Technical Document

Niagara Lexicon Guide

July 23, 2022

niagara4

Niagara Lexicon Guide

Tridium, Inc.
3951 Westerre Parkway, Suite 350
Richmond, Virginia 23233
U.S.A.

Confidentiality

The information contained in this document is confidential information of Tridium, Inc., a Delaware corporation ("Tridium"). Such information and the software described herein, is furnished under a license agreement and may be used only in accordance with that agreement.

The information contained in this document is provided solely for use by Tridium employees, licensees, and system owners; and, except as permitted under the below copyright notice, is not to be released to, or reproduced for, anyone else.

While every effort has been made to assure the accuracy of this document, Tridium is not responsible for damages of any kind, including without limitation consequential damages, arising from the application of the information contained herein. Information and specifications published here are current as of the date of this publication and are subject to change without notice. The latest product specifications can be found by contacting our corporate headquarters, Richmond, Virginia.

Trademark notice

BACnet and ASHRAE are registered trademarks of American Society of Heating, Refrigerating and Air-Conditioning Engineers. Microsoft, Excel, Internet Explorer, Windows, Windows Vista, Windows Server, and SQL Server are registered trademarks of Microsoft Corporation. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Mozilla and Firefox are trademarks of the Mozilla Foundation. Echelon, LON, LonMark, LonTalk, and LonWorks are registered trademarks of Echelon Corporation. Tridium, JACE, Niagara Framework, and Sedona Framework are registered trademarks, and Workbench are trademarks of Tridium Inc. All other product names and services mentioned in this publication that are known to be trademarks, registered trademarks, or service marks are the property of their respective owners.

Copyright and patent notice

This document may be copied by parties who are authorized to distribute Tridium products in connection with distribution of those products, subject to the contracts that authorize such distribution. It may not otherwise, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior written consent from Tridium, Inc.

The product(s) described herein may be covered by one or more U.S. or foreign patents of Tridium.

Contents

About this Guide 5

Document change log 5

Related documents 5

Chapter 1 About the Lexicon Tool 7

Lexicon overview 7

About nested lexicon values 8

Chapter 2 Common tasks 9

About English (en) lexicon usage 9

Editing the default English (en) lexicon 9

Adding key-value pairs to a lexicon module 9

Editing an existing key-value pair in a lexicon module 10

Creating of a new lexicon module for distribution 11

Migrating a lexicon module 12

Adding a pdf.encoding property to a lexicon module 12

Enabling the Diagnostic Lexicon feature 14

Chapter 3 Lexicon Tool Component Summaries 17

Lexicon Tool 17

Lexicon Display Options 17

Search Lexicon 18

Diagnostic Lexicon feature 19

Chapter 4 Lexicon Tool Plugin (view) Summaries 21

Lexicon Report view 21

Lexicon Editor view 22

About Adding Key/Value Pairs 23

Lexicon Module Builder view 23

Lexicon Module Migrator view 25

Index 27

About this Guide

This topic contains important information about the purpose, content, context, and intended audience for this document.

Product Documentation

This document is part of the Niagara technical documentation library. Released versions of Niagara software include a complete collection of technical information that is provided in both online help and PDF format. The information in this document is written primarily for Systems Integrators. To make the most of the information in this book, readers should have some training or previous experience with Niagara software, as well as experience working with JACE network controllers.

Document Content

This document provides information on the **Lexicon Tool** that is available in Workbench releases AX-3.7 and later. Included are basic procedures for working with lexicons, as well as the reference material which describes components and views.

Document change log

This topic provides the date this document was released and lists any subsequent document updates that have occurred.

July 23, 2022

Corrected typographical error that can cause diagnostic feature failure.

July 10, 2019

Added a procedure on using the Diagnostic Lexicon tool.

November 9, 2016

Edited the following topic: “Changes to lexicons in Niagara 4,“ and added a procedure for “Adding a pdf. encoding property to a lexicon module for a target locale.”

August 23, 2015

Initial release for Niagara 4

Related documents

Following, is a list of documents that have content that is related to the topics in this guide.

- *Getting Started with Niagara*
- *Niagara Platform Guide*

Chapter 1 About the Lexicon Tool

Topics covered in this chapter

♦ Lexicon overview
♦ About nested lexicon values

This document explains the **Lexicon Tool** and its usage.

For general information on lexicons, see Lexicon overview, page 7. For usage details see the chapter, "Common tasks".

For details on installing (edited) file-based lexicons in remote platforms, see the "Lexicon Installer" section in the *Niagara Platform Guide*. For details on installing module-based lexicons in remote platforms, see the "Software Manager" section in the *Niagara Platform Guide*.

Lexicon overview

Standard lexicons are distributed as *modules* (`niagaraLexiconXx.jar`) that are included in the Workbench installation. Each lexicon module contains one or more lexicon files (`moduleName.lexicon`). Lexicon files are simple text files that map various entity "keys" (such as interface names or error and status values) to localized language character values. Often, mapped characters use special encoding.

Niagara provides *non-English* language support with the use of *lexicons*. Lexicons are identified by Java *locale codes*, such as "fr" (French) or "de" (German). When installing Niagara on your PC, the full complement of lexicon modules are installed in the !modules folder.

NOTE: In some cases, lexicon editing provides utility even in English language scenarios. For more details, see About English (en) lexicon usage, page 9.

For example, the following 3 lines are a "snippet" from the `alarm.lexicon` file in a French (fr) lexicon set:

```
commands.reset.label=Remettent \u00E0 l'\u00E9tat initial
messages.connectionError=N'a pas pu se relier \u00E0\:  {0}
portal.wizard.title=Ajoutez La Console D'Alarme
```

NOTE: The factory supplied language lexicons typically require review and editing before they can be used on an actual job. Text file based customizations of the standard lexicons are saved under a "`!lexicon`" folder in your installation directory.

To edit lexicons, you typically use the Workbench **Lexicon Tool** (**Tools→ Lexicon Tool**). The **Lexicon Editor** view not only provides edit features, but also shows the default (English) value for each line entry. Any changes you make to a lexicon using the **Lexicon Editor** are saved as lexicon *files*, under the appropriate ! lexicon subfolder for the target locale. For example, changes to the French lexicon are saved in the !lexicon/ fr subfolder.

NOTE: The `!lexicon` folder does not exist in a new installation. The folder is created the first time you edit a lexicon and save it.

The lexicon tool also provides a **Lexicon Report** view, useful to see various details about a lexicon and its contained files. Additionally, the **Lexicon Report** view lists all lexicons installed on your Workbench PC. The **Lexicon Module Builder** view, facilitates collecting multiple lexicons in a module for ease of distribution.

The **Lexicon Module Migrator,** allows you to migrate AX-3.8 lexicon modules to a more recent installation.

Once reviewed and edited, you typically install any needed lexicon into all controller platforms. For more details, see the Lexicon Installer and Software Manager sections in the *Niagara Platform Guide*.

About nested lexicon values

Nested lexicon values are cases where, the value of a lexicon key can refer another key in either the same module's lexicon file, or a different module's lexicon file. Multiple levels of nesting are supported, as needed. This feature may be most used by Niagara developers who are creating new modules that need lexicons.

The syntax for nested lexicon values is as follows:

`{lexicon:moduleName:key}` (if referencing a key in a different lexicon file), or

`{lexicon:key}` (if referencing a key in the same lexicon file)

For a multi-level example within the same (module) lexicon file, consider these `key=value` pairs in the "test" module's lexicon:

```
fum.text={lexicon:foe.text} Fum
foe.text={lexicon:fie.text} Foe
fie.text={lexicon:fee.text] Fie
fee.txt=Fee
```

In the example above, the value for `fum.text` resolves to "Fee Fie Foe Fum".

This could be seen in a Px label widget using BFormat scripting syntax: `%lexicon(test.fum.text)%`

Chapter 2 Common tasks

Topics covered in this chapter

♦ About English (en) lexicon usage
♦ Adding key-value pairs to a lexicon module
♦ Editing an existing key-value pair in a lexicon module
♦ Creating of a new lexicon module for distribution
♦ Migrating a lexicon module
♦ Adding a pdf.encoding property to a lexicon module
♦ Enabling the Diagnostic Lexicon feature

This section provides information about the English language lexicon usage and instructions on how to use the **`Lexicon Tool`** to perform common tasks.

About English (en) lexicon usage

An important use of lexicons is for non-English language support. However, you can also modify the English (en) lexicon to provide a "global default" change for display of items seen in Workbench, or served by a running station to client web browsers. This could be a possible alternative to working from the slot sheet for multiple points to make the same type of interface or "display name" change, or from making similar changes using the **Batch Editor** view of the **ProgramService**.

The English language serves as the default Workbench lexicon. As such, it is not included as a separate lexicon "en" distribution subfolder with module.lexicon files. However using the Lexicon Editor view, you can still edit and add lexicon keys in different modules for English usage. These act as default overrides, saved as module.lexicon files in a `!lexicon/en` subfolder. After restarting Workbench, such changes are used when opening any station (including one on your Workbench PC, if applicable). Typically, you also install this modified `en` lexicon into remote host platforms, such that stations running on them serve up the same changed descriptions to browser clients.

Editing the default English (en) lexicon

This example describes how to add the English (en) lexicon to the list of currently installed lexicons in the **Lexicon Report** view.

Step 1 In Workbench, select **Tools→Lexicon Tool**.

The **Lexicon Report** view displays all lexicons currently installed.

Step 2 If an `en` lexicon is not listed in the left pane, click the **New Lexicon** button, and in the **New Lexicon** window, enter `en` in the **`Name`** property and click **OK**.

An `en` entry is added to the list of lexicons.

Step 3 Select the `en` lexicon, which populates the right pane with the different modules in which you can change or add lexicon keys.

Adding key-value pairs to a lexicon module

This procedure shows how to edit an existing lexicon to add key-value pairs. In this example, you edit the `control` module in the English (en) lexicon to add two key-value pairs in order to change the default right-click descriptors for the actions, "active" and "inactive". By default, there are no existing keys shown for the actions, `active` and `inactive`, so they must be added.

Step 1 In the **Lexicon Report** view, select the "`en`" lexicon in the left pane, which populates the right pane with the different modules in which you can change or add lexicon keys.

NOTE:

If the English (en) lexicon is not listed, you can add it by following the procedure for "Editing the default English (en) lexicon" on page 1-8, page 9.

Step 2 Double-click the `control` module, to load a list of existing key-value pairs in the **Lexicon Editor**.

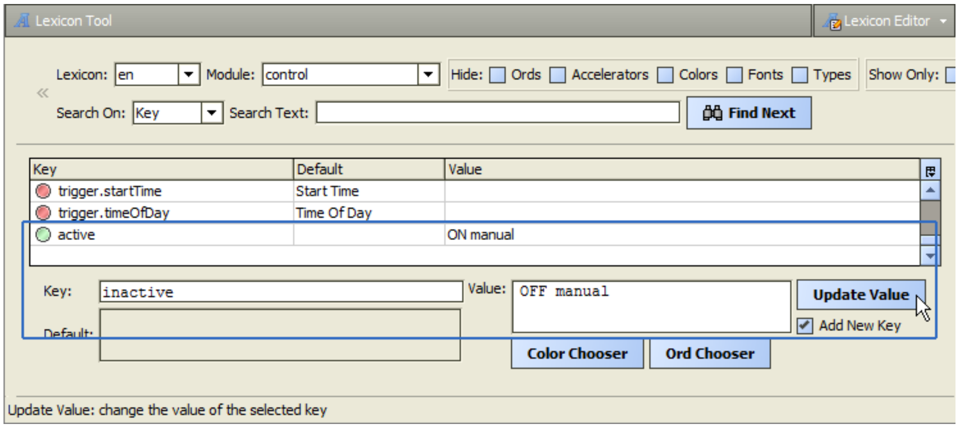
Step 3 Select the "**Add New Key**" check box at the bottom, far right in the view, which enables the **`Key`** field.

Step 4 Enter the new key-value pair (as shown here) and then click the **Update Value** button.

- **`Key:`** `active`
- **`Value:`** `ON manual`

Step 5 Repeat the previous step to add the other key-value pair (as shown here) and click the **Update Value** button.

- **`Key:`** `inactive`
- **`Value:`** `OFF manual`



Lexicon: en Module: control Hide: □ Ords □ Accelerators □ Colors □ Fonts □ Types Show Only: □

Search On: Key Search Text:

Key	Default	Value
trigger.startTime	Start Time	
trigger.timeOfDay	Time Of Day	
active		ON manual

Key: inactive Value: OFF manual

☑ Add New Key

Default:

Update Value: change the value of the selected key

Step 6 Deselect the **Add New Key** check box after adding this second key-value pair.

Step 7 Click the **Save** 💾 button on the toolbar.

This creates (or updates) the `control.lexicon` file located in your `!lexicon/en` folder

Step 8 Restart Workbench, open a station and right-click a writable point to issue actions.

The new default values (`ON manual` and `OFF manual`) replace the old defaults (`Active` and `Inactive`).

NOTE:

If working on a Supervisor (hosting a station), note that following a station restart, such lexicon changes become effective defaults to web browser clients connecting to the station.

Note also that any "display name" changes for slots on a component *override* any lexicon values, including any modified in this manner. For example, if a **BooleanWritable** has the **Display Name** of its "`active`" action slot edited to "`Start Pump`", that command text will always appear when issuing this action, whether from client browser access or from Workbench access.

Editing an existing key-value pair in a lexicon module

This procedure shows how to change the `alarm` module **Priority** key value from "Priority" to "Importance".

Step 1 In the **Lexicon Report** view, select the "`en`" lexicon in the left pane, which populates the right pane with the different modules in which you can change or add lexicon keys.

NOTE:

If the English (en) lexicon is not listed, you can add it by following the procedure for "Editing the default English (en) lexicon" on page 1-8, page 9.

Step 2 Double-click the `alarm` module, to load a list of existing key-value pairs in the **Lexicon Editor** view and scroll (or Search) to find the **`Priority`** key.

Step 3 In the **`Value`** field enter: `Importance` and click the **Update Value** button to replace the previous value.

Step 4 Click the **Save** 💾 button on the toolbar.

This creates (or updates) the `alarm.lexicon` file under your `!lexicon/en` folder.

Step 5 Restart Workbench, open a station and right-click a writable point to issue actions. The new default value, `Importance`, replaces the old default value, `Priority`.

NOTE:

If working on a Supervisor (hosting a station), note that following a station restart, such lexicon changes become effective defaults to web browser clients connecting to the station.

Note also that any "display name" changes for slots on a component *override* any lexicon values, including any modified in this manner. For example, if a BooleanWritable has the Display Name of its "`active`" action slot edited to "`Start Pump`", that command text will always appear when issuing this action, whether from client browser access or from Workbench access.

Creating of a new lexicon module for distribution

This procedure shows how to bundle two edited lexicon files (alarm, control) in a module for easy distribution.

Prerequisites:

NOTE: This method can be used to bundle legacy lexicon files for distribution or to compile custom lexicon files for use in a installation.

To create the new lexicon module:

Step 1 From the Tools menu, select **Lexicon Tool**.

The **Lexicon Report** view displays all lexicons currently installed.

Step 2 Click the **View** dropdown menu and select **Lexicon Module Builder**. The **Lexicon Module Builder view** displays populating the selection table with any subfolders and files currently located under your `!lexicon` folder.

Step 3 Define the new module information by filling in the text fields as shown here:

- **`Module Name`**: `abcLex`
- **`Preferred Symbol`**: `abc`
- **`Module Description`**: `Modified En alarm and control`
- **`Vendor Name`**: `abc Company`
- **`Version`**: `1.1`

Step 4 Click the check box to select the following Source files in the "`En`" lexicon:

- `alarm.lexicon`
- `control.lexicon`

Step 5 Click the check box to **Delete lexicon files after build** to remove these source files from the `!lexicon/en` folder.

Step 6 Click the **Build Module** button.

This creates the new `abcLex.jar` module under your `!modules` folder.

Step 7 Restart Workbench.

Navigate to the `!modules` folder and scroll to confirm the new module was created. You can install this new lexicon module on remote controller platforms (using platform **`Software Manager`**), such that stations running on them serve up the same changed descriptions to browser clients.

NOTE:

If working on a Supervisor (hosting a station), note that following a station restart, such lexicon changes become effective defaults to web browser clients connecting to the station.

Note also that any "display name" changes for slots on a component *override* any lexicon values, including any modified in this manner. For example, if a **BooleanWritable** has the **Display Name** of its "`active`" action slot edited to "`Start Pump`", that command text will always appear when issuing this action, whether from client browser access or from Workbench access.

Migrating a lexicon module

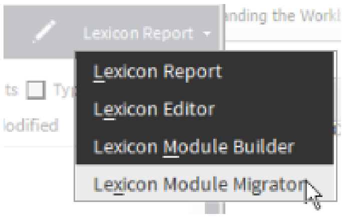
Use the **Lexicon Module Migrator** view to migrate AX-3.8 lexicon modules to the Niagara 4.0 installation.

Prerequisites:

- One or more lexicon modules from an AX-3.8 installation located anywhere in the file system.

Step 1 In Workbench, select **Tools→Lexicon Tool**.

Step 2 In the **Lexicon Report** view, click on the view dropdown, as shown, and click **Lexicon Module Migrator**.



Step 3 In the **Lexicon Module Migrator** view, if desired, click Change Destination... and in the **Choose Destination** dialog change the location as needed and click **OK**.

Step 4 Click Select Lexicon Modules... and in the **Module Chooser** popup, select the module(s) to be migrated and click **OK**.

Step 5 Click Migrate Modules to migrate the selected modules.

A confirmation popup lets you know the module(s) successfully migrated and is available at the indicated destination.

Adding a pdf.encoding property to a lexicon module

A pdf.encoding property enables fonts and characters for Eastern European and Russian language locales, and a few Western European characters. The encoding property is entered in a `pdf.lexicon` text file for the target locale. This procedure describes the steps to add the pdf.encoding property to a new lexicon module.

Prerequisites:

- A !lexicon subfolder in your Niagara installlation. If not already present in the installation you can create the folder.

Step 1 Within the !lexicon folder, create a subfolder for the appropriate target language locale.

As an example, for the Polish locale (indicated by the “pl” locale code) you would create: `!lexicon\pl`.

Step 2 Create a text file that contains the following four lines of text:

```
pdf.encoding=encodingValue
pdf.font=12pt ArialMT
pdf.table.header.font=bold 10pt Arial
pdf.table.cell.font=10pt Arial
```

Step 3 In the text file, replace the pdf.encoding value (shown in Step 2) with the appropriate value for your target language.

For example, for the Polish target language use: `CP1250`.

The following pdf.encoding values are available for use:

Target language locale	PDF.encoding value
Default Latin alphabet	`ISO-8859-1`
Central and Eastern European languages	`CP1250`
Cyrillic (Russian)	`CP1251`
Extended Latin alphabet (includes Euro currency glyph)	`CP1252`
Japanese	`UniJIS-UCS2-H`
Korean	`UniKS-UCS2-H`
Chinese	`UniGB-UCS2-H`
Chinese (Taiwan)	`UniCNS-UCS2-H`

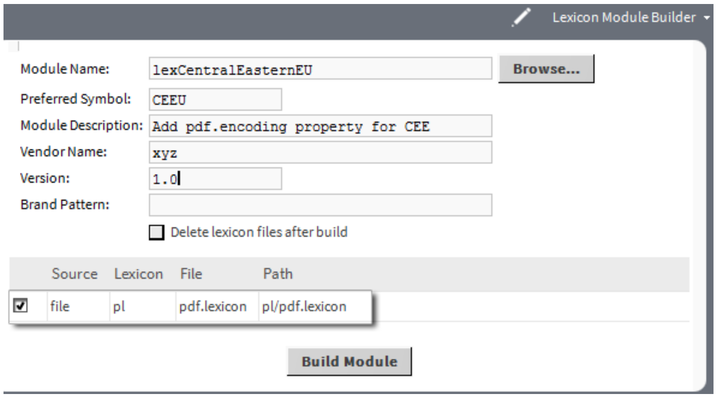
Step 4 Save the text file to the `!lexicon<locale>` subfolder created in Step 1, naming the text file: “`pdf.lexicon`”.

Step 5 In Workbench, click on **Tools→Lexicon Tool**.

Step 6 In the **Lexicon Report** view, click the **View** dropdown menu and select **Lexicon Module Builder**.

The view displays populating the selection table in the view with any subfolders and files currently located under your `!lexicon <locale>` subfolder.

Step 7 In the **Lexicon Module Builder** view, define the new module information by filling-in the text fields and clicking the checkbox to include the `pdf.lexicon` file (as shown), and click **Build Module**.

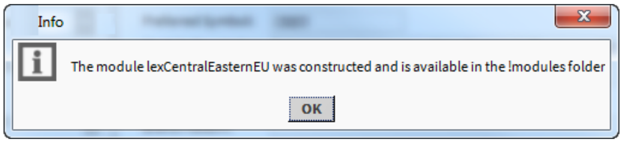


Module Name: lexCentralEasternEU
Preferred Symbol: CEEU
Module Description: Add pdf.encoding property for CEE
Vendor Name: xyz
Version: 1.0
Brand Pattern:
□ Delete lexicon files after build

	Source	Lexicon	File	Path
☑	file	pl	pdf.lexicon	pl/pdf.lexicon

NOTE: If your text file is not visible in the selection table, click **Refresh** in the Workbench toolbar to refresh the view. Also, for more details on using the Lexicon Module Builder, see the Niagara Lexicon Guide.

On successful completion a confirmation dialog explains that the new module is available in the !modules folder, similar to the one shown here.



The module lexCentralEasternEU was constructed and is available in the !modules folder

NOTE: It is possible to select an existing lexicon module via the Lexicon Module Builder's **Browse...** button, add a pdf.lexicon text file to the selected module, and build a new module. However, the default lexicon modules for the installation (`niagaraLexicon<xx>.jar` files in !modules) cannot be used in a new module.

Enabling the Diagnostic Lexicon feature

This section describes how to make the Diagnostic Lexicon feature active. When enabled and configured, this can help to ensure that a lexicon is properly translated and it can also reveal the proper lexicon module and lexicon key for lexicon's displayed on the page.

The lexicons for different languages are part of your installation and are present in the `!modules` folder. For example, the German lexicon module name is: `NiagaraLexiconDe-rt.jar`.

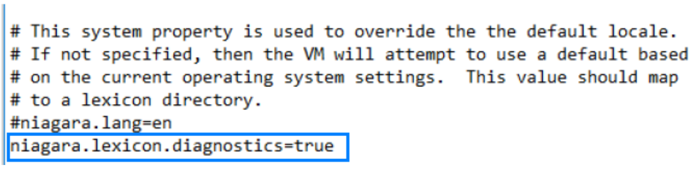
To enable the Diagnostic Lexicon feature, you need to add a system property, as described in the following steps.

Step 1 In the **NavTree**, expand **My Host→My File System→Sys Home**, and open the `defaults` folder.

Step 2 Scroll to locate the `system.properties` file and double-click to open it in the **Text Editor** view.

Step 3 In the **Text Editor**, scroll to locate the section beginning with: "`# This system property is used to override the default locale.`".

Step 4 At the end of this section, type this entry on a separate line (as shown in the blue box): `niagara.lexicon.diagnostics=true`.



```
# This system property is used to override the the default locale.
# If not specified, then the VM will attempt to use a default based
# on the current operating system settings.  This value should map
# to a lexicon directory.
#niagara.lang=en
niagara.lexicon.diagnostics=true
```

Step 5 **Save** this change and restart Workbench.

After restarting, the Diagnostic Lexicon feature is active.

Once the Diagnostic Lexicon feature is active you can configure the feature with different entries to help you find and fix lexicon issues. Examples of lexicon issues and the recommended system property entries are as follows:

- For a simple visual indicator that values come from a lexicon, you can add a single-character prefix such as "~" (example: "~January"). To add this prefix to all lexicon values, add this entry: `niagara.lexicon.diagnostics.prefix=~`
- To identify a lexicon key and module you can use a longer prefix: "`lexiconModule-lexiconLang-lexiconKey=`" (example: "`baja-en-january=January`". To add this longer prefix to all lexicon values, add this entry: `niagara.lexicon.diagnostics.prefix=lexiconModule.lexiconLang.lexiconKey=`
- To add a suffix to all lexicon values using the character, "~", add this entry: `niagara.lexicon.diagnostics.suffix=~`
- To fix missing lexicons for the java lexicon, add this entry: `niagara.lexicon.diagnostics.addMissing=true`

- To modify only missing lexicons for the java lexicon, add this entry:
  `niagara.lexicon.diagnostics.onlyMissing=true`

NOTE: : Following the Workbench restart, the changes in the lexicon are visible.

Chapter 3 Lexicon Tool Component Summaries

Topics covered in this chapter

♦ Lexicon Tool
♦ Lexicon Display Options
♦ Search Lexicon
♦ Diagnostic Lexicon feature

The following sections describe Lexicon Tool components:

Lexicon Tool

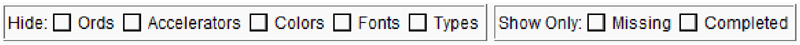
The Lexicon Tool component is available under the Workbench **Tools** menu. The lexicon tool is the container for the lexicon views that enable reviewing lexicon modules and files installed on your Workbench PC, editing the contents of lexicon files within lexicon modules installed on your Workbench PC, building new lexicon modules, and migrating AX-3.8 lexicons to aNiagara 4 installation. For detailed information, see "Lexicon overview".

Lexicon Display Options

Lexicon-specific controls and options are displayed at the top of both the **Lexicon Report** view and the **Lexicon Editor** view. For more details, see the view topics in the chapter, "Views".

Both **Lexicon Report view** and **Lexicon Editor view** use a type of tabular presentation with the standard table features and options, as described in the *Getting Started with Niagara*. Lexicon-specific controls and options are displayed at the top of both the Lexicon Editor and Lexicon Report. These controls and options are shown here and described below:

Figure 1 Lexicon display options


Hide: □ Ords □ Accelerators □ Colors □ Fonts □ Types
Show Only: □ Missing □ Completed

- **Hide Ords**

  If enabled, it filters out any referenced based entries, such as `modules://icons/x16/alarm.png`

- **Hide Accelerators**

  If enabled, it filters out entries that define keyboard accelerators (entries end with ".accelerator").

- **Hide Colors**

  If enabled, it filters out entries that define colors (entries end with ".bg" or ".fg"), such as those defined with hexadecimal code `#FFFF0000`.

- **Hide Fonts**

  If enabled, it filters out entries that define fonts.

- **Hide Types**

  If enabled, it filters out entries that end with ".displayName" or ".icon".

- **Show Only Missing**

  If enabled, this shows only entries without any assigned values—entries will be the "red" orb ones only.

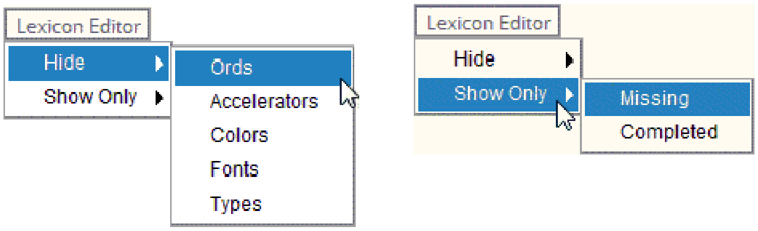
- **Show Only Completed**

If enabled, this shows only entries that do have assigned values—entries will be the "green" orb ones only.

NOTE: The two "Show Only" settings are mutually exclusive.

In addition, the Lexicon display options can be accessed under the **Lexicon Editor** pull-down menu, as shown below. You can clear both of the "Show Only" options, but set only one.

Figure 2 Lexicon Editor menu bar display options



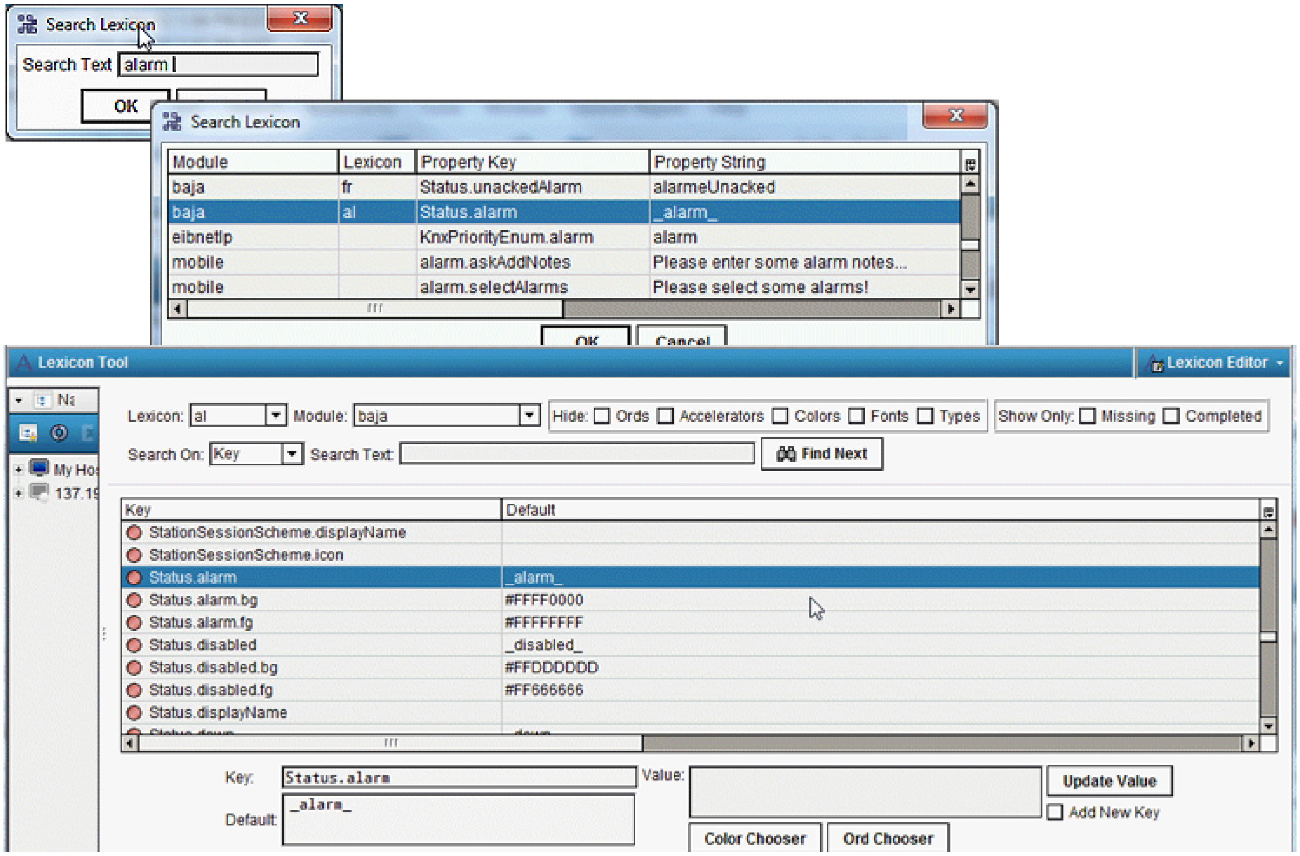
NOTE: The default settings for these display options are set in the **Options** dialog box. However, these controls allow you to override the Workbench defaults.

Search Lexicon

The Search Lexicon feature is available in both the **Lexicon Report** view and **Lexicon Editor** view. The search capability lets you search for specific text on either the lexicon property **`Key`**, the property **`Default`**, or the custom **`Value`**. For detailed information, the view topics in the chapter, "Views".

Rows in the search results table are selectable. If you select a row containing only a Module value and click **OK**, the corresponding row is selected in the lexicon report table. If you select a row that contains Module and Lexicon values and click **OK**, the view will switch to the **Lexicon Editor view**. In the **Lexicon Editor view**, the Lexicon and Module values are set, the properties are loaded for that combination and the row containing the selected key/value pair from the search is selected, as shown here. The selection is also applied to the Lexicon Report, so that if you go back to the Lexicon Report view, the corresponding Lexicon list and Module table entries are selected

Figure 3 Search functionality in Lexicon Tool



Search Text: alarm

Module	Lexicon	Property Key	Property String
baja	fr	Status.unackedAlarm	alarmeUnacked
baja	al	Status.alarm	_alarm_
eibnetIp		KnxPriorityEnum.alarm	alarm
mobile		alarm.askAddNotes	Please enter some alarm notes...
mobile		alarm.selectAlarms	Please select some alarms!

Lexicon: al Module: baja Hide: □ Ords □ Accelerators □ Colors □ Fonts □ Types Show Only: □ Missing □ Completed

Search On: Key Search Text:

Key	Default
StationSessionScheme.displayName	
StationSessionScheme.icon	
Status.alarm	_alarm_
Status.alarm.bg	#FFFF0000
Status.alarm.fg	#FFFFFFFF
Status.disabled	_disabled_
Status.disabled.bg	#FFDDDDDD
Status.disabled.fg	#FF666666
Status.displayName	

Key: Status.alarm Value:

Default: _alarm_

□ Add New Key

Diagnostic Lexicon feature

Information in this section includes introduction of new feature in the Lexicon Tool.

Diagnostic Lexicon is a new feature that provides a number of new localization tools for Niagara 4.8 and later versions. This new feature is useful for UI developers and Lexicon experts to find and fix lexicon issues. With this new feature, translation quality is increased and consistency is maintained. All lexicon entries are when this tool is active. When this feature is enabled and configured, it can help to ensure that the lexicon is properly translated. This also reveals that the proper lexicon module and lexicon key is displayed on the page.

For usage details, see “Enabling the Diagnostic Lexicon feature” in the “Common tasks” chapter.

Chapter 4 Lexicon Tool Plugin (view) Summaries

Topics covered in this chapter

- ♦ Lexicon Report view
- ♦ Lexicon Editor view
- ♦ Lexicon Module Builder view
- ♦ Lexicon Module Migrator view

The following sections describe plugins (views) available in the lexicon tool.

Lexicon Report view

The **Lexicon Report** is the default view displayed when you select the **`Lexicon Tool`**. **Lexicon Report** view lists all module-based and file-based lexicon locales installed on your Workbench PC. For detailed information, see Lexicon Report view.

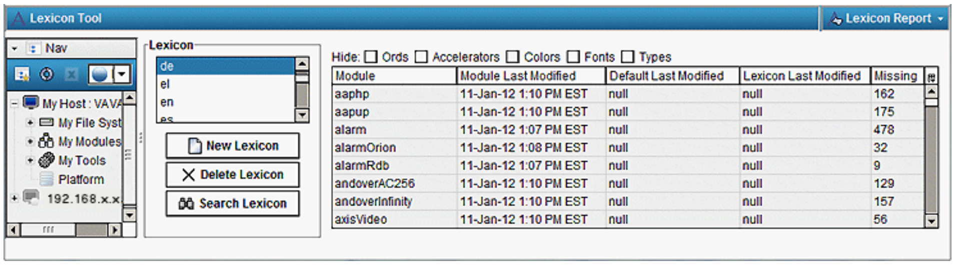
When selecting the **Lexicon Tool** from the **Tools** menu, the default view is the **Lexicon Report**, as shown below. The left side of this view shows a list of all module-based and file-based lexicon locales installed on your Workbench PC. Each lexicon module (one module per language) contains text-based lexicon files (`moduleName.lexicon`).

NOTE: It is possible to have lexicon files in the !lexicon folder, and not have a lexicon module. You can deploy custom lexicons in any file-based and/or module-based combination.

A lexicon module is conventionally named `niagaraLexiconXx` and includes a two-letter language designation, such as “`De`” for German or “`Fr`” for French (`niagaraLexiconFr.jar` is the French lexicon module). As needed, you click a lexicon to select it.

NOTE: To load the default English lexicon in the **Lexicon Report** view, click the **New Lexicon** button and type in “en” in the text field of the **New Lexicon** dialog box. Click the **Ok** button and the lexicon displays in the Lexicon Report view. After this initial display, the “en” name is always listed in the lexicon pane unless you use the **Delete Lexicon** button to remove it.

Figure 4 Lexicon Report view



Hide: □ Ords □ Accelerators □ Colors □ Fonts □ Types

Module	Module Last Modified	Default Last Modified	Lexicon Last Modified	Missing
aaphp	11-Jan-12 1:10 PM EST	null	null	162
aapup	11-Jan-12 1:10 PM EST	null	null	175
alarm	11-Jan-12 1:07 PM EST	null	null	478
alarmOrion	11-Jan-12 1:08 PM EST	null	null	32
alarmRdb	11-Jan-12 1:07 PM EST	null	null	9
andoverAC256	11-Jan-12 1:10 PM EST	null	null	129
andoverInfinity	11-Jan-12 1:10 PM EST	null	null	157
axisVideo	11-Jan-12 1:10 PM EST	null	null	56

Selecting a lexicon in report view shows various status parameters about each lexicon (broken down by module) in the table columns on the right side. These status parameters include the following:

- **`Module`**

  Each module that contains a lexicon file (`moduleName.lexicon`).

- **`Module Last Modified`**

  Timestamp of when the lexicon in the module (JAR file) was last modified.

- **`Default Last Modified`**

Timestamp of when the default lexicon file (`!modules/moduleName.jar/moduleName.lexicon`) for that module was last modified. This field is blank if there is no lexicon file for that module. However, you can still edit that module's lexicon and Save, which creates the lexicon file and updates this timestamp field.

- **`Lexicon Last Modified`**

  Timestamp of when any module-based or file-based lexicon file for the selected lexicon was last modified. This field is blank if there is no lexicon file for that module. However, you can still edit that module's lexicon and Save, which creates the lexicon file and updates this timestamp field.

- **`Missing`**

  Number of keys that are missing values (in the lexicon for that module).

  NOTE: Check boxes at the top of report view allow you to "hide" values that could affect "missing" status counts, as described in "Lexicon display options".

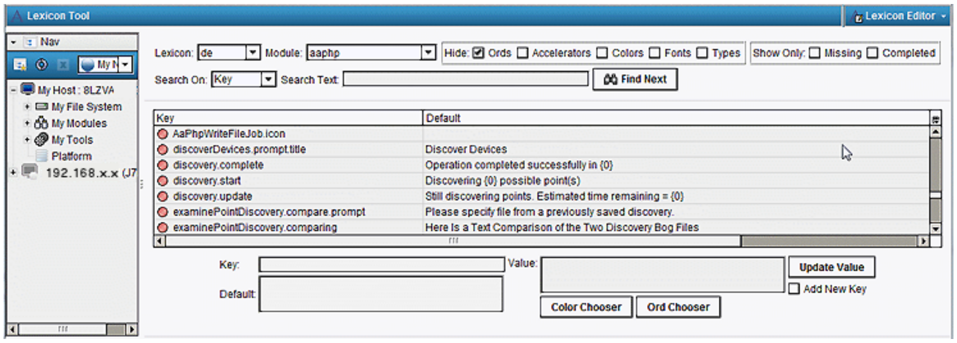
In this view, you can also add (new) lexicons, delete unwanted lexicons, or search all available lexicon property values that contain a given text. Additionally, from this view you can launch the **Lexicon Editor view** by double-clicking any module row, which shows the contents of that lexicon file.

Lexicon Editor view

The **Lexicon Editor** lets you view and edit the contents of lexicon files within the lexicon modules installed on your Workbench PC.

You can then install the edited lexicon files in a remote controller using the platform **Lexicon Installer** view. Also, you can create (or update) a lexicon module with those edited lexicon files, using the new **Lexicon Module Builder** view. You can install the module in a remote controller using the platform **Software Manager** view.

Figure 5 Lexicon editor



Lexicon: de Module: aaphp

Hide: ☑ Ords ☐ Accelerators ☐ Colors ☐ Fonts ☐ Types

Show Only: ☐ Missing ☐ Completed

Search On: Key Search Text:

Key	Default
AaPhpWriteFileJob.icon	
discoverDevices.prompt.title	Discover Devices
discovery.complete	Operation completed successfully in {0}
discovery.start	Discovering {0} possible point(s)
discovery.update	Still discovering points. Estimated time remaining = {0}
examinePointDiscovery.compare.prompt	Please specify file from a previously saved discovery.
examinePointDiscovery.comparing	Here Is a Text Comparison of the Two Discovery Bog Files

Key:

Default:

Value:

☐ Add New Key

Typically, you access the editor from the **Lexicon Report** view with a lexicon selected on the left side, by double-clicking a module in the selection table on the right side

The lexicon editor displays a table listing the defined keys for that module, with columns for mapping to a localized value override for the default value, as follows:

- **`Key`**

  Interface entity, constant in that module from lexicon to lexicon.

- **`Default`**

  Default character value, used if no local value is defined for that key (English equivalent).

- **`Value`**

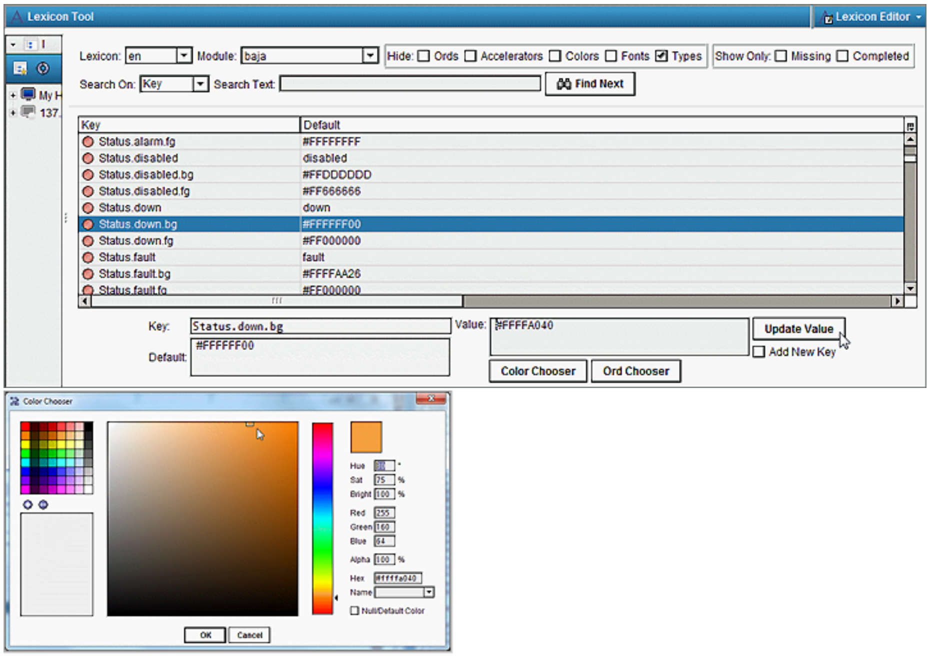
  Localized character value (overrides default value).

> **NOTE:** Check boxes at the top of the view allow you to "hide" values that could affect "missing" status counts, or filter the view to show only those keys "missing" localized character values or only those keys with "completed" localized character values, as described in "Lexicon display options".

At the top of this view, you can search for the specified text on either the lexicon property Key, the property Default, or the custom Value. As needed, you can resort the table columns to find information needed, and simply click any row to select that key. Clicking any row in the table populates the **Key** field.

At the bottom of the **Lexicon Editor** view, the **Add New Key** button enables the Key field. The **Value** text field allows you to modify the value of the selected key or add a value for a new key. The **Value** field has associated **Color Chooser** and **Ord Chooser** buttons. These options allow you to browse for a specific color or ORD element. Update the table with the Value field entry by clicking the **Update Value** button. Write changes to the lexicon file by clicking the **Save** button on the toolbar.

Figure 6 Use color chooser to enter hex code for background color value for Status Alarm down



Lexicon: en Module: baja Hide: □ Ords □ Accelerators □ Colors □ Fonts ☑ Types Show Only: □ Missing □ Completed

Search On: Key Search Text:

Key	Default
Status.alarm.fg	#FFFFFFFF
Status.disabled	disabled
Status.disabled.bg	#FFDDDDDD
Status.disabled.fg	#FF666666
Status.down	down
Status.down.bg	#FFFFFF00
Status.down.fg	#FF000000
Status.fault	fault
Status.fault.bg	#FFFFAA26
Status.fault.fg	#FF000000

Key: Status.down.bg

Default: #FFFFFF00

Value: #FFFFA040

□ Add New Key

Hue [illegible] °
Sat 75 %
Bright 100 %
Red 255
Green 160
Blue 64
Alpha 100 %
Hex #ffffa040
Name
□ Null/Default Color

About Adding Key/Value Pairs

As mentioned above, the **Add New Key** option allows you to add a new key/value pair to a lexicon file. If a particular key is not included in a lexicon file, you can use this option to add the key. For example, if you want to globally replace the "Elapsed Active Time" and "Elapsed Active Time Numeric" property descriptors with "Runtime" and "Runtime Numeric", you must add those keys to the lexicon file. This is sourced in the `control` module. So, you must add the new keys in the `control` module.

Refer to Chapter 2 "Common tasks" on page 1-7, page 9 for more details on adding key-value pairs.

Lexicon Module Builder view

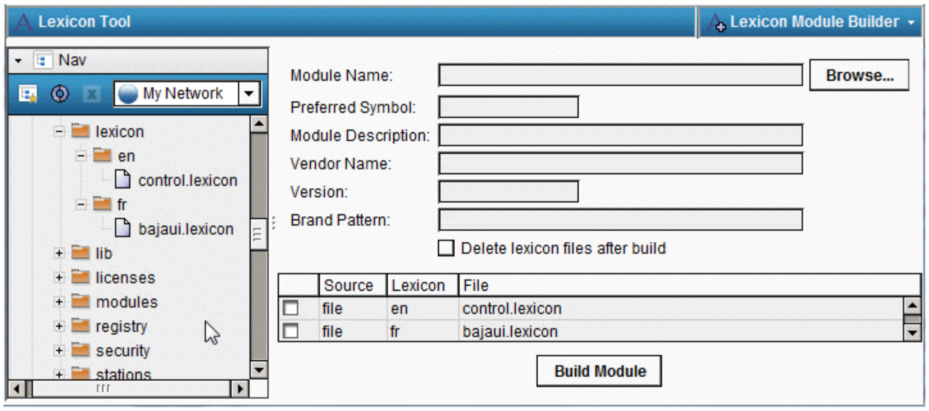
The**Lexicon Module Builder** view allows you to bundle multiple lexicon files into a module for ease of distribution.

The lexicons available for use are those lexicon files (`modulename.lexicon`) located in the appropriate `!lexicon` subfolder for the target locale. For example, `!lexicon/fr` is the subfolder which would contain French lexicon files.

You can build new lexicon modules or replace existing ones.

In this view there are text fields for defining module information, a **Browse** button that allows you to locate existing lexicon modules, a selection table of available lexicon files that can be selected for inclusion, a check box to delete source files after the build is completed, and a **Build Module** button to initiate the build, as shown here.

Figure 7 Lexicon module builder



☐ Delete lexicon files after build

	Source	Lexicon	File
☐	file	en	control.lexicon
☐	file	fr	bajaui.lexicon

The text fields in this view allow you to provide basic information for the module. Several of these fields are required. Leaving a required field empty will result in a failed build. The required fields are:

- **`Module Name`**

  Recommended naming conventions for new modules are: "`customLexicon<Xxxx>`" or "`<Vendor>-Lexicon<Xxxx>`" or "`<Brand>Lexicon<Xxxx>`".

- **`Preferred Symbol`**

  An abbreviation (1 to 8 alphabetic characters) of the module name.

- **`Vendor Name`**

  A text string to describe the vendor.

- **`Version`**

  An integer or integers separated by periods, for example 1 or 1.4.2.

`Brand Pattern`, an optional text field, allows you to specify a "brand" attribute for this module. The Lexicon system refers to this field at runtime, comparing it to the brand for the current system (controller platform or Workbench). If you provide a brand name in this field, the set of lexicon files in your module will be available for loading only if this value matches the runtime brand identifier defined as part of the license.

NOTE: Providing a Brand Pattern value is optional. A lexicon module with no brand pattern will always be available for use.

Located just below the brand pattern text field is a check box option to delete lexicon files after the build is completed. If you select this option, when you click the **Build Module** button you will be prompted to confirm that you want to delete files in the `!lexicon` folder.

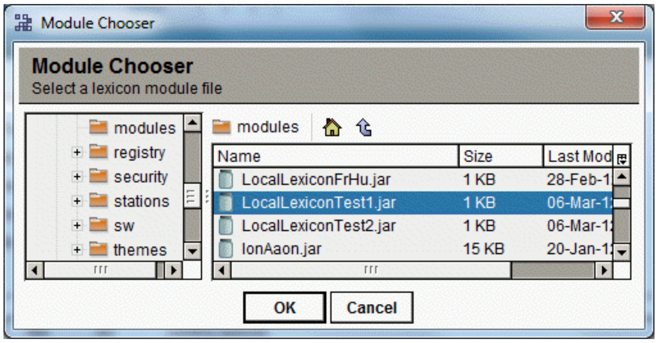
As mentioned earlier, the selection table in the lower half of the view lists the lexicons available for use. Click a check box in the left column of a table row to mark a lexicon file for inclusion in the module. Double-clicking an entry in the selection table will switch to the **Lexicon Editor view** for the selected language-module combination. Switching between the lexicon views will retain your selections (text box values, selected table rows, check box values) unless a module was loaded from the Module Chooser. In that case, the module must be re-loaded in order to re-populate the selection table with the lexicons within that module.

If rebuilding an existing lexicon module, click the **Browse** button to open a Module Chooser dialog box. Once you select a module in the Chooser and click **OK**, it fills in the context of that module in the available text fields in the **Lexicon Module Builder** view and the module's existing lexicon files are added to the selection table, as shown below. Clicking the **Cancel** button in the Chooser closes the dialog box. When you click the Build Module button, you will get separate confirmation dialog boxes to overwrite the existing module, to delete the lexicon files, and indicating the module was rebuilt successfully.

NOTE:

Lexicon modules can be rebuilt multiple times.

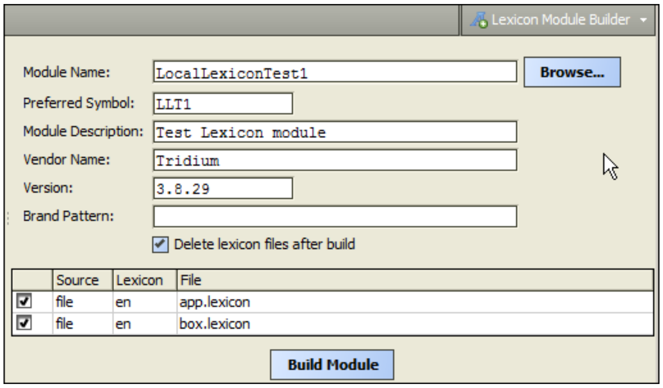
Figure 8 Browse to select existing lexicon modules



Module Chooser

Select a lexicon module file

Name	Size	Last Mod
LocalLexiconFrHu.jar	1 KB	28-Feb-1
LocalLexiconTest1.jar	1 KB	06-Mar-1
LocalLexiconTest2.jar	1 KB	06-Mar-1
IonAaon.jar	15 KB	20-Jan-1



Module Name: LocalLexiconTest1

Preferred Symbol: LLT1

Module Description: Test Lexicon module

Vendor Name: Tridium

Version: 3.8.29

Brand Pattern:

☑ Delete lexicon files after build

	Source	Lexicon	File
☑	file	en	app.lexicon
☑	file	en	box.lexicon

NOTE: A lexicon module edited with the Lexicon Module Builder must adhere to the following restrictions:

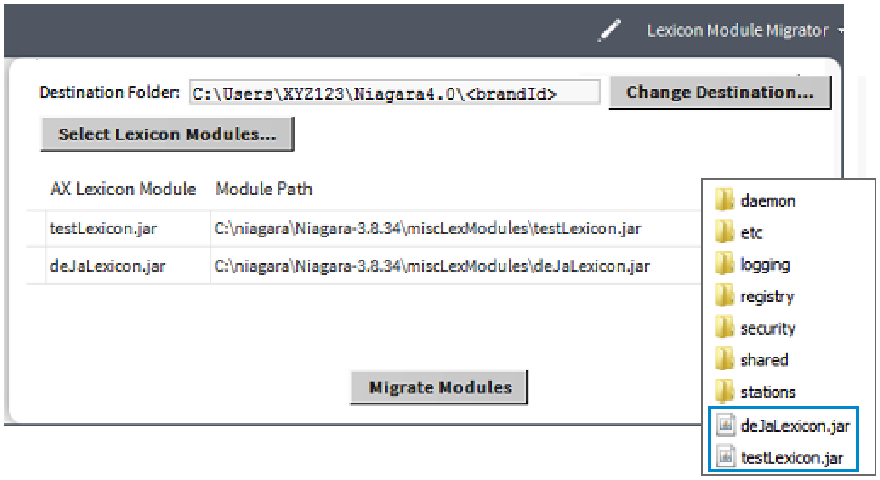
- Module may not contain Java class files (compiled Java code).
- Module may not contain default lexicon modules (`niagaraLexicon<xx>`.jar files in the `!modules` folder).
- Source files selected for inclusion in the module must be uniquely named. For example, selecting a file-based lexicon and a module-based lexicon both of which are named `control.lexicon` will result in an error message.

On completion of the build, you will see a confirmation message indicating that the module was constructed and placed in the `!modules` folder.

Lexicon Module Migrator view

Use the **Lexicon Module Migrator** view to migrate AX-3.8 lexicon modules to aNiagara 4 installation.

Figure 9 Lexicon Module Migrator view and user home folder with migrated lexicon files



Destination Folder: C:\Users\XYZ123\Niagara4.0\<brandId>

AX Lexicon Module	Module Path
testLexicon.jar	C:\niagara\Niagara-3.8.34\miscLexModules\testLexicon.jar
deJaLexicon.jar	C:\niagara\Niagara-3.8.34\miscLexModules\deJaLexicon.jar

You can select modules to migrate from anywhere in the PC file system. By default, the destination for migrated lexicon files is the Workbench user home. However, you can change the destination by selecting an alternate location.

NOTE: The **Lexicon Module Migrator** works for lexicon files present in an AX-3.8 installation that are still valid for Niagara 4. However, for any modules that are new in Niagara 4, such as webEditors, etc., you need to build new lexicon modules using the **Lexicon Module Builder** view.

Index

D

Diagnostic Lexicon 14
document change log 5

F

framework...19

L

Lexicon Module Migrator
    about...25
Lexicon related topics..5

M

migrating lexicon modules
    AX-3.8 to Niagara 4..12

S

Search Lexicon 18