selections, click on the '**OK**' button.

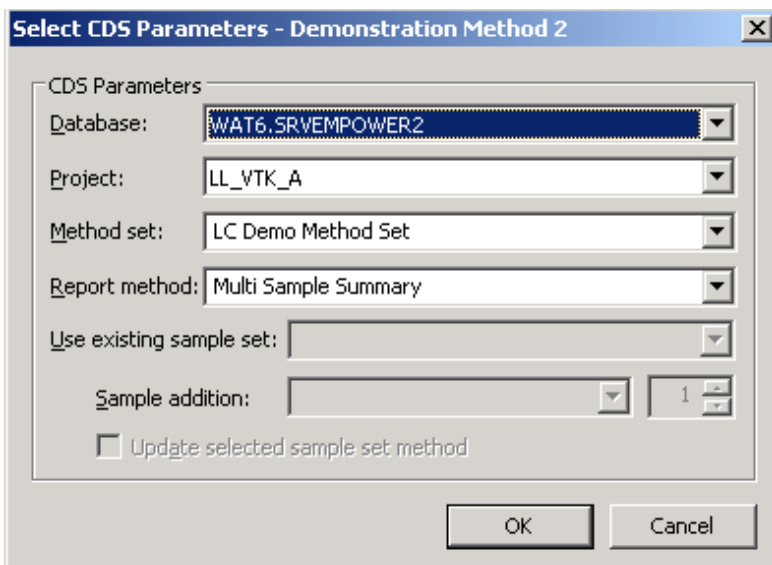


Figure 11: Select CDS Parameters

Selecting LIMS Parameters

The contents of the next dialog(s) will vary depending on the LIMS being used, and the Method Setup. You will want to reference your specific LIMS module documentation to see the actual runtime dialog(s). This information is contained within the LimsLink EI User Manual. Refer to the section entitled 'Runtime Behavior'.

Viewing the Sample List

Once the list of samples to be analyzed by Empower has been retrieved from the LIMS, the software will subsequently display the Sample List containing those samples.

Note: The contents of the next dialog will vary depending on the Method Setup. Not all options may be available.

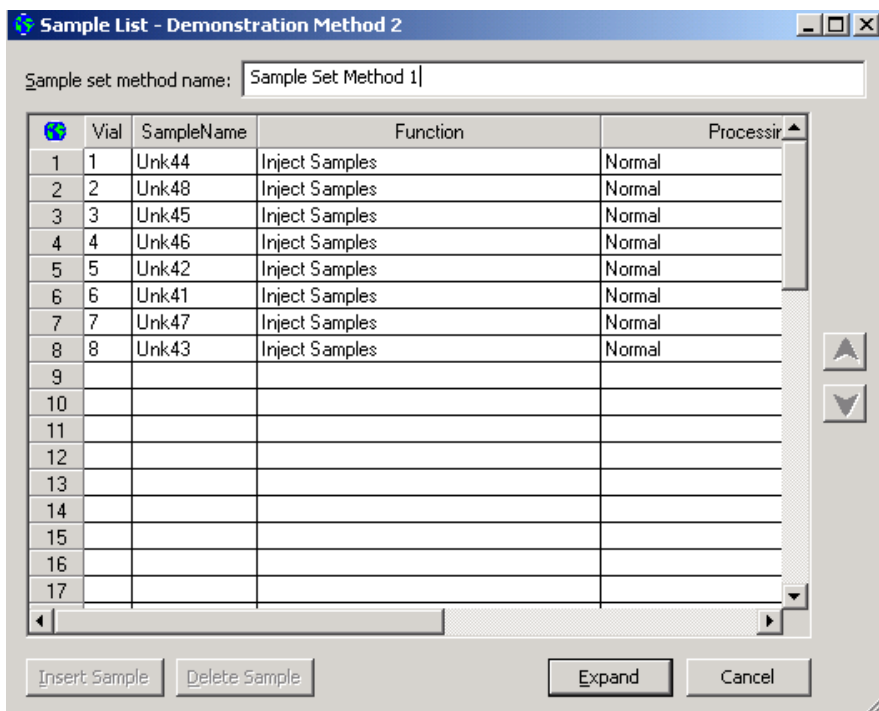


Figure 12: Sample List

Editing the Sample List

At this point, you can review the sample list, and if, during Method Setup, runtime editing of various capabilities (e.g., sample deletion, sample insertion, sample editing, sample sorting) was enabled, you can revise the sample list. The Sample List will only be displayed if one or more of inserting, deleting, sorting, or editing was selected in the Method Setup.

Inserting a Sample

Clicking on the **'Insert Sample'** button will insert a blank sample row directly above the currently selected row. This button will only be available if the option to permit runtime editing for sample insertion was enabled during the Method Setup. It will take on the default fields established during setup, and the user will need to directly enter the sample information into the grid.

Deleting a Sample

Clicking on the **'Delete Sample'** button will delete the currently selected sample. If multiple samples are selected, all selected samples will be deleted. This button will only be available if the option to permit runtime editing for sample deletion was enabled during the Method Setup.

Sample Editing

The software permits the editing of individual sample fields. The ability to edit existing samples depends on the options selected in the Method Setup and the LIMS/CDS Field Names Cross Reference Setup window. Samples that are added manually at this point can be edited regardless of the editing permissions set.

Sorting the Sample List

You can rearrange the sample list by clicking on the column headers to alternate between ascending and descending sorts based on the selected column. You also can move samples in the list by clicking and dragging a sample from one position in the list to another. For example, you can click and drag the 5th sample up to the 1st sample position. Alternatively, you can use the up and down arrows to move the selected sample up and down in the list, respectively.

Expanding the Sample List

Once the Sample List is correct, clicking on the **'Expand'** button automatically incorporates special samples such as controls and standards, set up via the Expansion Setup window, into the Sample Set.

If expansion is not enabled, the button will be labeled **'Create Sample Set Method'**. Clicking this button will upload the information to Empower.

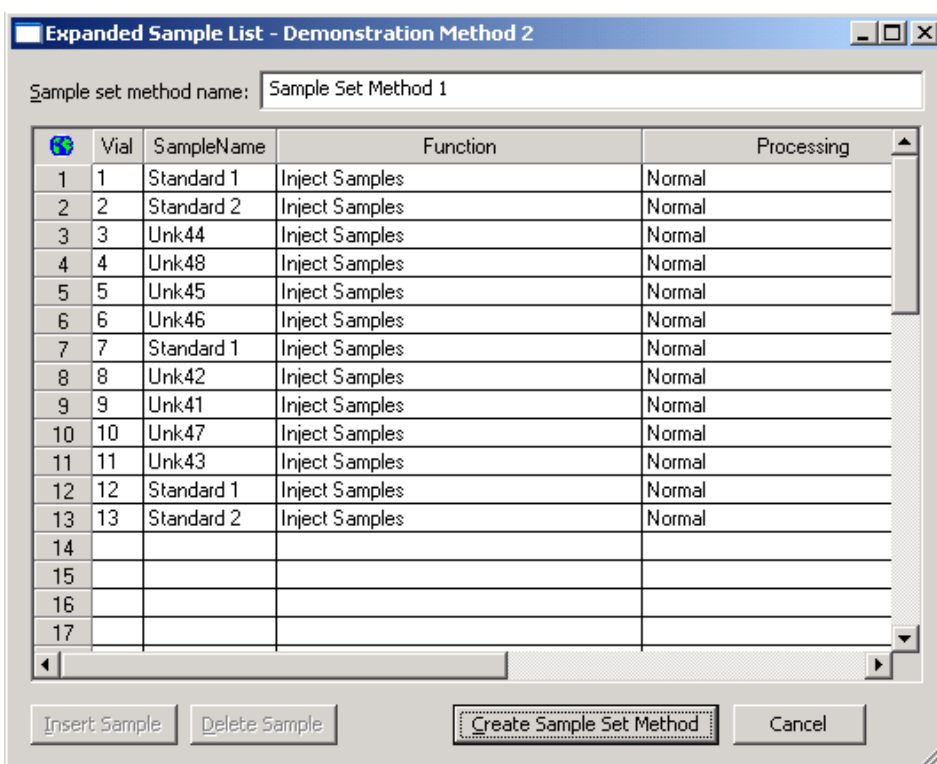


Figure 13: Expanded Sample List

Again, you have the option to insert and delete samples from the Expanded Sample List as well as edit individual samples, provided these runtime editing options were enabled during the Method Setup. There is no option to sort the Expanded Sample List.

Once the Expanded Sample List is correct, clicking on the **'Create Sample Set Method'** button, automatically uploads the information to Empower.

Note: If the Substitute Unknowns option for Sample Addition was selected in the setup, then the mappings specified in the Label Cross-Reference window (see Figure 7) will be applied to the samples before uploading the sample set.

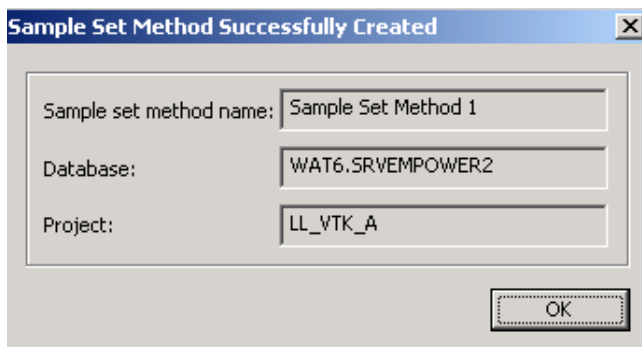


Figure 14: Sample Set Method Successfully Created

Viewing the Runtime Messages

Empower Message Center

The Empower Message Center stores and displays messages generated by the Empower application. All runtime errors pertaining to LimsLink EI are also displayed to the user via the Empower Message Center.

Audit Trail

If the Audit Trail option was enabled during setup via the Preferences window (and the project being used supports Audit Trail functionality), all changes will be recorded in the Empower Audit Trail.

Reporting Results to the LIMS

Using your LimsLink Method (note this is not the same Method used for the LimsLink EI application) and Labtronics ReportLink or an Export of the results acquired by Empower the analytical results can be reported to your LIMS.

Refer to the LimsLink EI User Manual as well as the ReportLink User Manual for further details.

Exporting Results to LIMS

To export results to LIMS, select the LimsLink EI – Results to LIMS menu item from the Empower 2 Client. Exports will only occur if the menu item is selected when in the 'Result Set' or 'Results' tab in the Empower browse project window. What happens once this menu item is selected is dependent on the configured project and how the run-time feature was accessed.

The following are possible scenarios which could occur:

Scenario 1 (occurs if the applicable Custom fields (previously described) have not been defined, or if the contents of the Custom fields are blank.)

In this situation, if you select a single result set, or one or more results in Empower, and then access the **LimsLink EI – Results to LIMS** menu item, you will be presented with a message indicating that the project is not configured to run any of the LimsLink EI LIMS Exports, and that you should check with your

Empower Administrator. No exports will be generated in this case. **Scenario 2** (occurs if one or more of the applicable Custom Fields (previously described) are present and contain valid contents)

In this situation, if you select a single result set, or one or more results in Empower, and then access the **LimsLink EI – Results to LIMS** menu item, the system will execute the specified exports. You will be presented with a message indicating that the exports were successfully generated.

Scenario 3 (occurs when you select more than one Result Set)

In this situation, if you select more than one result set, and then access the **LimsLink EI – Results to LIMS** menu item, you will be presented with a message indicating that you are permitted to export results for a single result set at a time. You will be prompted to select a single result set and retry. No exports will be generated in this case.

Scenario 4 (occurs if one or more of the applicable Custom Fields (previously described) are present but contain invalid contents)

In this situation, if you select a single result set, or one of more results in Empower, and then access the **LimsLink EI – Results to LIMS** menu item, the system will perform the valid exports, and you will be presented with a message indicating that there were errors in generating one or more of the exports.

Scenario 5 (occurs if one or more of the applicable Custom Fields (previously described) are present and contain valid contents, but one or more of the defined exporting paths cannot be written to)

In this situation, if you select a single result set, or one of more results in Empower, and then access the **LimsLink EI – Results to LIMS** menu item, the system will perform the valid exports, and you will be presented with a message indicating that there were errors in generating one or more of the exports.

Scenario 6 (occurs if you do not select a result set or results before selecting the **LimsLink EI – Results to LIMS** menu item)

In this situation, you will be presented with a message indicating that you must select a valid result set or results in order to execute the exports. No exports will be generated in this case.

Scenario 7 (occurs when the LL_NUM_SIGNOFFS Empower Custom Field is present and is populated with an integer greater than 0 and the results have at least that number of signoffs)

In this situation, if you select a single result set, or one of more results in Empower, and then access the **LimsLink EI – Results to LIMS** menu item, the system will verify that all results to be reported have the correct number of sign offs as defined in the Custom Field. The system will execute the specified exports. You will be presented with a message indicating that the exports were successfully generated.

Scenario 8 (occurs when the LL_NUM_SIGNOFFS Empower Custom Field is present and is populated with an integer greater than 0 and the results do NOT have at least that number of signoffs)

In this situation, if you select a single result set, or one of more results in Empower, and then access the **LimsLink EI – Results to LIMS** menu item, the system will execute the specified exports for the results that have the required number of signoffs. It will not perform the export for the results that do NOT have the required number of signoffs. You will be presented with a message indicating that exports were not successfully generated for results that did not have the required number of signoffs.

Scenario 9 (occurs when the LL_REPORTED Empower Custom Field is present and is not NULL.

Template Script

Empower Script

```
Option Explicit
```

```
Dim g_CDSSMod2 As CDSLINKLib.ICDSModule2
```

```
'-----
```

```
' Main runtime function.
```

```
'-----
```

```
Function Execute(ByVal LIMSSMod As CDSLINKLib.ILIMSSModule, ByVal CDSSMod As CDSLINKLib.ICDSModule,  
ByVal FieldMap As CDSLINKLib.FieldNameMap, ByVal SampleCont As CDSLINKLib.Samples, ByVal Expand As  
Boolean) As Boolean
```

```
    Execute = False
```

```
    On Error GoTo ErrHandler
```

```
    Dim objLIMS As Object
```

```
    Set objLIMS = LIMSSMod
```

```
    Dim objCDS As Object
```

```
    Set objCDS = CDSSMod
```

```
    Set g_CDSSMod2 = objCDS
```

```
    Dim Ret As Long
```

```
    Ret = vbOK
```

```
    ' Display the runtime display in the CDS module.
```

```

Dim CDSRuntime As CDSLINKLib.ICDSRuntimeDisplay
Set CDSRuntime = objCDS
If Not CDSRuntime Is Nothing Then
    Ret = CDSRuntime.DoRuntimeDisplay
    If Ret = vbCancel Then
        CDSMod.Logoff
        Exit Function
    End If
End If

' Initialize the fields in the sample container
' Any changes to the database or project global fields beyond this point will be ignored
CDSMod.InitializeFields SampleCont

' Get the samples from the LIMS module.
Ret = LIMSMOD.GetSamples( SampleCont )
If Ret = vbCancel Then
    CDSMod.Logoff
    LIMSMOD.Logoff
    Exit Function
End If

' Autonumber the samples' "Vial" field.
If Not g_CDSMod2 Is Nothing Then
    g_CDSMod2.AutoNumberField(SampleCont, "Vial", FieldKeyType_Name, 1, 1)
Else
    SampleCont.AutoNumberField("Vial", FieldKeyType_Name, 1, 1)
End If

' Display the sample list in the CDS module.
Dim CDSSampList As CDSLINKLib.ICDSSampleDisplay
Set CDSSampList = objCDS
If Not CDSSampList Is Nothing Then
    Ret = CDSSampList.DoSampleDisplay( SampleCont )
    If Ret = vbCancel Then

```

```

        CDSMod.Logoff
    Exit Function
End If
End If
' If we are using sample expansion...
If Expand Then
    ' Expand the samples if the CDS module supports expansion.
    Dim CDSExp As CDSLINCLib.ICDSExpansion
    Set CDSExp = objCDS
    If Not CDSExp Is Nothing Then
        CDSExp.ExpandSamples SampleCont

        ' Autonumber the samples' "Vial" field.
        If Not g_CDSMod2 Is Nothing Then
            g_CDSMod2.AutoNumberField(SampleCont, "Vial", FieldKeyType_Name, 1, 1)
        Else
            SampleCont.AutoNumberField("Vial", FieldKeyType_Name, 1, 1)
        End If
    End If
End If
' Display the sequence list in the CDS module.
Dim CDSSeqList As CDSLINCLib.ICDSSequenceDisplay
Set CDSSeqList = objCDS
If Not CDSSeqList Is Nothing Then
    Ret = CDSSeqList.DoSequenceDisplay( SampleCont )
    If Ret = vbCancel Then
        CDSMod.Logoff
        Exit Function
    End If
End If
End If
' Generate the sequence and upload it to the CDS.
Dim Status As Boolean

```

```

        Status = CDSMod.UploadSequence( SampleCont )

        ' Notify the LIMS module of the upload sequence status.
        LIMSMOD.SequenceCompleted Status

        LIMSMOD.Logoff

        CDSMod.Logoff

        Execute = True

        Exit Function

ErrorHandler:

        MsgBox Err.Description & vbCrLf & "The script will be terminated"

        LIMSMOD.Logoff

        CDSMod.Logoff

End Function

'-----
' Sample change event handler.
'-----

Sub CDSModuleEvents_OnSampleChange(ByVal Source As CDSLINKLib.SampleChange, ByVal ChangeType As
CDSLINKLib.SampleChangeType, ByVal SampleIndex As Long, ByVal SampleCont As CDSLINKLib.Samples,
ByRef Changed As Boolean)

        ' Autonumber the samples' "Vial" field.

        If Not g_CDSMod2 Is Nothing Then

                g_CDSMod2.AutoNumberField(SampleCont, "Vial", FieldKeyType_Name, 1, 1)

        Else

                SampleCont.AutoNumberField("Vial", FieldKeyType_Name, 1, 1)

        End If

        Changed = True

End Sub

'-----
' Sample edit event handler.
'-----

Sub CDSModuleEvents_OnSampleEdit(ByVal Source As CDSLINKLib.SampleChange, ByVal SampleIndex As Long,
ByVal FieldIndex As Long, ByVal SampleCont As CDSLINKLib.Samples, ByRef Changed As Boolean)

        ' Add code here

```

End Sub

'-----

' Sample sort event handler.

'-----

Sub CDSModuleEvents_OnSampleSort(ByVal FieldIndex As Long, ByVal SampleCont As CDSLINKLib.Samples,
ByRef Changed As Boolean)

 ' Autonumber the samples' "Vial" field.

 If Not g_CDSDMod2 Is Nothing Then

 g_CDSDMod2.AutoNumberField(SampleCont, "Vial", FieldKeyType_Name, 1, 1)

 Else

 SampleCont.AutoNumberField("Vial", FieldKeyType_Name, 1, 1)

 End If

 Changed = True

End Sub

Programming Reference

CDSLINKEMPOWERLib Library Objects

CDSMillennium Object

Description:

This is the CDS object used to interface with the Waters Empower CDS.

Functions and Properties

AllowEditCDSDatabase Property

Syntax:

object.AllowEditCDSDatabase

object

Data Type: CDSMillennium object

Any object variable returning a CDSMillennium object.

Description:

A read/write property that either sets or returns a Boolean value that determines if the Database may be changed in the Select CDS Parameters dialog.

AllowEditCDSProject Property

Syntax:

object.AllowEditCDSProject

object

Data Type: CDSMillennium object

Any object variable returning a CDSMillennium object.

Description:

A read/write property that either sets or returns a Boolean value that determines if the Project may be changed in the Select CDS Parameters dialog.

AllowEditSampleSetName Property*Syntax:*

object.AllowEditSampleSetName

object

Data Type: CDSMillennium object

Any object variable returning a CDSMillennium object.

Description:

A read/write property that either sets or returns a Boolean value that determines if the Sample Set Name may be changed in the Select CDS Parameters dialog.

MethodSet Property*Syntax:*

object.MethodSet

object

Data Type: CDSMillennium object

Any object variable returning a CDSMillennium object.

Description:

A read/write property that either sets or returns a string value that that is the name of the current MethodSet.

ReportMethod Property*Syntax:*

object.ReportMethod

object

Data Type: CDSMillennium object

Any object variable returning a CDSMillennium object.

Description:

A read/write property that either sets or returns a string value that is the name of the current report.

GetMethodSetList Function*Syntax:*

```
object.GetMethodSetList()
```

object

Data Type: CDSMillennium object

Any object variable returning a CDSMillennium object.

Return Value:

A Variant containing an array of strings that are the method set names available to the currently logged on user.

Description:

Retrieves a list of method set names available to the currently logged on user.

GetProjectList Function*Syntax:*

```
object.GetProjectList()
```

object

Data Type: CDSMillennium object

Any object variable returning a CDSMillennium object.

Return Value:

A Variant containing an array of strings that are the project names available to the currently logged on user.

Description:

Retrieves a list of project names available to the currently logged on user.

GetDatabaseList Function*Syntax:*

```
object.GetDatabaseList()
```

object

Data Type: CDSMillennium object

Any object variable returning a CDSMillennium object.

Return Value:

A Variant containing an array of strings that are the database names available to the currently logged on user.

Description:

Retrieves a list of database names available to the currently logged on user.

GetReportMethodList Function*Syntax:*

```
object.GetReportMethodList()
```

object

Data Type: CDSMillennium object

Any object variable returning a CDSMillennium object.

Return Value:

A Variant containing an array of strings that are the method names available for the current report.

Description:

Retrieves a list of method names available for the current report.

GetFieldList Function*Syntax:*

```
object.GetReportMethodList()
```

object

Data Type: CDSMillennium object

Any object variable returning a CDSMillennium object.

Return Value:

A Variant containing an array of field names available for the current project.

Description:

Retrieves a list of field names available for the current project.

ICDSMillennium2 Interface

Description:

This is an additional interface on the CDSMillennium object.

Properties

CDSDatabase Property

Syntax:

interface.CDSDatabase

interface

Data Type: ICDSMillennium2 object

Any object variable returning an ICDSMillennium2 interface.

Description:

A read/write property that either sets or returns the string value that is the name of the current Empower database. When this property is set it will attempt to logon to Empower using the new database and current project, user name and password.

CDSProject Property

Syntax:

interface.CDSProject

interface

Data Type: ICDSMillennium2 object

Any object variable returning an ICDSMillennium2 interface.

Description:

A read/write property that either sets or returns the string value that is the name of the current Empower project. When this property is set it will attempt to logon to Empower using the new project and current

database, user name and password.

SampleSetName Property

Syntax:

interface.SampleSetName

interface

Data Type: ICDSMillennium2 object

Any object variable returning an ICDSMillennium2 interface.

Description:

A read/write property that either sets or returns the current sample set name.

ExistingSampleSet Property

Syntax:

interface.ExistingSampleSet

interface

Data Type: ICDSMillennium2 object

Any object variable returning an ICDSMillennium2 interface.

Description:

A read/write property that either sets or returns the name of the selected existing sample set if the use existing sample set option is enabled.

SampleAddition Property

Syntax:

interface.SampleAddition

interface

Data Type: ICDSMillennium2 object

Any object variable returning an ICDSMillennium2 interface.

Description:

A read/write property that either sets or returns the sample addition option if the use existing

sample set option is enabled. The value is a SampleAddition enumeration value.

SampleAdditionPosition Property

Syntax:

interface.SampleAdditionPosition

interface

Data Type: ICDSMillennium2 object

Any object variable returning an ICDSMillennium2 interface.

Description:

A read/write property that either sets or returns the sample addition position if the use existing sample set option is enabled and the sample addition option is insert. The value is the index number at which to insert the new samples.

UpdateSelectedSampleSet Property

Syntax:

interface.UpdateSelectedSampleSet

interface

Data Type: ICDSMillennium2 object

Any object variable returning an ICDSMillennium2 interface.

Description:

A read/write property that either sets or returns the update selected sample set option if the use existing sample set option is enabled. The value is True or False.

UseExistingSampleSet Property

Syntax:

interface.UseExistingSampleSet

interface

Data Type: ICDSMillennium2 object

Any object variable returning an ICDSMillennium2 interface.

Description:

A read/write property that either sets or returns the use existing sample set option. The value is True or False.

Export File Formats

Generic Export File Format

```
# LIMSLINK STANDARD REPORT
LIMSLINK REPORT
Result Id 3258
Result Set Id
System Name System_198
Acquired By Analyst_1
Date Acquired 4/1/2010 10:37:32 PM
Project Name: Defaults
Database Service: Vm-emp2-nopatch
```

AcqMethodSet	AcqSWVersion	AcquiredBy	Altered	AutoAdditions	AverageDetectorDrift		
AverageDetectorNoise	AveragePeaktoPeakNoise	BarcodeBCD	CalibrationId	Channel	ChannelDescription		
ChannelId	ChannelName	ChannelType	Comments	DataEnd	DataStart	DateAcquired	
DateProcessed	DetectorDrift	DetectorNoise	DetUnits	Dilution	EmpowerNode	Faults	Injection
InjectionId	InjectionVolume	InjectionVolumeResult	InstrumentMethodId	InstrumentMethodName			
IntegrationAlgorithm	IntegrationSystemPolicies	Label	Level	LL_GEN_RPT	LL_IND_RPT	LL_MEAN_RPT	
Manual	NumberOfSignOffs	NumOfResultsStored	PDAExposureTime	PeaktoPeakNoise	PeakWidth		
PercentUnknowns	ProcessedAs	ProcessedBy	ProcessedChanDesc	ProcessedChannelType			
ProcessingLocked	ProcessingMethod	ProcessingMethodId	ProcessingNode	ResultCodes	ResultComments		
ResultId	ResultNum	ResultSampleSetMethod	ResultSetComments	ResultSetDate	ResultSetId		
ResultSetName	ResultSource	ResultSuperseded	ResultType	RunTime	SampleName		
SampleSetAcquiredBy	SampleSetAcquiring	SampleSetAltered	SampleSetComments				
SampleSetFinishDate	SampleSetId	SampleSetIdResult	SampleSetMethod	SampleSetName			
SampleSetStartDate	SampleType	SampleValuesUsedInCalculations	SampleWeight	SamplingRate			
ScaletopuV	SecondChannelId	SoftwareVersion	SourceSoftwareInfo	SummaryFaults	Superseded		
SystemComments	SystemCreateDate	SystemName	Task_ID	Threshold	TotalArea	Vial	VialIdResult
HP1100_system_A	Empower 2	Software Build 2154	SPs Installed:	Service Pack D	DB ID: 696003075	Analyst_1-1	
-6.96587629370877E-03	1.41378075738072E-04	5.29773817826289E-04	0	A1100 DAD AU			
Ch1	DAD AU Ch 1	Sample 230, Bw 5	3236	A1100 DAD AU Ch1	2	84	0
4/1/2010 10:37:32 PM	9/13/2010 4:06:21 PM	-1.32630143794091E-02	1.48261001727509E-03	AU			
4	Emastrwlmcbx4p	0	1	3235	100	100	3214
U1	SampleName Task_ID Height PctArea	SampleName Task_ID Height PctArea	-1	0			
5	6.90038433062411E-03	0	3	Analyst_2	DAD AU Ch 1	Sample 230,	
Bw 5	2	0	AM0180_HI_HPLC_B2036	3219	Emastrwlmhr3tn	3258	3
-1	4/2/2010 5:52:13 AM	0	1	84	200051657--AM0180B/1	Analyst_10	
4/1/2010 3:12:26 PM	1	3212	3212	B2036_AM0180_20100401	B2036_AM0180_20100401		
Injection Volume = 100,00 SampleWeight = 1,00000 4,00000Dilution = 4,00000 1							

2.5	9.99999997475243E-07	0	4	Empower 2 Software Build 2154 SPs Installed: Service Pack G Hotfix				
1 DB ID: 902408837	0	0	Agilent HP1100 system id A, kvalificering Str-002201				2/18/2010 10:15:39 AM	
System_198	300207754:1:1		38283870.1977175	3	3256	3256		
2ndDerivativeApex	Amount	Area	BaselineEnd	BaselineStart	Concentration	ControlValue	Curveld	
CurveRT	EndHeight	EndTime	Height	InflectionWidth	IntType	Name	Offset	
PctDeviation	PctHeight	PDA/FLRMatch1Angle	PDA/FLRMatch1Error	PDA/FLRMatch1Error	PDA/FLRMatch1Flag	PDA/FLRMatch1Flag	PctArea	
PDA/FLRMatch1Ideal		PDA/FLRMatch1Threshold	PDA/FLRMatch1WvinRMS	PDA/FLRMatch2Angle	PDA/FLRMatch2Threshold	PDA/FLRMatch2Flag		
PDA/FLRMatch2Error		PDA/FLRMatch2Flag	PDA/FLRMatch2Ideal	PDA/FLRMatch2Threshold	PDA/FLRMatch3Flag	PDA/FLRMatch3Flag		
PDA/FLRMatch2WvinRMS		PDA/FLRMatch3Angle	PDA/FLRMatch3Error	PDA/FLRMatch3Flag	PDA/FLRMatch3LibName	PDA/FLRMatch3LibName		
PDA/FLRMatch3Ideal		PDA/FLRMatch3Threshold	PDA/FLRMatch3WvinRMS	PDA/FLRMatch3LibName	PDA/FLRMatch3LibName	PDA/FLRMatch3LibName		
PDAMatch1SpectName		PDAMatch2LibName	PDAMatch2SpectName	PDAMatch2SpectName	PDAMatch3LibName	PDAMatch3LibName		
PDAMatch3SpectName		PeakCodes	PeakLabel	PeakLambdaMax	PeakLevel	PeakType		
PointsAcrossPeak	Purity1Angle	Purity1Flag	Purity1Threshold	Purity2Angle	Purity2Flag	Purity2Flag		
Purity2Threshold	Purity3Angle	Purity3Flag	Purity3Threshold	Purity4Angle	Purity4Flag	Purity4Flag		
Purity4Threshold	PurityErrors	RelativeResponse	RelativeRT	Response	RetentionTime	RetentionTime	RF	
RTRatio	Slope	StartHeight	StartTime	Units	Width			
42606.615500481	34.428833329156	31.668833329156				34	0	
34.428833329156	469.804389449152	bb	des-Phe B2036	-3.00466256485323E-03				
0.111291296518452		0.150679668154384						
	Q20	1		1	414			
					1	42606.615500481		
33.1547032776723		1.30138030964071E-04	0	31.668833329156		165.6		
	36810735.608256	49.6221666624894	36.9554999958227			42	8398.28393116089	
45.4954999958227	289119.640958361	bv	B2036 Main peak	-9.32886595460082E-03				
96.1520750596699		92.728915554833						
	Q20	2		1281				
				1				
3.07175465614388E-04	0	36.9554999958227	512.4	36810735.608256	41.4074378308095			
	632413.724730441	49.6221666624894	36.9554999958227			45	4452.11446154464	
47.0621666624894	8398.28393116089	vv	Trisulfide B2036	-9.32886595460082E-03				
1.651906459468		2.69356920504162						
	I06 S12 S21 S14 S27 S28 Q20		3		1	235		
						1	632413.724730441	
45.4954999958227		3.07175465614388E-04	8398.28393116089	45.4954999958227				
93.9999999999998								
	327621.976063199	49.6221666624894	36.9554999958227			47	0	
49.6221666624894	4452.11446154464	vb	Clip B2036	-9.32886595460082E-03				
0.85577026139518		1.4279200976335						
	I06 S12 S21 S14 S27 S28 Q20		4		1	384		
						1	327621.976063199	
47.0621666624894		3.07175465614388E-04	4452.11446154464	47.0621666624894			153.6	
	41301.2913793217	53.4554999958227	49.7821666624894			50	506.064762876739	
51.108833329156	782.58171751761	bv	Clip 1 B2036	1.86885152288532E-02				
0.107881703615702		0.250996278765087						
	S05 S07 S08 S09 S11 S21 S06 S10 S14 S27 S28 Q20				5		1	
199								
1	41301.2913793217	50.6727478890343			-2.56333481554219E-04		0	
49.7821666624894		79.5999999999997						
	351752.440004009	53.4554999958227	49.7821666624894			53	0	
53.4554999958227	6847.39421234427	vb	Clip 2 B2036	1.86885152288532E-02				
0.918800628534627		2.19615463543886						
	S05 S06 S14 S27 S28 Q20		6		1	352		
						1		

Clip 4 B2036	57	Missing
	3	57
77438.54178414 62.0954999958227 59.2221666624894	61	0
62.0954999958227 1720.34764522879 bb Clip 5 B2036	-7.10721201574162E-03	
0.202274590798181 0.551764560133608		
Q20 9	1 431	
60.772333166462 2.18136274966729E-04	0 59.2221666624894	77438.54178414 172.4
798114.24923067	46	68 13802.4380366353
Group Total Clip/Other B2036	2.08472718434369	4.42683557197106
	Q20 10	4
798114.24923067 47.0621666624894	46	1

Individual Export File Format

```
# LIMSLINK STANDARD REPORT
LIMSLINK REPORT
Task_ID 300221315:4:1
Result Id 2066
Result Set Id 2054
SampleName 200024195--Empower Operation 2
System Name Alliance
Acquired By System
Date Acquired 9/17/1997 5:37:56 PM EDT
Project Name: Defaults_FAT
Database Service: TEST
Peak Results
# Name Amount % Area
1 Acetone 3740.906 28.02
2 Acetophenone 9.973 27.63
3 Propiophenone 9.972 21.99
4 Butyrophenone 9.980 22.36
```

Mean Export File Format

```
# LIMSLINK MEAN REPORT
LIMSLINK MEAN REPORT
Result Set Id 2054
```

Acquired By System

Project Name: Defaults_FAT

Database Service: TEST

System Name Alliance

Peak Summary with Statistics

Name: Acetone

# SampleName	Task_ID	Name	% Area	Amount	Result	Id
1 200024195--Empower 2065	Operation 2	300221315:3:1 Acetone	28.0397			3755.8534
2 200024195--Empower 2066	Operation 2	300221315:4:1 Acetone	28.0232			3740.9058
Mean		28.0314		3748.3796		
% RSD		0.0416	0.2820			
Std. Dev.		0.0116	10.5695			

Peak Summary with Statistics

Name: Acetophenone

# SampleName	Task_ID	Name	% Area	Amount	Result	Id
1 200024195--Empower 2065	Operation 2	300221315:3:1 Acetophenone	27.6173			10.0018
2 200024195--Empower 2066	Operation 2	300221315:4:1 Acetophenone	27.6318			9.9730 2066
Mean		27.6245		9.9874		
% RSD		0.0371	0.2037			
Std. Dev.		0.0102	0.0203			

Peak Summary with Statistics

Name: Butyrophenone

# SampleName	Task_ID	Name	% Area	Amount	Result	Id
1 200024195--Empower 2065	Operation 2	300221315:3:1 Butyrophenone	22.3558			10.0128
2 200024195--Empower 2066	Operation 2	300221315:4:1 Butyrophenone	22.3583			9.9800 2066
Mean		22.3570		9.9964		
% RSD		0.0080	0.2320			
Std. Dev.		0.0018	0.0232			

Peak Summary with Statistics

Name: Propiophenone

# SampleName	Task_ID	Name	% Area	Amount	Result	Id
1 200024195--Empower 2065	Operation 2	300221315:3:1 Propiophenone	21.9873			10.0061
2 200024195--Empower 2066	Operation 2	300221315:4:1 Propiophenone	21.9867			9.9719 2066
Mean		21.9870		9.9890		

% RSD	0.0017	0.2420
Std. Dev.	0.0004	0.0242

MOLWT Export File Format

LIMSLINK FRAGMIN REPORT

Task_ID 300212480

Result Id 2612

Result Set Id

SampleName 47484-51 Etot

System Name System_019_B

Date Acquired 6/23/2010 2:21:05 PM CEST

Acquired By Analyst_1

Total Area 2575594

Project Name: Defaults

Database Service: TEST

Molecular weight parameters

of Fragmin

# Mn	Mw	MP	Retention Time
------	----	----	----------------

1	4822	5951	4508	8.340
---	------	------	------	-------

MW distribution of Fragmin

# MW Markers	Cumulative %	Mp: 4508 Name: Peak1
--------------	--------------	----------------------

1	3000	91.018
---	------	--------

2	8000	18.897
---	------	--------

3	10000	9.751
---	-------	-------

Galaxie

Introduction

LimsLink Embedded Interface, an accessory module to the Labtronics LimsLink CDS product, provides an embedded link between your CDS and LIMS. Designed to be a Plug & Play system, you can substitute in various CDS and LIMS modules to work with the interface, and set up the individual modules to suit your requirements.

The Galaxie module is suitable for use with Varian Galaxie version 1.8 or 1.9 accompanied by the Galaxie 1.8 or 1.9 toolkit, and is compatible with any LIMS module, selected during installation.

The Galaxie module provides the functionality required to view sample lists retrieved from the LIMS, edit and expand the sample list to include control samples, and then convert this list to a Sequence now ready for analysis by the Galaxie application. Once analysis is complete, the Galaxie module then allows for the reporting of results back to the LIMS.

As part of the LimsLink Embedded Interface installation procedure, you are required to select the CDS and LIMS module for the installation. Upon selecting the Varian Galaxie CDS, the Galaxie module is then installed as part of your LimsLink Embedded Interface installation.

The LimsLink Embedded Interface installation creates the menu items '**LimsLink – Create Sequence**' and '**LimsLink – Method Setup**'. These appear as sub-menu items of the '**Plug-ins**' menu option contained in the Galaxie application.

With the Embedded Interface application, LIMS integration is embedded directly into your Galaxie application providing users with a direct and transparent connection to their LIMS.

Installation

Refer to Chapter 2 of the LimsLink Embedded Interface manual for installation instructions of the application as a whole. Information noted here pertains only to the Galaxie module.

Software Requirements

The LimsLink Embedded Interface requires that the following software be installed, and operational prior

to installing the embedded interface:

- Galaxie version 1.8 or 1.9
- Galaxie toolkit version 1.7 (or higher)

References

The documentation for the Galaxie system only refers to that information which pertains to the direct setup and runtime of the LimsLink Embedded Interface with respect to the Galaxie application. You will need to refer to the LimsLink EI User Manual for specifics relating to the setup of the application as a whole.

System Setup

A set of parameters governing the transfer of information between the LIMS and the CDS must first be established before data retrieval and analysis can occur. This information would typically be configured by someone with administrative privileges, and would only need to be configured once during the initial setup.

To set up the system parameters, perform the following steps:

1. From the Galaxie main screen, pull down the **Plug-ins** menu item.
2. Select the LimsLink – Method Setup sub-menu item to open the LimsLink Embedded Interface Setup window, similar to the example shown here.

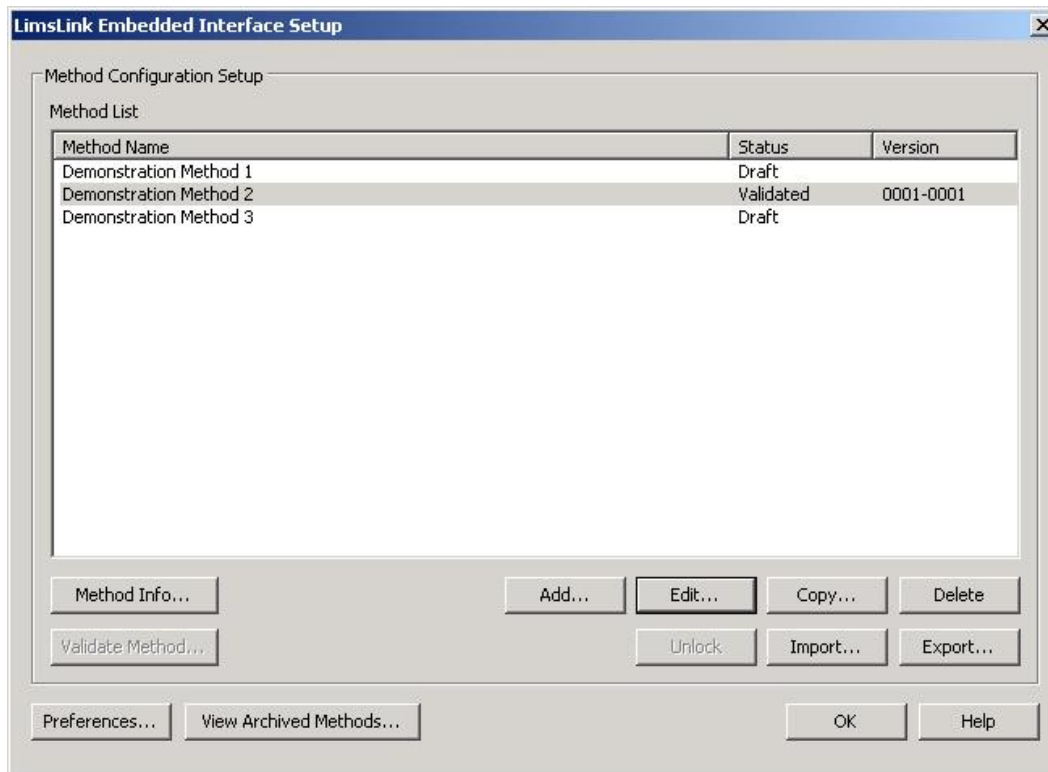


Figure 1: LimsLink Embedded Interface Setup

3. Click on the '**Preferences**' button on the LimsLink Embedded Interface Setup window to open the Preference window, similar to the example shown here.

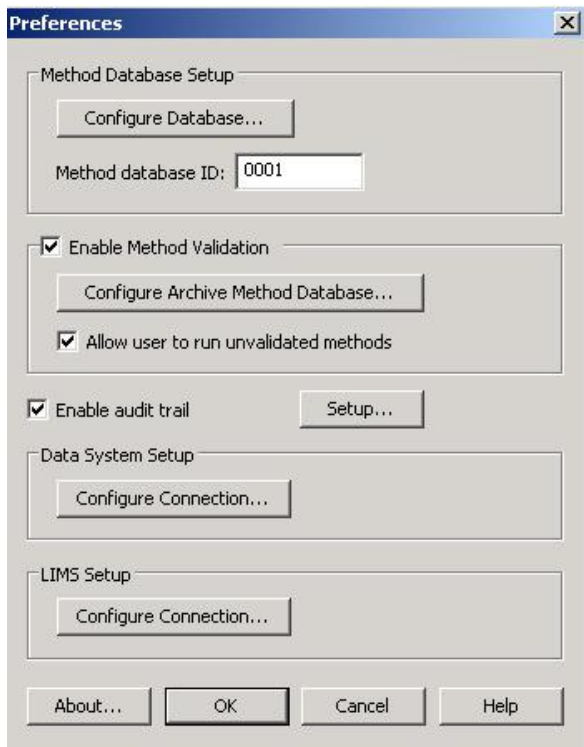


Figure 2: Preferences

System Preferences

Through this window, you are able to determine the location of data repositories such as the Method Database and the Archive Method Database. Other specifics such as Method Validation can be established here. Refer to Chapter 3 - System Setup in the LimsLink Embedded Interface manual for additional details.

Galaxie Configuration Setup

To verify the Galaxie and the Galaxie toolkit version numbers detected by the LimsLink Embedded Interface application, click on the '**Configure Connection**' button within the Data System Setup group box to open the Galaxie Configuration Setup window. The Galaxie application requires no additional configuration, and therefore the window will simply display an information box.

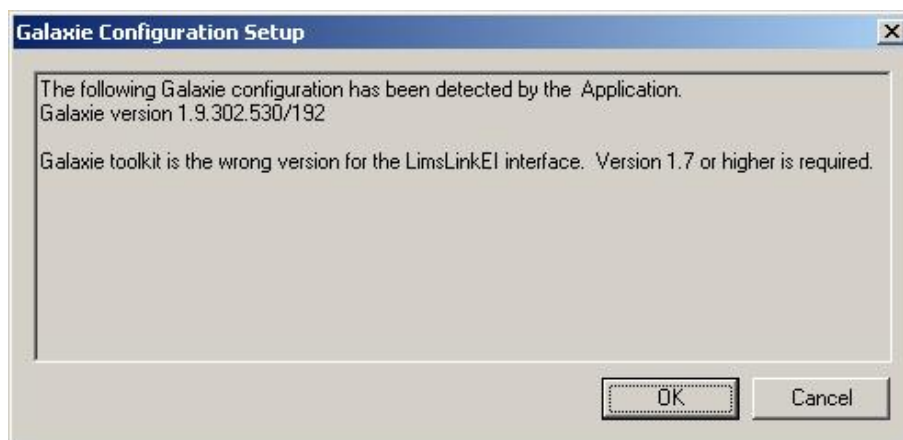
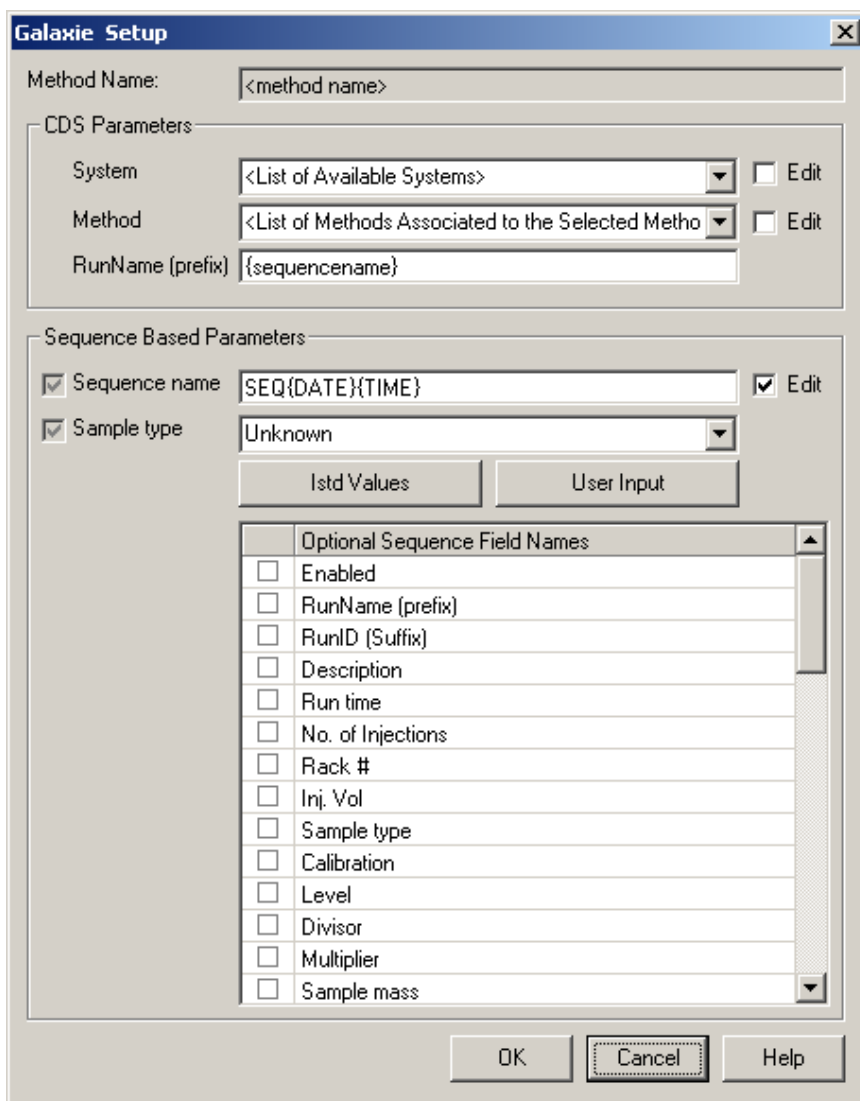


Figure 3: Galaxie Configuration Setup

Defining the CDS Parameters

Accessing the Galaxie Setup

1. To access the Method Setup window, click on the **'Add'** button from the LimsLink Embedded Interface Setup window
2. From the Method Setup window, click on the **'Data System Parameters Setup'** button to open the Galaxie Setup window.



The Galaxie Setup dialog box is divided into two main sections: CDS Parameters and Sequence Based Parameters.

CDS Parameters:

- Method Name:** A text field containing the placeholder text "<method name>".
- System:** A dropdown menu showing "<List of Available Systems>" with an "Edit" checkbox to its right.
- Method:** A dropdown menu showing "<List of Methods Associated to the Selected Metho..." with an "Edit" checkbox to its right.
- RunName (prefix):** A text field containing the placeholder text "{sequencename}".

Sequence Based Parameters:

- Sequence name:** A checked checkbox followed by a text field containing "SEQ{DATE}{TIME}" and an "Edit" checkbox.
- Sample type:** A checked checkbox followed by a dropdown menu showing "Unknown".
- Istd Values:** A button.
- User Input:** A button.
- Optional Sequence Field Names:** A list box with a scroll bar containing the following items, each with an unchecked checkbox:
 - Enabled
 - RunName (prefix)
 - RunID (Suffix)
 - Description
 - Run time
 - No. of Injections
 - Rack #
 - Inj. Vol
 - Sample type
 - Calibration
 - Level
 - Divisor
 - Multiplier
 - Sample mass

At the bottom of the dialog are three buttons: OK, Cancel, and Help.

Figure 4: Galaxie Setup

Setting up Galaxie

The Galaxie Setup allows you to define the information that is required to create the Sample List or the list of samples to be analyzed by the Galaxie CDS. The CDS Parameters section defines the System and Method to be used, and allows a default value to be set for the RunName (prefix) field. The Sequence Based Parameters section allows you to select the fields that will be uploaded to the CDS and allows you to define default values for some of the sequence fields, as well as name the sequence.

Method Name – The name of the current Method is displayed. It is not editable.

Defining the CDS Parameters

System

From here, you can select the system name to be used for sequence analysis. The drop down list box will be comprised of all system names that are currently available to the Galaxie Client.

You may permit runtime editing of the System name by checking the associated Edit check box.

Method

You can select the default Method that will be used for all samples to be contained within the Sequence. The list box, supplied by the Galaxie Client, will contain all Methods available to the current user for the selected system.

Methods are system dependent, so changing the system may re-populate the Method list box.

You may permit runtime editing of the Method name by checking the associated Edit check box.

RunName (prefix)

Here you can specify a prefix that will be used to create the Sample name. The complete Sample name will be the concatenation of the RunName (prefix), sequence name, and RunID (suffix). In most cases, the RunName(prefix) is used as a sample name equivalent.

Defining the Sequence Based Parameters

Sequence name

Here you can enter a name for the Sequence. You have the option to use a combination of a date and/or time macro within the Sequence name to allow a unique naming convention, or you can enter a fixed Sequence name that will be applicable to all Sequences created using the selected Method.

If using the {DATE} macro, the software, at runtime, will substitute a six digit (YYMMDD) value that will represent the current date. If using the {TIME} macro, the software, at runtime, will substitute a six digit (HHMMSS) value that will represent the current time (24 hour format). For example, the Sequence name with prefix SEQ and the date and time variable would be entered as SEQ{DATE}_{TIME}. The Sequence name would be SEQ090414_230521 if the Sequence were run on April 14, 2009 at 11:05:21 PM.

You may permit runtime editing of the Sequence name by checking the associated Edit check box.

Sample Type

Here you can specify the default Sample Type. The Type field is a mandatory part of the Sequence, and therefore the accompanying check box is disabled.

Istd values

Clicking on the '**Istd Values**' button will open an Internal Standards Values window, similar to the example shown here.

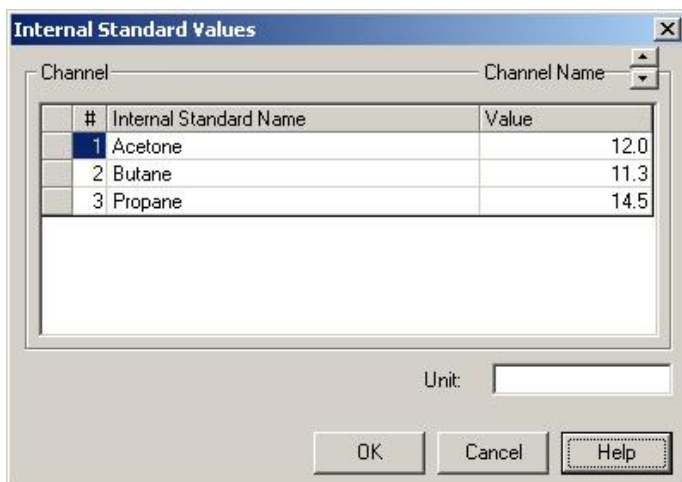


Figure 5: Internal Standard Values

The dialog shows all the internal standards (e.g., acetone, butane, propane) that have been configured for the selected Method, along with their respective values. From here, you can set default values for each internal standard.

Whether or not the internal standards and their values are editable is dependent on the Method. If a Method does not have Internal Standards, this window will not be available.

A similar dialog can be opened, at run-time, through the Sample/Expanded Sample lists on a per sample basis, allowing these values to be modified there.

Multiple channels can exist on a system; each channel can have a different list of Internal Standard Values (dependant on the Galaxie Method being used). The scroll control in the top right hand corner of the window allows you to switch between different channels; the current channel's name will replace 'Channel Name' in the group box.

User Input

Clicking on the **'User Input'** button will open a Variables window similar to the example shown here.

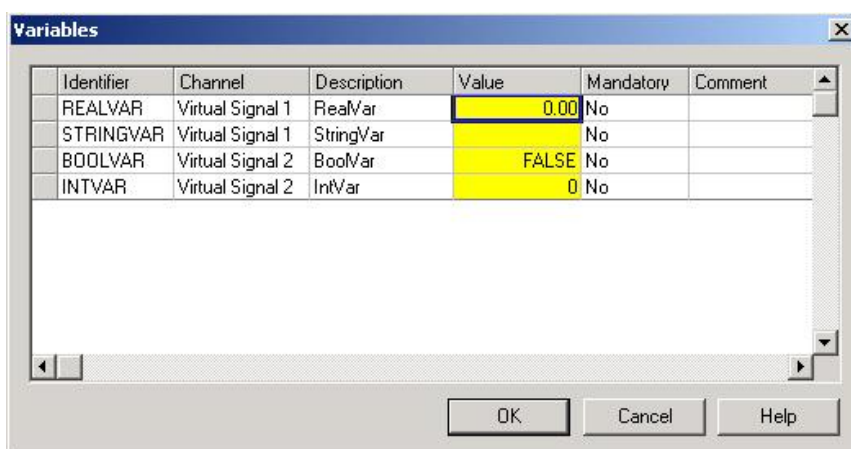


Figure 6: Variables

From here, you can view and enter the default values for the User Input variables configured for the selected Method. The variable value is defined via the Method and retrieved using the Galaxie toolkit. From here, you can set the default values for the variables.

Whether or not the User Input variables are editable is dependent on the Method. If a Method does not have any User Input Variables configured, this window will not be available.

You are not permitted to enter additional input variables here, but only enter and/or edit the value for each existing variable.

The following columns are also displayed on the grid:

Identifier – displays the Identifier name associated with the User Input variable. The Identifier field is non-editable.

Channel – displays the Channel associated with the User Input variable. The Channel field is non-editable.

Description – displays the associated description for the User Input variable. The Description field is non-editable.

Value – displays the associated value for the User Input variable. If a value does not exist for the User Input variable, a default value is entered. The default values are as follows:

- If the variable type is String, the default value is blank.
- If the variable type is Real, the default value is 0.00.
- If the variable type is Integer, the default value is 0.
- If the variable type is Boolean, the default value is 'FALSE'.

If the Mandatory field is set to 'Yes' (see below), the default value is blank, and you must enter a value or string depending on the variable type in order to exit from the dialog box.

You are permitted to edit the value displayed in the cell.

Mandatory – if a value for the User Input Variable is mandatory, the column cell will display 'Yes'. If an entry is not mandatory, 'No' will be displayed. The Mandatory field is non-editable.

Comment – displays the Comment associated with the User Input variable. The Comments field is non-editable.

Note: A similar dialog can be opened, at runtime, through the Sample/Expanded Sample lists on a per sample basis, allowing these values to be modified there as well.

Optional Sequence Field Names

A series of optional sequence field names from the Galaxie Client, is presented. From here, you can select additional field names to be uploaded to the Sequence List.

Activating the check box for any additional fields will result in the respective field being uploaded to Galaxie at runtime.

If Istd values exists for the selected Method, the Istd values will be appended to the optional sequence field names list using the following format:

Istd Value – Istd variable name (Channel Name)

For example, if a value is required for Acetone (Istd variable name) for Virtual Signal 1 (channel name), the entry in the optional sequence list will be Istd Value – Acetone (Virtual Signal 1).

If Variables exists for the selected Method, the Variables will be appended to the optional sequence field names list using the following format:

Variable – Variable name (Channel Name) For example, if a value is required for LIMS ID (Variable name) for Virtual Signal 1 (channel name), the entry in the optional sequence list will be Variable – LIMS ID (Virtual Signal 1).

LIMS/CDS Field Name Cross Reference Setup

Via the LIMS/CDS Field Name Cross-Reference Setup (accessed from the Method Setup window), you mapped the field names of samples originating from the LIMS to the field names required by Galaxie.

Editable	CDS Field Name	LIMS Field Name	Default Value
<input type="checkbox"/>	Vial #		
<input checked="" type="checkbox"/>	Method		
<input type="checkbox"/>	RunName (prefix)		
<input checked="" type="checkbox"/>	Description		
<input checked="" type="checkbox"/>	Run time		
<input type="checkbox"/>	No. of Injections		1
<input checked="" type="checkbox"/>	Rack #		
<input type="checkbox"/>	Inj. Vol.		1
<input checked="" type="checkbox"/>	SampleType		Unknown
<input type="checkbox"/>	Level		1

Figure 7: LIMS/CDS Field Name Cross Reference Setup

Through this same window, you are able to define default values for each field. The default value will be used if no value is retrieved from the LIMS for the field.

Refer to the LimLink EI User Manual for more information on the LIMS/CDS Field Cross Reference Setup.

Running a Method

All runtime activities are accessible via Galaxie. For example, sample retrieval from the LIMS, modifications to the retrieved sample list, and subsequent uploading of the sequence to Galaxie for analysis.

Note: The runtime dialogs reflect the combination of CDS and LIMS being used. You will want to reference your specific LIMS documentation contained within the LimsLink EI User Manual to review the runtime dialogs specific to your LIMS. Refer to the section entitled 'Runtime Behavior'.

Accessing the Methods

To access the available Methods, perform the following steps:

1. From the Galaxie main screen, pull down the **Plug-ins** menu item.
2. Select the **LimsLink – Create Sequence** sub-menu item to open the Select Method to Run window. This window will only open if more than one Method exists for the current system. If only one Method exists, the Method will start automatically upon selecting the **LimsLink – Create Sequence** sub-menu item.

The Method list is retrieved from the Method Database defined via the Preferences window (see Figure 2). If the Method validation option is activated via the Preferences window, only validated Methods will be displayed and available for selection. If the option to 'Allow user to run unvalidated methods' is activated via the Preferences window then all Methods currently configured for the system will be displayed and available for selection.

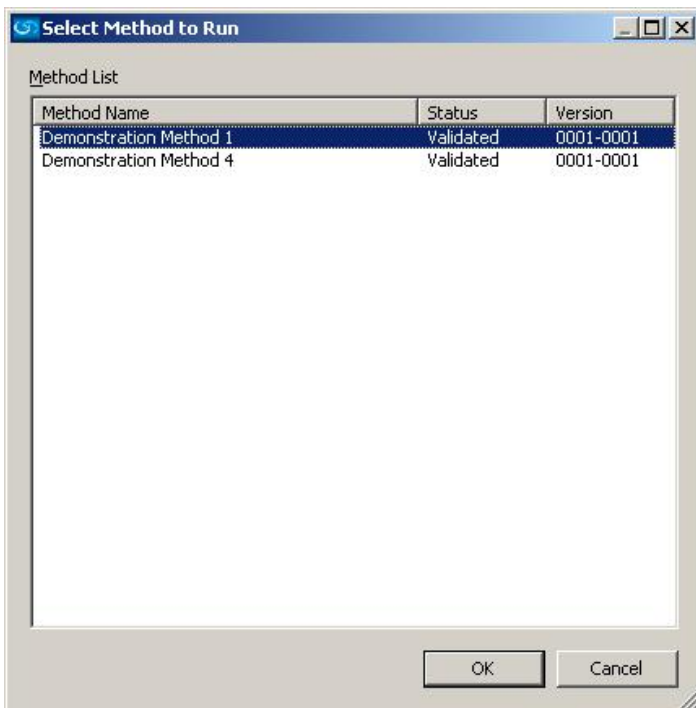


Figure 8: Select Method to Run

3. Highlight the Method to run and click on the '**OK**' button.

Selecting CDS Parameters

The next dialog will be presented only if one or more of its parameters (i.e., System, Methods) can be edited. If through the Data System Parameters Setup, the System and Methods to be used were defined and non-editable at runtime, you will not see this window. Once you have made your selections, click on the '**OK**' button.

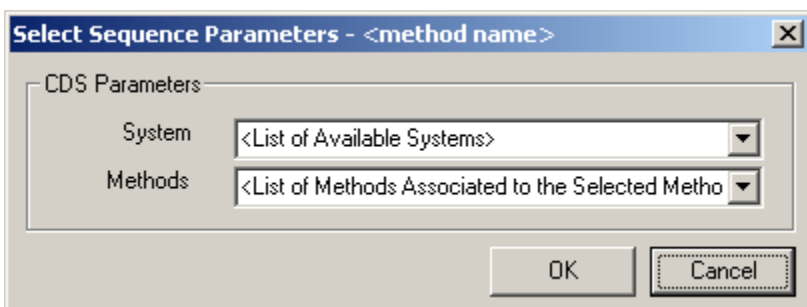


Figure 9: Select Sequence Parameters

Selecting LIMS Parameters

The contents of the next dialog(s) will vary depending on the LIMS being used, and the Method Setup. You will want to reference your specific LIMS module documentation to see the actual runtime dialog(s). This information is contained within the LimsLink EI User Manual. Refer to the section entitled 'Runtime Behavior'.

Viewing the Sample List

Once the list of samples to be analyzed by Galaxie has been determined, the software will subsequently display the Sample List containing those samples.

Note: The contents of the next dialog will vary depending on the Method Setup. Not all options may be available.

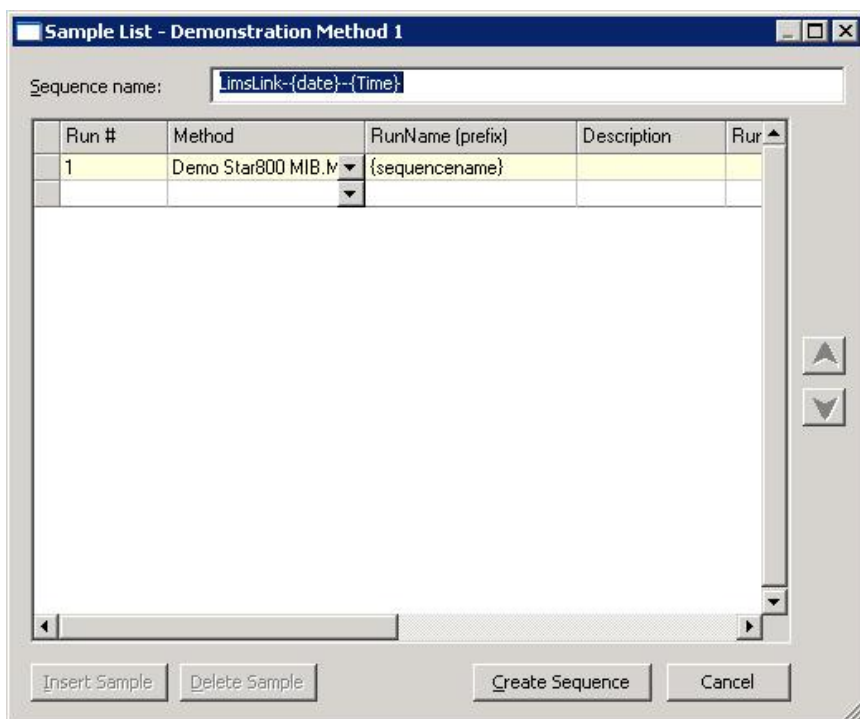


Figure 10: Sample List

Editing the Sample List

At this point, you can review the sample list, and if, during Method Setup, runtime editing of various capabilities (e.g., sample deletion, sample insertion, sample editing, sample sorting) was enabled, you can revise the sample list. The Sample List will only be displayed if one or more of inserting, deleting, sorting, or editing was selected in the Method Setup.

Inserting a Sample

Clicking on the **'Insert Sample'** button will insert a blank sample row directly above the currently selected row. This button will only be available if the option to permit runtime editing for sample insertion was enabled during the Method Setup. It will take on the default fields established during setup, and the user will need to directly enter the sample information into the grid.

Deleting a Sample

Clicking on the **'Delete Sample'** button will delete the currently selected sample. If multiple samples are selected, all selected samples will be deleted. This button will only be available if the option to permit runtime editing for sample deletion was enabled during the Method Setup.

Sample Editing

The software permits the editing of individual sample fields. The ability to edit existing samples depends on the options selected in the Method Setup and the LIMS/CDS Field Names Cross Reference Setup window. Samples that are added manually at this point can be edited regardless of the editing permissions set.

Sorting the Sample List

You can rearrange the sample list by clicking on the column headers to alternate between ascending and descending sorts based on the selected column. You also can move samples in the list by clicking and dragging a sample from one position in the list to another. For example, you can click and drag the 5th sample up to the 1st sample position.

Re-ordering the Sample List

Using the Up Arrow and Down Arrow buttons located on the right hand side of the window, you are able to selectively move a sample either up or down in the Sample List.

Expanding the Sample List

Once the Sample List is correct, clicking on the '**Expand**' button automatically incorporates special samples such as controls and standards, set up via the Expansion Setup window, into the Sequence.

If expansion is not enabled, the button will be labeled '**Create Sequence**'. Clicking this button will upload the data to Galaxie.

Again, you have the option to insert and delete samples from the Expanded Sample List as well as edit individual samples, provided these runtime editing options were enabled during the Method Setup. There is no option to sort the Expanded Sample List or re-order the Sample List.

Once the Expanded Sample List is correct, clicking on the '**Create Sequence**' button, automatically uploads the data to Galaxie.

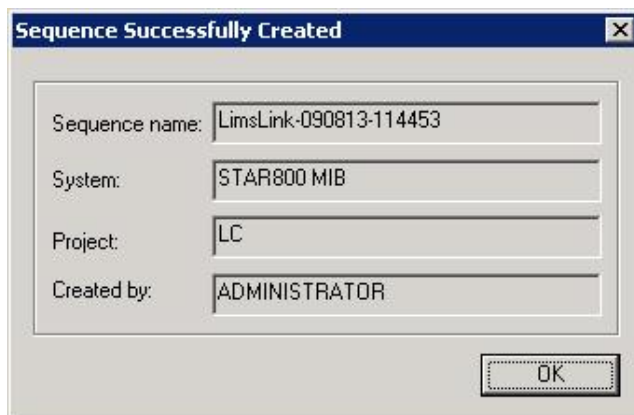


Figure 10: Sequence Successfully Created

Reporting Results to the LIMS

Using your LimsLink CDS Method (note this is not the same Method used for the embedded interface application) and Labtronics ReportLink, the results acquired by Galaxie can be reported to your LIMS. A status report message will optionally be returned back to Galaxie, indicating the results were successfully written to the LIMS. This message will be placed in a window that appears on screen.

Refer to the LimsLink CDS User Manual for more information on a LimsLink Method and ReportLink.

The TCP/IP based message sent from LimsLink to LimsLink Embedded Interface must follow a particular format. The format is as follows:

```
<message length>LLEI<CRLF>Default<CRLF><message>
```

where <message length> is the 4 byte binary representation of the length of the remainder of the message. For instance,

```
{23}{0}{0}{0}LLEI{13}{10}Default{13}{10}Uploaded
```

would be a valid message. If the LabtronicsWinSock.Client COM object (installed with LimsLink EI) is used for communication, the message length will automatically be included at the beginning of the message.

The following is an example of a LimsLink ScriptReporter script that sends a message to the LimsLink Embedded Interface application:

```
Sub OnRunFormatter(ByVal RepRunStruct As Variant, ByRef Status As Boolean)

    On Error GoTo errhandler

    Status = True ' Do not run post reporting.

    Dim winsoc As New LABTRONICSWINSOCKLib.Client

    Dim message As String
    message = "LLEI" + vbCrLf + "Default" + vbCrLf + "Uploaded"

    winsoc.Connect "<computer name>", <port number>
    winsoc.SendEx message
    winsoc.CloseConnection

    Exit Sub
errhandler:
    MsgBox "An error occurred"
End Sub
```

In the above script, <computer name> represents the name of the machine to which the message is to be sent. The computer name can be determined by adding the name of the computer sending data to LimsLink to the Info Block of ReportLink's Printing Preferences. The LimsLink Method can then parse this data out.

In the above script, <port number> represents the port on which the LimsLink Embedded Interface is listening for messages. This value can be set in the Preferences window and defaults to 3012.

LimsLink EI™

Version 4.2

LIMS Modules



SQL Query LIMS

Introduction

LimsLink EI, a component of the Labtronics LimsLink CDS product, provides an embedded link between your CDS and LIMS. Designed to be a Plug & Play system, you can substitute in various CDS and LIMS modules to work with the interface, and set up the individual modules to suit your requirements.

The SQL Query LIMS module is a generic module for any LIMS which supports a SQL query interface. As a minimum, the following LIMS are supported:

Applied Bio SQL*LIMS
Aspen LIMS
ChemWare Horizon LIMS
Honeywell Uniformance LIMS
LabSys LIMS
LabVantage Sapphire LIMS
LabWare LIMS
Matrix LIMS
QSI WinLIMS
SampleManager LIMS

Other LIMS may be supported using the Generic SQL Query LIMS option.

As part of the LimsLink EI installation procedure, you are required to select the CDS and LIMS module for the installation. Upon selecting one of the above listed LIMS, the SQL Query LIMS module is then installed as part of your LimsLink EI installation. The SQL Query LIMS module provides the functionality to retrieve the sample list from the LIMS.

With the LimsLink EI application, LIMS integration is embedded directly into your CDS client providing users with a direct and transparent connection to their LIMS.

Installation

Refer to the LimsLink EI Installation and Configuration Manual for installation instructions of the application as a whole. Information noted here pertains only to the SQL Query LIMS module.

In order to access the LIMS with the SQL Query LIMS module, you must ensure that an appropriate database provider and database client are available.

References

The documentation for the SQL Query LIMS only refers to that information which pertains to the direct setup and runtime of LimsLink EI with respect to the SQL Query LIMS module. You will need to refer to the LimsLink EI User Manual for specifics relating to the setup of the application as a whole.

System Setup

A set of parameters governing the transfer of information between the LIMS and the CDS must first be established before information retrieval and analysis can occur. This information would typically be configured by someone with administrative privileges, and would only need to be configured once during the initial setup.

To access the setup dialogs, select the **LimsLink EI – Method Setup** menu item or toolbar button from the CDS client. This will open the LimsLink Embedded Interface Setup window.

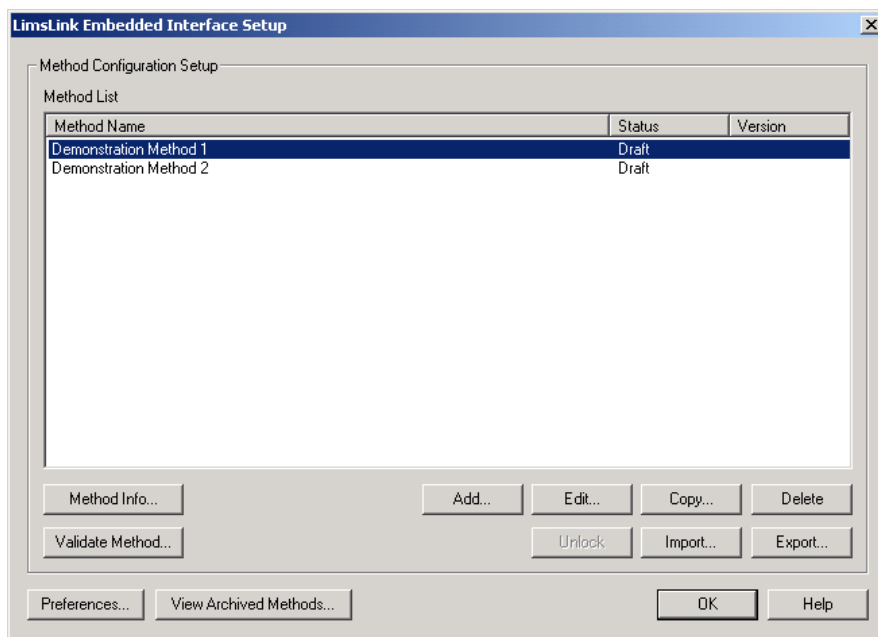


Figure1: LimsLink Embedded Interface Setup

System Preferences

Clicking on the '*Preferences*' button on the LimsLink Embedded Interface Setup window will open the Preferences window, similar to the example shown here.

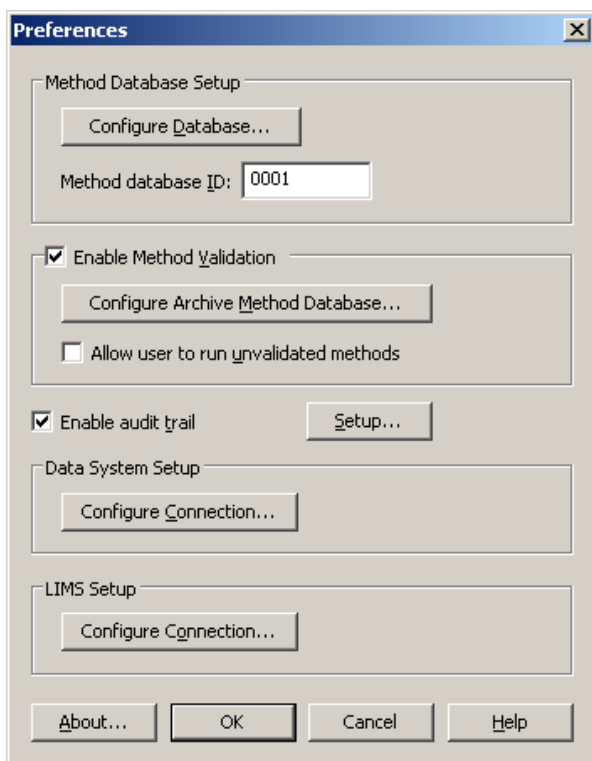


Figure 2: Preferences

Through this window, you are able to determine the location of data repositories such as the Method Database and the Archive Method Database. Other specifics such as Method Validation, and Audit Trail can be established here. Refer to Chapter 2 - System Setup in the LimsLink EI User Manual for details.

SQL Query LIMS Configuration Setup

Clicking on the '**Configure Connection**' button within the LIMS Setup group box will open the LIMS Configuration Setup window.

Note: The actual dialog title will reflect the type of LIMS selected during the installation.

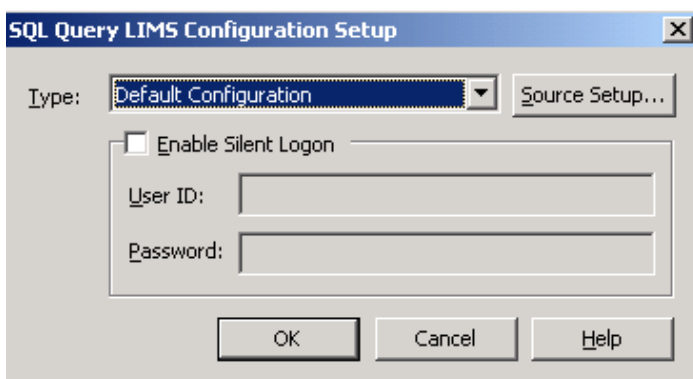


Figure 3: SQL Query LIMS Configuration Setup

LIMS Type – From the drop down list box, select the name and versions of the LIMS application to which you want to connect to and retrieve data from. Currently the only available option is 'Default Configuration'.

Clicking on the '**Source Setup**' button will open a Data Link Properties window through which you can define the connection to the LIMS database.

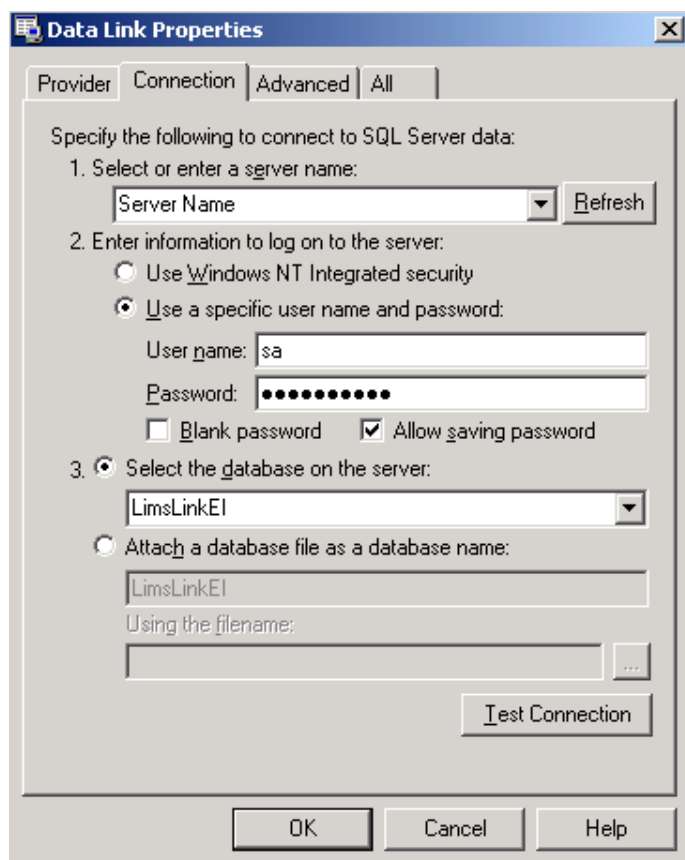


Figure 4: Data Link Properties

Enable Silent Logon

Enabling the Silent Logon option in the SQL Query LIMS Configuration Setup window allows you to bypass logging onto the LIMS each time you access the LIMS (e.g., to query the database). Here you must enter the User ID and Password required to access the LIMS. Each time you subsequently access the LIMS during setup and runtime, you will be 'silently' logged on, and will not be required to enter a User ID and Password each time.

Defining the LIMS Parameters

Method Setup

Recall that a Method is responsible for retrieving information from the LIMS, modifying the retrieved

information, and subsequently uploading the information to the CDS. As part of the Method Setup, the parameters for the respective LIMS being used need to be defined in order that the correct information be retrieved from the LIMS.

Accessing the SQL Query LIMS Setup

1. To access the Method Setup window, click on the **'Add'** button from the LimsLink Embedded Interface Setup window.
2. Click on the **'LIMS Parameters Setup'** button to open the SQL Query LIMS Setup window.

Note: The actual dialog title will reflect the type of LIMS selected during the installation.

Setting up the SQL Query LIMS

SQL Query LIMS Setup

Method name:

☒ Over-ride global connection settings

Configure Runtime Setup

Parameter		Default selection	
<input type="text" value="Parameter Name"/>	<input type="button" value="Create list..."/>	<input type="text" value="<none>"/>	<input type="checkbox"/> Edit
<input type="text"/>	<input type="button" value="Create list..."/>	<input type="text" value="<none>"/>	<input type="checkbox"/> Edit
<input type="text"/>	<input type="button" value="Create list..."/>	<input type="text" value="<none>"/>	<input type="checkbox"/> Edit
<input type="text"/>	<input type="button" value="Create list..."/>	<input type="text" value="<none>"/>	<input type="checkbox"/> Edit
<input type="text"/>	<input type="button" value="Create list..."/>	<input type="text" value="<none>"/>	<input type="checkbox"/> Edit
<input type="text"/>	<input type="button" value="Create list..."/>	<input type="text" value="<none>"/>	<input type="checkbox"/> Edit
<input type="text"/>	<input type="button" value="Create list..."/>	<input type="text" value="<none>"/>	<input type="checkbox"/> Edit
<input type="text"/>	<input type="button" value="Create list..."/>	<input type="text" value="<none>"/>	<input type="checkbox"/> Edit

Figure 5: SQL Query LIMS Setup

A SQL statement is required to retrieve the desired information from the LIMS database before it can ultimately be uploaded to the CDS.

Via the SQL Query LIMS Setup window, you can write the SQL statement to access the LIMS database, as well as define parameters that will be configurable at runtime. First, the writing of SQL statements via the SQL Statement Editor is covered. The definition of runtime parameters is described in a subsequent section. Refer to the section "Configuring the Runtime Setup".

Method Name – The name of the current Method is displayed. It is not editable.

Over-ride global connection settings

You have the option to over-ride the SQL Query LIMS connection settings as defined via the Preferences window. When this option is activated, you must define a valid connection via the '**Configure Connection**' button.

Click on the '**Configure Connection**' to open the SQL Query LIMS Configuration Setup window. Any values specified in the SQL Query LIMS Configuration Setup window will be stored within the specified Method, and are not applicable to the entire LimsLink EI configuration.

Editing the SQL Statement

Clicking on the '**Edit SQL Statement**' button opens the SQL Statement Editor window. Here you can write an SQL statement to access data from the database.

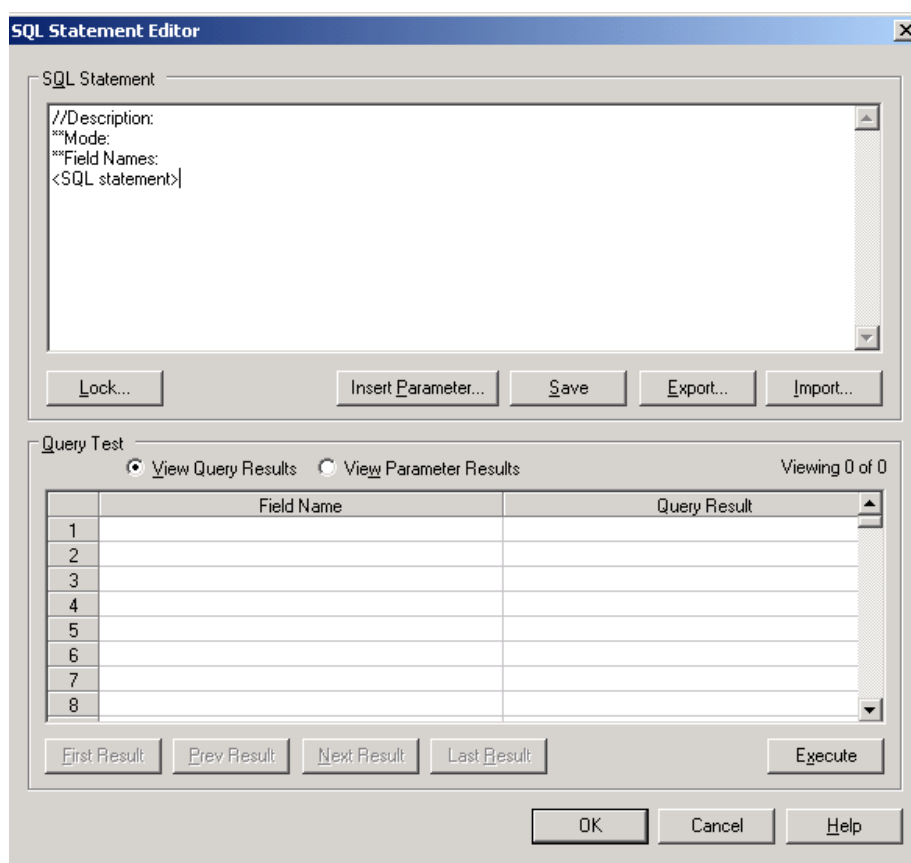


Figure 6: SQL Statement Editor

All SQL Statements in the SQL Statement Editor contain four distinct parts. All statements will be formatted as follows:

Description

Mode statement
Field Names statement
SQL Statement

Description

You may add a comment (description) into the SQL text by starting the line with two forward slashes (i.e., //). This feature can be used to add comments or to comment out lines to facilitate the testing of the entered statement.

Mode statement

The Mode statement is formatted as follows: ****Mode:** <mode>, where <mode> is one of 'Parameter', 'List', or 'Recursive'. The specified mode determines how the SQL statement will be run and how the associated results will be incorporated into the retrieved sample set.

If the 'Parameter' mode is used, none of the data retrieved is placed in the sample repository; it is simply used to further define parameters that will be used in subsequent queries. A 'Parameter' mode query should return only a single record. If more than one record is returned, only the first one will be used. The names specified in the ****Field Names** statement will be available as parameter names in subsequent SQL statements.

If the 'List' mode is used, the data retrieved are used to define the sample list. Retrieved data are placed in unique fields as determined by the ****Field Names** statement. Retrieved 'List' mode fields are used when cross-referencing CDS fields in the sample list.

If multiple 'List' mode queries are used, the data retrieved will be appended to the result of the previous 'List' mode query. If, for example, the first 'List' query returns 10 records and the second 'List' query returns 5 records, the final result will be 15 records. If the queries specify different field names, all field names will be present in each record returned; if a field was not used when retrieving a particular record, the field will be blank for that record.

If the 'Recursive' mode is used, the SQL statement is executed once for each record retrieved by the preceding 'List' mode SQL statement. Field names from the preceding 'List' mode statement can be used as parameters in a 'Recursive' mode query. A 'Recursive' mode query should return only a single record. If more than one record is returned, only the first one will be used. Retrieved data are appended to the current record in fields specified by the ****Field Name** statement.

Field Names statement

The Field Names statement is formatted as follows: ****Field Names:** <name1>, <name2>, <name3>, etc, where <nameX> is the field name to be used to reference the retrieved data. The Field name is also used when cross-referencing CDS fields. A field name must be provided for each result field retrieved by the statement. The order of the fields retrieved corresponds to the order of the Field Names statement.

For example, if the Field Names statement is ****Field Names: Field1, Field2**, the SQL statement is **'SELECT FieldA, FieldB FROM Table1'**, data from FieldA is placed in Field1 and data from FieldB is placed in Field2.

SQL statement

The SQL statement is just that, an SQL statement. The statement may span several lines to format the statement for clarity.

There can be one or more SQL statements each having its own ****Field Names** and ****Mode** criteria.

Statements are executed in order from top to bottom.

Comments (beginning with '//') may be inserted between the statements. This feature can be used to add comments or to comment out lines to facilitate the testing of the entered statement.

Inserting Parameters into the SQL Statement

Parameters may be inserted into the SQL statement. Parameters may be added to the statement at the cursor position by selecting the **'Insert Parameter'** button and then choosing a parameter, or by manually entering '~<Parameter Name>~', where <Parameter Name> is the name of a parameter. Values returned by previous SQL statements may also be used by entering '~<Field Name>~', where <Field Name> is the name of a field from a previous **Field Names statement. The first value of the field being referenced is used in this case.

If a parameter has been configured with the 'Allow other values at runtime' option enabled, it will be substituted at runtime with a comma-delimited list of the selected values. In this case, the parameter should be used in a SQL statement with the 'IN' operator and enclosed in parentheses. For example, a parameter called 'Param1' has the values 123, 456, and 789 at runtime. That parameter could be used in a SQL query as follows: 'SELECT FieldA, FieldB FROM Table1 WHERE FieldC IN (~Param1~)'. At runtime, the SQL query would become 'SELECT FieldA, FieldB FROM Table1 WHERE FieldC IN (123, 456, 789)'. If the parameter is enclosed in single quotes, each value in the comma-delimited list will be enclosed with single quotes at runtime.

Locking the SQL Statement

Clicking on the **'Lock'** button will lock the SQL Statement from further editing. Once selected, you will be presented with a message stating that 'Once the SQL text is locked, this password will be required to make any further changes'. The password to unlock the SQL statement will be requested at this time.

Exporting the SQL Statement to a text file

Clicking on the **'Export'** button will export the SQL statement to a text file. You will be prompted to select a folder and file name in which to store the SQL statement. By default, the file name is '<Method Name>.txt'.

Importing a text file into the SQL Statement

Clicking on the **'Import'** button will import a text file into the SQL statement. You will be prompted to select a folder and file name to retrieve the SQL statement from. Before importing the text file, you will be warned that the importing process will overwrite the existing SQL statement.

Testing the SQL Statement

Clicking on the **'Execute'** button sends the query to the database and then retrieves and displays the data. This tool allows you to test the query without actually running the Method. In the event that the query is quite large and complicated, this is a convenient way to test it as it is being built.

Viewing the Retrieved Data

After testing a query statement, the retrieved data is displayed in the Query Test grid of the SQL

Statement Editor. The Field Name (corresponding to the fields listed in the **Field Names statement) and Query Result (the actual retrieved value) are displayed in the grid. Results for different records can be seen by using the 'First Result', 'Prev Result', 'Next Result', and 'Last Result' buttons. If the View Query Results radio button is selected, the data displayed are those from the execution of 'List' mode and 'Recursive' mode queries. These data are available to be cross-referenced with CDS fields. If the View Parameter Results radio button is selected, the data displayed are those retrieved using 'Parameter' mode queries. These data are not available to be cross-referenced with CDS fields.

Configuring the Runtime Setup

This section allows you to define up to eight different parameters that will be configurable at runtime. The purpose here is to allow the Method to handle parameters entered by the user when the Method is executed. In some cases, a Method may be identical with the exception of certain query parameters contained within the SQL sent to the LIMS; this option removes the requirement to create a separate Method for each parameter combination. The parameters, defined here, will be resolved by the user at runtime with the selected value substituted for the associated parameter reference in the SQL statement.

Parameter Name – specify the parameter name that is to be used at runtime. Up to eight parameters may be individually defined.

Create List – this button allows you configure the values that will be available for selection at runtime. See the next section for details.

Default Selection– here you can select the default runtime entry for the associated parameter. The drop down list box contains user-defined entries, as specified through the List Setup window. A default selection of 'None' will force the user to make a selection at runtime, as there will not be a default selection available. The '**Edit**' check box must be checked in this case.

Edit – this check box is used to indicate that the user can selected the value of the parameter at runtime.

List Setup

Click on the '**Create list**' button to open the List Setup window, similar to the example shown here.

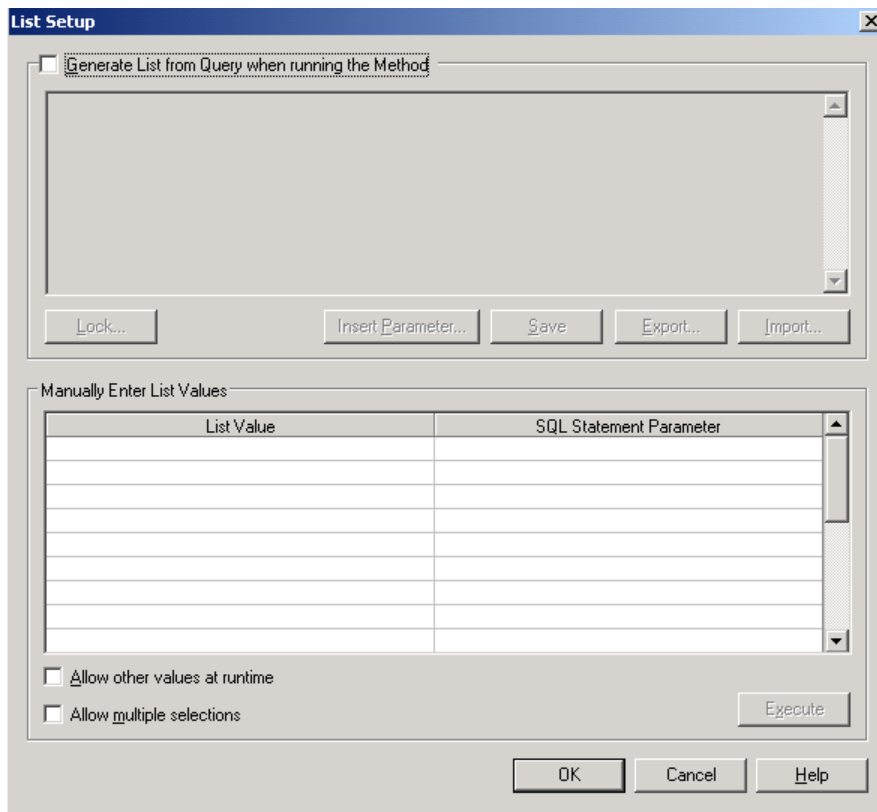


Figure 7: List Setup

The List Setup window allows you to establish a list of values that may be selected by the user for an associated parameter at runtime.

There are two different circumstances whereby list values can be entered.

Entering the List Values Manually

The first situation allows you to manually enter the list value and corresponding SQL statement parameter under the Manually Enter List Values section of the window. In some cases, the parameter value required by the SQL statement to retrieve the information from the LIMS database will be meaningless to the user when it comes time to make a selection at runtime. To eliminate any uncertainty two values can be entered, a list value (i.e., what the user can select from at runtime) in the first column, and a corresponding entry in the second column for what will be substituted into the SQL statement.

At least one list value must be manually entered, or the 'Allow other values at runtime' checkbox must be enabled. See below.

Allow other values at runtime – if this checkbox is enabled, you will be allowed to enter custom list values at runtime.

Allow multiple selections – if this checkbox is enabled, multiple selections will be permitted for the parameter.

Generating the List from a Query at Runtime

The second situation arises when the contents of the list are dynamic and will not be resolved until runtime, or when the contents of the list are dependent on a previous list selection. In this case, you have the option to generate the list at runtime from a query to the LIMS database.

To do this, activate the 'Generate List from Query when running the Method' checkbox.

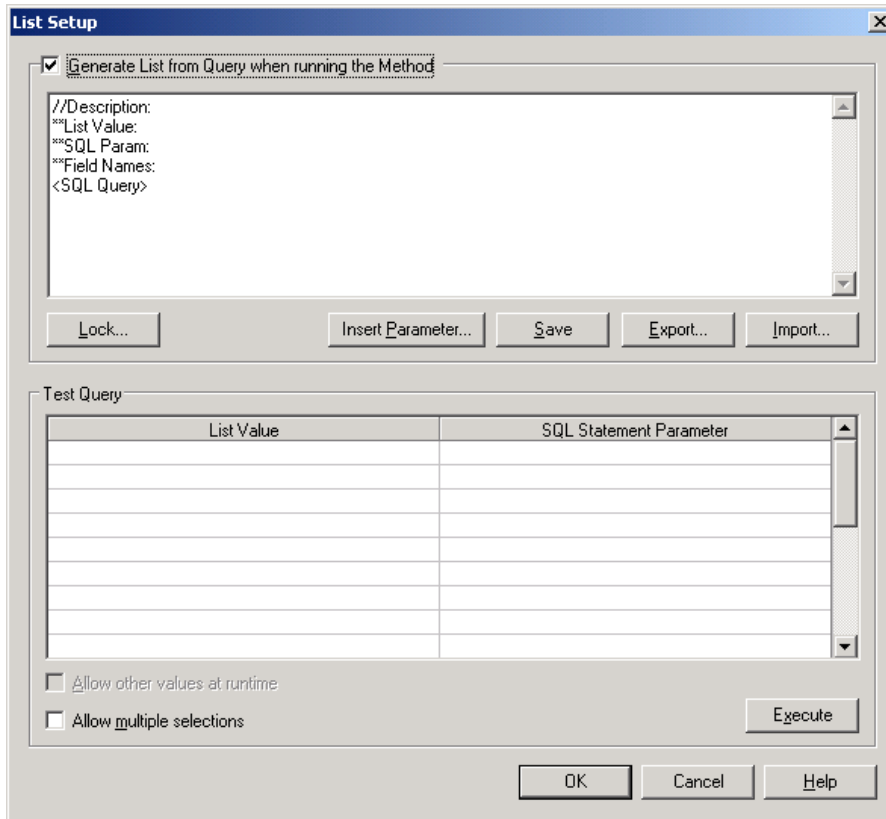


Figure 8: List Setup with Generate List from Query when running the Method option selected

The query is comprised of a SQL statement for which one of the fields returned is identified as the content for the selection list and another is identified for substitution into a subsequent SQL query. Only one SQL statement is permitted.

The default SQL statement is:

```
//Description:
**List Value:
**SQL Param:
**Field Names:
<SQL Query>
```

You may add a comment (description) into the SQL text by starting the line with two forward slashes (i.e., //). This feature can be used to add comments or to comment out lines to facilitate the testing of the entered statement.

The SQL query is preceded by a '**List Value' and a '**SQL Param' statement. The '**List Value'

statement is followed by the field name (defined by the '**Field Names' statement) that represents the data that are presented to the user at runtime as the contents of the list.. The '**SQL Param' statement is followed by the field name (defined by the '**Field Names' statement) that represents the data that are to be inserted into subsequent SQL statements.

You may concatenate several fields together to form the contents of the List Value list. The concatenation is represented by the entry of a simple equation using the '+' symbol to concatenate the contents of referenced fields. You also concatenate static text by enclosing the desired text in quotes. For example, the statement *FieldA+":"FieldB* where the content of Field A is TestMoisture, and FieldB is 012345 the resulting entry into the List Value list would be 'TestMoisture:012345'.

The SQL statement is preceded by a single '**Field Names:' statement. The Field Names statement is formatted as follows: **Field Names: <name1>, <name2>, <name3>, etc, where <nameX> is the field name to be used to reference the retrieved data. A field name must be provided for each result field retrieved by the statement. The order of the fields retrieved corresponds to the order of the Field Names statement.

For example, if the Field Names statement is '**Field Names: Field1, Field2', the SQL statement is 'SELECT FieldA, FieldB FROM Table1', data from FieldA are placed in Field1 and data from FieldB are placed in Field2.

Inserting Parameters into the SQL Statement

A parameter name, which will be resolved at runtime, may be inserted into the SQL statement

To do this, you must enter ~<Parameter Name>~ where <Parameter Name> is the name given to the parameter in the SQL Query LIMS Setup window. Alternatively, a parameter can be inserted into the SQL Statement by clicking on the '**Insert Parameter**' button. This will display a list box from which you can select a parameter name to be inserted at the current cursor position in the SQL statement. Upon selection, ~<Parameter Name>~ will be inserted into the SQL statement at the current cursor position, where <Parameter Name> represents the parameter name selected from the list box.

If a parameter has been configured with the 'Allow other values at runtime' option enabled, it will be substituted at runtime with a comma-delimited list of the selected values. In this case, the parameter should be used in a SQL statement with the 'IN' operator and enclosed in parentheses. For example, a parameter called 'Param1' has the values 123, 456, and 789 at runtime. That parameter could be used in a SQL query as follows: 'SELECT FieldA, FieldB FROM Table1 WHERE FieldC IN (~Param1~)'. At runtime, the SQL query would become 'SELECT FieldA, FieldB FROM Table1 WHERE FieldC IN (123, 456, 789)'. If the parameter is enclosed in single quotes, each value in the comma-delimited list will be enclosed with single quotes at runtime.

Note: In order for a parameter to be valid, it must have been previously defined. For example, you cannot specify a parameter name that is resolved after the current parameter in the setup.

Locking the SQL Statement

Clicking on the '**Lock**' button will lock the SQL Statement from further editing. Once selected, you will be presented with a message stating that once locked, the Create List option for the associated parameter name will require the entry of a password before any further changes can be made. The password to unlock the SQL statement will be requested at this time.

Exporting the SQL Statement to a text file

Clicking on the '**Export**' button will export the SQL statement to a text file. You will be prompted to select a folder and file name in which to store the SQL statement. By default, the file name is '<Method

Name>_<Parameter Name>.txt’.

Importing a text file into the SQL Statement

Clicking on the **‘Import’** button will import a text file into the SQL statement. You will be prompted to select a folder and file name to retrieve the SQL statement from. Before importing the text file, you will be warned that the importing process will overwrite the existing SQL statement.

Allow multiple selections

If this checkbox is enabled, multiple selections will be permitted for the parameter.

Testing the SQL Statement

Clicking on the **‘Execute’** button sends the query to the database and then retrieves and displays the data. This tool allows you to test the query beforehand, without actually running the Method. In the event that the query is quite large and complicated, this is a convenient way to test it as it is being built.

Viewing the Retrieved Data

After executing a query at setup time, the retrieved data is displayed in the Query Test grid of the List Setup. The List Value (corresponding to the **List Value statement) and SQL Statement Parameter (corresponding to the **SQL Param) are displayed in the grid.

Runtime Behavior

At runtime, if one or more SQL statement parameters exist and are editable, a Select Sample Parameters window will appear similar to the example shown here.

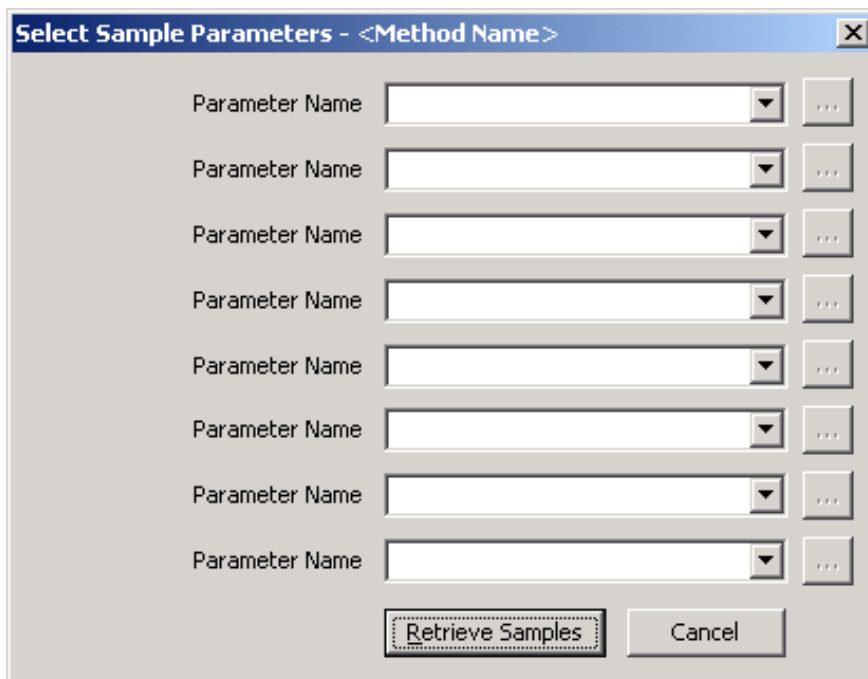


Figure 9: Select Sample Parameters

This will allow the user to select the values for each parameter. The values presented correspond to the List Value list that was set up for the particular parameter. Some parameters may be defined in such a way that their values are not selectable until other parameters have had their values selected. You are expected to fill out the parameter list in a top down format. If a parameter has the 'Allow other values at runtime' option enabled, you will be able to enter a value rather than selecting one from the drop-down list.

When the 'Allow multiple selections' option is enabled for a parameter via the List Setup window, a Select Sample Parameter window, similar to the example shown here, will be available at runtime using the '...' button.

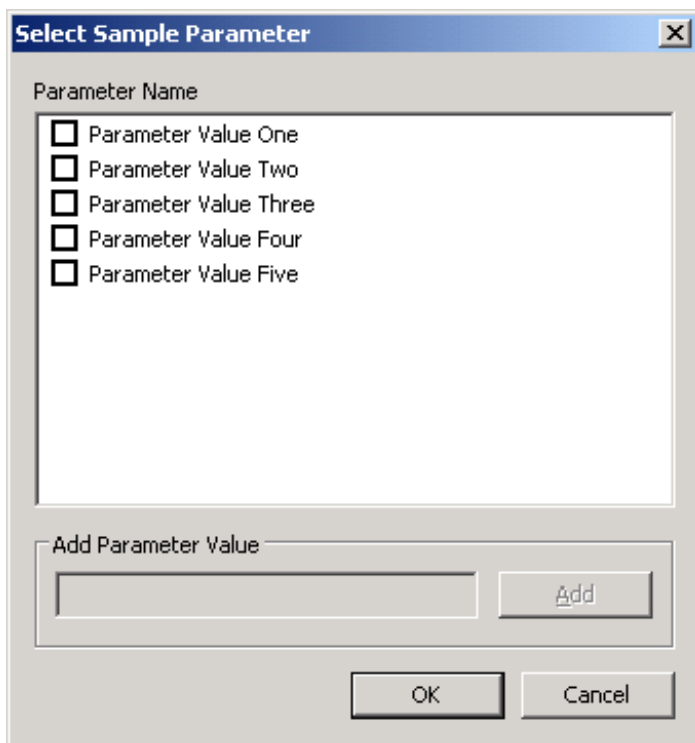


Figure 10: Select Sample Parameter

From here, you can specify the runtime values for the selected parameter.

If the 'Allow other values at runtime' checkbox was enabled via the List Setup window, the Add Parameter Value group box will be enabled. Entering a Parameter value, and clicking on the '**Add**' button will append the parameter value to the current list of parameter values.

Programming Reference

CDSLINKSQLLIMSLib Library Objects

LIMSSqlLims Object

Description:

This is the generic SQL Query LIMS object.

Parameters Property

Syntax:

object.Parameters

object

Data Type: LIMSSqlLims object

Any object variable returning a LIMSSqlLims object.

Description:

A read-only property that returns the internal LIMSParams object.

SQLText Property

Syntax:

object.SQLText

object

Data Type: LIMSSqlLims object

Any object variable returning a LIMSSqlLims object.

Description:

A read/write property that sets or returns the SQL statement text that is used to retrieve samples

from the LIMS database.

ExecuteSQL Function

Syntax:

object.ExecuteSQL(sqltext)

object

Data Type: LIMSSqlLims object

Any object variable returning a LIMSSqlLims object.

sqltext

Data Type: String

The SQL statement text.

Return Value:

A Variant containing an array of strings that are the results returned by an SQL query. Each string consists of a set of comma delimited values, one per returned field. A script error is thrown if the SQL statement text is empty or invalid.

Description:

Executes an arbitrary SQL statement.

SetSQLMacroValue Function

Syntax:

object.SetSQLMacroValue(name, value)

object

Data Type: LIMSSqlLims object

Any object variable returning a LIMSSqlLims object.

name

Data Type: String

The name of a parameter contained in the SQL statement text.

value

Data Type: String

The parameter replacement value.

Return Value:

A long integer value that is the number of instances of the parameter name that were replaced.

Description:

Replaces all instances of a parameter name in the SQL statement text with the specified value. A script error is thrown if the macro name is empty.

ILIMSSqlLims2 Interface

Description:

This is an additional interface on the generic SQL Query LIMS object.

GetOverrideLIMSConfig Function*Syntax:*

```
interface.GetOverrideLIMSConfig
```

interface

Data Type: ILIMSSqlLims2 object

Any object variable returning an ILIMSSqlLims2 interface.

Return Value:

An instance of the ILIMSConfig interface to the LIMSConfig object.

Description:

Gets an ILIMSConfig interface to the LIMSConfig object with connection settings that are used instead of the global LIMS connection settings for this method if the over-ride global connection settings option is turned on.

PutOverrideLIMSConfig Function*Syntax:*

```
interface.PutOverrideLIMSConfig( configObj )
```

interface

Data Type: ILIMSSqlLims2 object

Any object variable returning an ILIMSSqlLims2 interface.

configObj

Data Type: ILIMSConfig interface.

Description:

Sets the LIMSConfig object with connection settings that are used instead of the global LIMS connection settings for this method if the over-ride global connection settings option is turned on.

