DTP Help-Helper
for
Dynamic Context-Sensitive Help

This document introduces the Data Tools Platform™ (DTP) help-helper plug-in, which is provided in the Eclipse™ DTP project, since version 1.5 (released June 29, 2007).

The DTP help-helper plug-in (org.eclipse.datatools.help) contributes two key enablers for dynamic context-sensitive help:

- A ContextProviderDelegate implementation, whose methods allow the abstraction of help context IDs and context-specific help search expressions from the user interface (UI) code
- An extension point (org.eclipse.datatools.help.helpKeyProperties), which allows other plug-ins to contribute ResourceBundle properties files that define the mapping of abstract helpKey constants (used in the UI code) to externalized help context ID strings and context-specific help search expressions

Note: This document deals specifically with the Eclipse implementation of dynamic context-sensitive help. It does not address issues related to statically-defined help contexts (i.e., non-dynamic context-sensitive help).

Contents

1. Introduction

Context-sensitive help is focused user assistance content, which is specific to the current application context (i.e., the help context), and presented on demand when a platform-specific trigger is activated (e.g., F1 on Windows).

Context-sensitive help requires interactions between user interface (UI) components, user assistance (UA) components, and the Eclipse platform help system.

UI components:

- Contribute UI controls
- Define the association of help contexts with UI controls

UA components:

- Contribute context-specific help content and online documentation content
- Define the association of help contexts with context-specific help content and related online documentation topics

Eclipse help system:

- Resolves help context references to matching context-specific help contributions
- Contributes the help presentation mechanisms (e.g., the Help view)
- Provides API to enable context-sensitive help interactions for UI and UA components

1.1. Static vs. Dynamic Help Contexts

UI components can define the associations between UI controls and help contexts, either statically or dynamically.

- When defined statically, the effective help context changes only when the user explicitly requests context-sensitive help (i.e., when the context-sensitive help trigger is activated).
- When defined dynamically, the effective help context changes whenever the user activates any UI control with a dynamic context association (e.g., selecting a menu option, moving from one page to another in a dialog, etc.).

That distinction is significant when the Help view is visible.

- When the context association is defined statically, context-specific content shown in the Help view remains unchanged, regardless of the user's action, unless the user explicitly requests context-sensitive help (e.g., by pressing F1).
- When the context association is defined dynamically, context-specific content shown in the Help view is updated automatically, whenever the user activates a UI control.

All help contexts are defined and subsequently identified by a help context ID. The Eclipse help system uses help context IDs to locate matching org.eclipse.help.IContext objects, which represent the context-specific help content.

1.2. Dynamic Context-Sensitive Help

Since 3.1, Eclipse supports dynamic context-sensitive help with:

- An interface class (org.eclipse.help.IContextProvider), and
- The Help view (a workbench part)

To provide dynamic context-sensitive help, a UI component must define the associations between its UI controls and help contexts dynamically, by implementing methods of IContextProvider:

- getContext(Object target) — returns a help context (ID) for the given target
- getContextChangeMask() — returns the mask created by combining supported change triggers, using the bitwise OR operation
- getSearchExpression(Object target) — returns a search expression that the help system should use to find more help content related to the current target

The Help view tracks the activation of workbench parts (views and editors), and it checks to see if they adapt to the IContextProvider interface. If they do, the Help view uses the IContextProvider methods (implemented in the UI component) to locate a corresponding IContext object.

The IContextProvider.getContextChangeMask method provides a mask that the Help view uses to determine when a context change occurs dynamically (e.g., as a result of a user gesture to change the UI focus).

When the Help view detects a context change, it updates its context-sensitive help presentation to display:

- The content of the IContext object that is associated with the current context (i.e., the name and description of the UI control, and a list of related help topics)
- Dynamic Help search results — a list of help topics found using the search expression that was returned by the IContextProvider.getSearchExpression method

The content of an IContext object (and its association with a particular help context ID) is defined by a context XML file, which is usually contributed by a UA component.

To register IContext content contributions with the help system, a UA plug-in must declare an extension of org.eclipse.help.contexts, identifying both the context XML file and the UI plug-in for which the IContext content is contributed.

1.3. Help Context Abstraction

Help context abstraction is a technique to simplify the handling of help context IDs and help search expressions in the UI code, by abstracting them to “help keys.”

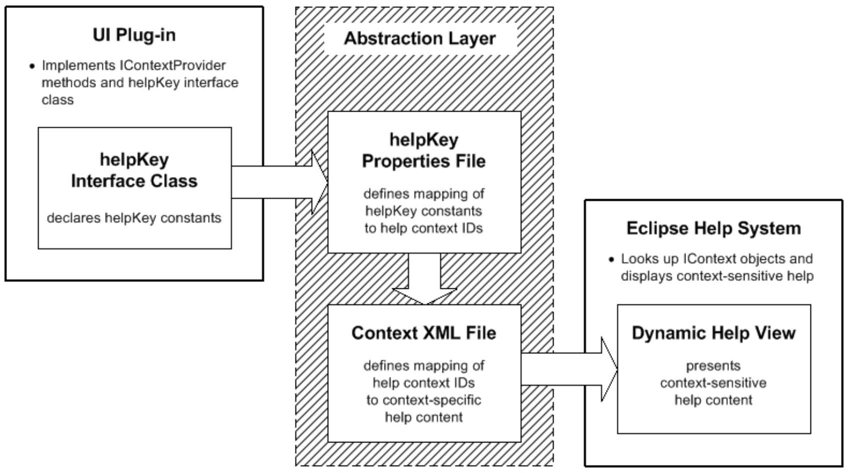
Help context abstraction provides the following benefits:

- Development teams are free to associate new help contexts with UI controls, without necessitating that corresponding help context IDs or help search expressions exist.
- Documentation teams are free to define UA help contexts, modify help context IDs, and control the mapping from abstract help contexts to concrete help context IDs, without necessitating any change in the UI code.
- Documentation teams are free to define and modify context-specific help search expressions, and the mapping from abstract help contexts to help search expressions, without necessitating any change in the UI code.

Note: Help search expressions include human-readable text, so they must be externalized for localization.

This separation of responsibilities enables the project team to provide higher quality, and more precisely targeted, dynamic context-sensitive help.

Together, the DTP help-helper and a context-sensitive help UA plug-in provide an abstraction layer, which enables dynamic context-sensitive help, with a practical and easily maintained implementation in the UI code.


UI Plug-in
• Implements IContextProvider methods and helpKey interface class
helpKey Interface Class
declares helpKey constants
Abstraction Layer
helpKey Properties File
defines mapping of helpKey constants to help context IDs
Context XML File
defines mapping of help context IDs to context-specific help content
Eclipse Help System
• Looks up IContext objects and displays context-sensitive help
Dynamic Help View
presents context-sensitive help content

2. DTP Help-Helper

The DTP help-helper plug-in (org.eclipse.datatools.help) provides a "help key" extension point (**org.eclipse.datatools.help.helpKeyProperties**), and supplies a context provider delegate implementation (**org.eclipse.datatools.help.ContextProviderDelegate**).

The helpKeyProperties extension point allows any plug-in to contribute ResourceBundle properties files that define the mapping of abstract help keys to concrete help context IDs and help search expressions.

The ContextProviderDelegate, along with abstract help keys, enables help context abstraction for any UI control that implements methods of IContextProvider.

2.1. Help Keys

Rather than declaring help context IDs directly, UI controls use help keys (i.e., helpKey constants). Though meaningless to the Eclipse help system, helpKey constants provide an abstraction in the Java code from the actual help context IDs and help search expressions.

Each helpKey constant is used to reference an actual help context ID string and a corresponding help search expression, which are defined in properties files (one each for help context IDs and help search expressions).

Note: Any plug-in can contribute the helpKey properties files for a UI component by declaring an extension to org.eclipse.datatools.help.helpKeyProperties in its plugin.xml file. For example, a dedicated context-sensitive help plug-in could contribute both the helpKey properties files and the Eclipse context XML files, independent of the UI plug-in and the UA content contributor (online documentation) plug-in.

Similar to externalized messages, helpKey constants are declared as *public static final String* in an interface class.

UI components must implement a helpKey constants interface class to reference the externalized help context IDs in a ResourceBundle properties file. For more information, see helpKey Constants Interface Class.

2.2. Context Provider Delegate

The ContextProviderDelegate allows a UI control to provide the actual help context ID and help search expression to the methods of IContextProvider, while handling references to those strings as an abstract help key (helpKey constant).

To support the abstraction from help context ID to helpKey constant, a UI control must:

- Adapt to the IContextProvider.class object as a key
- Create an instance of the ContextProviderDelegate, passing in the symbolic name of the plug-in associated with the help context

- Implement static methods of IContextProvider (getContext, getContextChangeMask, and getSearchExpression), each of which returns a corresponding method of the ContextProviderDelegate instance
- Define the help context using IWorkbench.setHelp (e.g., in the createPartControl method), by passing the abstract helpKey constant string

The ContextProviderDelegate methods return the actual help context ID, change mask, or help search expression to the IContextProvider methods.

Note: The DTP help-helper handles unqualified help context IDs defined in properties files to fully-qualify them, before it returns them through ContextProviderDelegate methods.

3. Context-Sensitive Help UI Implementation

The following sections give examples of dynamic help context implementation, with help context abstraction, in a view and a dialog.

Java code examples are from the Eclipse Data Tools Platform (DTP) 1.6 source code.

3.1. View Example

The DataSourceExplorerView (in org.eclipse.datatools.connectivity.ui.dse) first implements the org.eclipse.help.IContextProvider interface, and then it creates an instance of the org.eclipse.datatools.help.ContextProviderDelegate:

```
private ContextProviderDelegate contextProviderDelegate = new
ContextProviderDelegate(DSEPlugin.getDefault().getBundle().getSymbolicName())
;
```

The symbolic name is usually the plug-in ID. Providing the symbolic name as shown above will avoid broken code, if the plug-in’s ID changes.

Note: The plug-in ID passed to the ContextProviderDelegate must indicate which plug-in the help context is actually associated with. In the example above, the help context is associated with the DSEPlugin (i.e., the plug-in that contributes the DataSourceExplorerView UI component). However, the help context could be associated with a separate plug-in, for example, if the plug-in “owner” of the UI component is not the same as the plug-in that contributes the UI component.

Implementation of the IContextProvider methods looks like this:

```
public IContext getContext(Object target) {
   return contextProviderDelegate.getContext(target);
}

public int getContextChangeMask() {
   return contextProviderDelegate.getContextChangeMask();
}

public String getSearchExpression(Object target) {
   return contextProviderDelegate.getSearchExpression(target);
}
```

After implementing the IContextProvider methods, the view must set the help context for the control in the createPartControl method:

```
public void createPartControl(Composite parent) {
   super.createPartControl(parent);
   PlatformUI.getWorkbench().getHelpSystem()
      .setHelp(getCommonViewer().getTree(),
      IHelpContextsConnectivityUIDSE.CONTEXT_ID_CONNECTIVITY_DSE_VIEW);
}
```

Note: The string passed to the setHelp method is actually an abstract helpKey constant, whose property key is mapped to a concrete help context ID string by the ContextProviderDelegate.

All helpKey constant strings must be declared in an interface class. For more information, see helpKey Constants Interface Class.

3.2. Dialog Example

Dialogs must set the help context in the createDialogArea method (to provide context-sensitive help for the dialog itself).

EditDriverDialog (in org.eclipse.datatools.connectivity.internal.ui.dialogs) implements the createDialogArea method like this:

```
protected Control createDialogArea(Composite parent) {
    getShell().setData( HelpUtil.CONTEXT_PROVIDER_KEY, this);
    HelpUtil.setHelp( getShell(),
     HelpUtil.getContextId(IHelpConstants.CONTEXT_ID_EDIT_DRIVER_DIALOG,
     ConnectivityUIPlugin.getDefault().getBundle().getSymbolicName()));
    ...
}
```

This is similar to the technique in the DataSourceExplorerView, except that EditDriverDialog uses methods of the org.eclipse.datatools.help.HelpUtil class to associate an instance of the context provider delegate with the dialog.

- The first call, *getShell().setData()* tells the help system that the dialog will use the help-helper's delegated setHelp method to provide help contexts.
- The second call, *HelpUtil.setHelp()* is a method that wraps the IWorkbench.setHelp call in a unique listener, so that the help system can find the correct help context for this control (in this case, the shell for the dialog).

To provide more granular help, at a lower level in the dialog, the dialog could call setHelp, with a unique help context for each control in the dialog.

Note: The string passed to the setHelp method is actually an abstract helpKey constant, whose property key is mapped to a concrete help context ID string by the ContextProviderDelegate.

All helpKey constant strings must be declared in an interface class. For more information, see helpKey Constants Interface Class.

3.3. helpKey Constants Interface Class

To reference the externalized help context IDs, which are defined in a ResourceBundle properties file, UI components must implement an interface class to declare the abstract helpKey constants.

The following example is from the org.eclipse.datatools.connectivity.ui source code.

```
package org.eclipse.datatools.connectivity.internal.ui;

/**
 * helpKey_constants_for_plug-in: org.eclipse.datatools.connectivity.ui
    */

public interface IHelpConstants {

    /*
     * CONTEXT_ID_CP_PROPERTY_PAGE =
     * Basic profile name/description/auto-connect property page
     */
    public static final String CONTEXT_ID_CP_PROPERTY_PAGE =
            "CONTEXT_ID_CP_PROPERTY_PAGE"; //$NON-NLS-1$

    /*
     * CONTEXT_ID_CP_WIZARD_PAGE =
     * wizard selection page in New Connection Profile wizard
     */
    public static final String CONTEXT_ID_CP_WIZARD_PAGE =
            "CONTEXT_ID_CP_WIZARD_PAGE"; //$NON-NLS-1$

    /*
     * CONTEXT_ID_INTRO_WIZARD_PAGE =
     * Basic intro page for new connection profile wizard
     */
    public static final String CONTEXT_ID_INTRO_WIZARD_PAGE =
            "CONTEXT_ID_INTRO_WIZARD_PAGE"; //$NON-NLS-1$

...
}
```

Notice the following features of the helpKey constants interface class:

- A Javadoc comment block appears between the package and class declarations. The comment identifies every UI plug-in that will use this class to pass an abstract helpKey constant to the ContextProviderDelegate.
- The class declares only helpKey constants; no other constants are declared in this class.
- The class declares all of the helpKey constants for at least one UI plug-in.

Note: The class can declare helpKey constants for more than one UI plug-in, but all of the helpKey constants for any one UI plug-in must be declared in only one helpKey constants interface class.

- Each helpKey constant declaration should be preceded by a comment, which must provide sufficient information for the Documentation team to determine the location and purpose of the associated UI control.
- Each helpKey constant is declared as *public static final String*, in the form:

  ```
  public static final String MY_HELP_KEY = "MY_HELP_KEY";
  ```

  where *MY_HELP_KEY* is the literal character string used as the helpKey constant in a UI control.

Note: The helpKey constants should be declared as literal string values only. Indirect declarations (such as *HELP_KEY = PREFIX + "HELP_KEY_STRING"*) should be avoided.

- The only characters allowed in a helpKey constant string are: upper case and lower case letters (a-z, A-Z), numbers (0-9), and the underscore (_).

Note: A helpKey constant string must not appear in the format of a fully-qualified help context ID. Any period (.) or blank space character will cause problems, if the helpKey constant is mapped to itself (as an unqualified help context ID) in the helpKey properties file.

4. helpKey Properties Files

A helpKey properties file defines the mapping of helpKey constants (property keys), either to actual help context IDs, or to context-specific help search expressions. helpKey properties files are usually contributed in pairs, one for context IDs and one for search expressions.

Note: The help context ID strings defined in a helpKey properties file cannot be fully-qualified context IDs. They must **not** include any blank space or period (.) characters. The DTP help-helper handles unqualified help context IDs defined in properties files to fully-qualify them, before it returns them to the UI control through the ContextProviderDelegate.

The following excerpt is from a helpKey properties file, which is contributed by the org.eclipse.datatools.doc.user.contexts plug-in to define concrete help context ID strings.

```
...
#   CONTEXT_ID_CP_PROPERTY_PAGE =
#   Basic profile name/description/auto-connect property page
CONTEXT_ID_CP_PROPERTY_PAGE = CONNECTION_PROFILE_COMMON_PROPERTY_PAGE

#   CONTEXT_ID_CP_WIZARD_PAGE =
#   wizard selection page in New Connection Profile wizard
CONTEXT_ID_CP_WIZARD_PAGE = NEW_CONNECTION_PROFILE_WIZARD

#   CONTEXT_ID_INTRO_WIZARD_PAGE =
#   Basic intro page for new connection profile wizard
CONTEXT_ID_INTRO_WIZARD_PAGE = NEW_CONNECTION_PROFILE_WIZARD
...
```

Notice that two abstract help contexts (represented by the helpKey constants *CONTEXT_ID_CP_WIZARD_PAGE* and *CONTEXT_ID_INTRO_WIZARD_PAGE*) are mapped to one concrete help context ID (*NEW_CONNECTION_PROFILE_WIZARD*). This is an example of the flexibility afforded by abstract help contexts, allowing the UA component to combine multiple abstract help contexts in a single, concrete help context ID, which is independent of the help context implementation in the UI component.

The following example shows the mapping of a helpKey constant to a context-specific help search expression in a helpKey properties file.

```
...
#    * CONTEXT_ID_CP_PROPERTY_PAGE =
#    * Basic profile name/description/auto-connect property page
CONTEXT_ID_CP_PROPERTY_PAGE = "connection profile" AND "properties"
...
```

Search expression strings must conform to the rules for Apache Lucene query parser syntax: http://lucene.apache.org/java/docs/queryparsersyntax.html

Note: The terms (keywords) in a search expression are not case sensitive, but the Boolean operators (e.g., AND, NOT, OR) must be all upper case.

4.1. helpKey Properties File Names

Names of helpKey properties files are arbitrary (as long as they end with .properties). However, a context-sensitive help plug-in could contribute pairs of helpKey properties files for several UI plug-ins, and the files would need to have unique names, if they were located in the same directory.

As a convention, helpKey properties file names should correspond with the UI plug-in that they serve. For example:

- org.eclipse.datatools.connectivity.ui**.contextIds.properties**
- org.eclipse.datatools.connectivity.ui**.searchExpressions.properties**

To allow arbitrary names for helpKey properties files, and to abstract the names of helpKey properties files from UI components, the DTP help-helper provides an extension point (org.eclipse.datatools.help.helpKeyProperties). That extension point allows any plug-in to declare helpKey properties file contributions for a UI plug-in.

For more information about the org.eclipse.datatools.help.helpKeyProperties extension point, see Contributing helpKey Properties Files.

5. Context-Sensitive Help UA Plug-ins

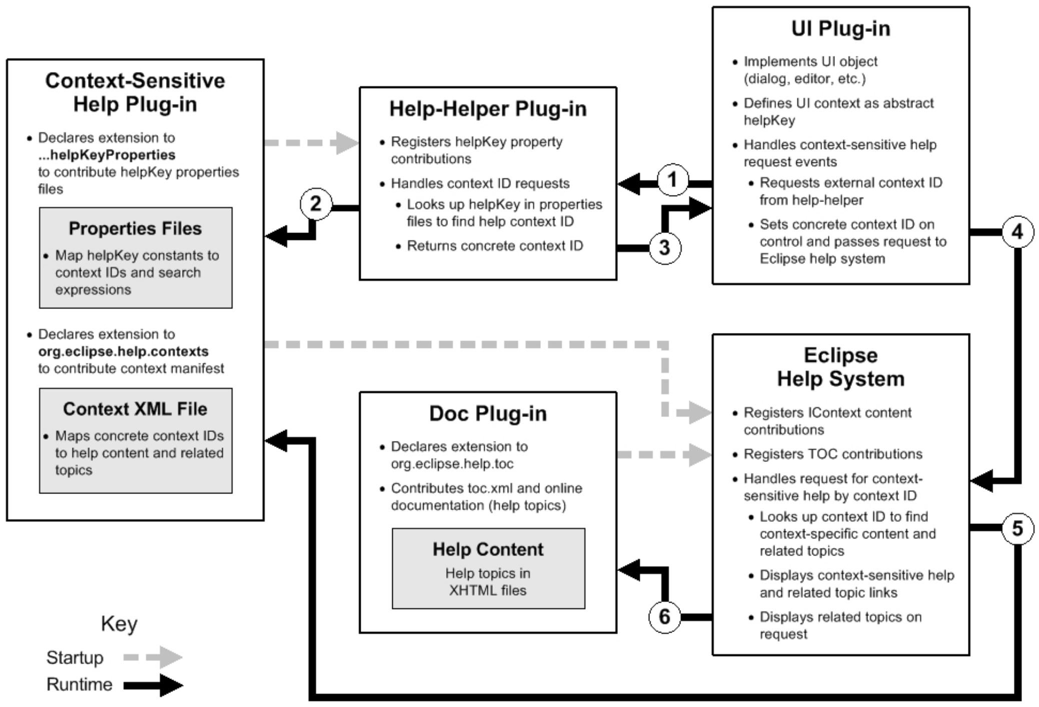
Context-sensitive help UA plug-ins handle the mapping of help contexts to context-specific help content and related online documentation topics.

For projects that use the DTP help-helper, context-sensitive help plug-ins contribute:

- Eclipse context XML files, which supply the context-specific help content for each help context, and point to other help topic contributions related to the help context
- Java properties files, which define the property key-value pairs that map abstract helpKey constants to concrete help context IDs and context-specific help search expressions

Note: The helpKey properties files are usually contributed in pairs, one each for help context IDs and help search expressions.

The following diagram shows how a dedicated context-sensitive help plug-in interacts with the DTP help-helper and the Eclipse help system at runtime, to provide dynamic context-sensitive help for a UI plug-in.


Context-Sensitive Help Plug-in
• Declares extension to ...helpKeyProperties to contribute helpKey properties files
Properties Files
• Map helpKey constants to context IDs and search expressions
• Declares extension to org.eclipse.help.contexts to contribute context manifest
Context XML File
• Maps concrete context IDs to help content and related topics
Help-Helper Plug-in
• Registers helpKey property contributions
• Handles context ID requests
• Looks up helpKey in properties files to find help context ID
• Returns concrete context ID
UI Plug-in
• Implements UI object (dialog, editor, etc.)
• Defines UI context as abstract helpKey
• Handles context-sensitive help request events
• Requests external context ID from help-helper
• Sets concrete context ID on control and passes request to Eclipse help system
Eclipse Help System
• Registers IContext content contributions
• Registers TOC contributions
• Handles request for context-sensitive help by context ID
• Looks up context ID to find context-specific content and related topics
• Displays context-sensitive help and related topic links
• Displays related topics on request
Doc Plug-in
• Declares extension to org.eclipse.help.toc
• Contributes toc.xml and online documentation (help topics)
Help Content
Help topics in XHTML files
1
2
3
4
5
6
Key
Startup
Runtime

During workbench startup:

- The DTP help-helper registers helpKey properties file contributions, declared by context-sensitive help plug-ins
- The Eclipse help system registers:
  - IContext content contributions, declared by context-sensitive help plug-ins
  - Online help (TOC and topics) contributions, declared by documentation plug-ins

After the workbench is initialized, the GUI is presented, with workbench parts and controls contributed by UI plug-ins. Context-sensitive help requests are usually handled by a UI plug-in (or its delegated help event listener), for the workbench parts and UI controls that it contributes.

When the user requests context-sensitive help (e.g., by pressing F1, or clicking a help icon), the following sequence occurs:

1. The UI plug-in requests a concrete help context ID from the DTP help-helper, by passing an abstract helpKey.
2. The DTP help-helper looks up the help context ID by its associated helpKey, in the helpKey properties file that was registered for the UI plug-in.
3. The DTP help-helper returns the help context ID to the UI plug-in.
4. The UI plug-in sets the concrete help context ID on its associated workbench part or UI control, and then passes that help context ID to the Eclipse help system to request the display of context-sensitive help.
5. The Eclipse help system looks up the context-specific content and related topics by help context ID, retrieves them from the context XML file that was registered for the UI plug-in, and then displays them in the Help view.
6. Upon user request (by clicking a related topic link), the Eclipse help system retrieves the help topic content contributed by a documentation plug-in and displays it in the Help view.

5.1. Contributing Eclipse Context XML Files

Each plug-in that contributes an Eclipse context XML file must declare an extension to **org.eclipse.help.contexts** in its plug-in manifest (plugin.xml file). For example:

```
<extension point="org.eclipse.help.contexts">
   <contexts file="dtp_contexts.xml"
      plugin="org.eclipse.datatools.connectivity.ui"/>
</extension>
```

For more information about Eclipse context XML files and contributing them, see:

http://help.eclipse.org/help33/topic/org.eclipse.platform.doc.isv/guide/ua_help_context_xml.htm

5.2. Contributing helpKey Properties Files

Each plug-in that contributes helpKey properties files must declare an extension to **org.eclipse.datatools.help.helpKeyProperties** in its plug-in manifest (plugin.xml file). For example:

```
<extension point="org.eclipse.datatools.help.helpKeyProperties">
  <contextIds
    plugin="org.eclipse.datatools.connectivity.ui"
    file="org.eclipse.datatools.connectivity.ui.contextIds.properties"/>
  <searchExpressions
    plugin="org.eclipse.datatools.connectivity.ui"
    file="org.eclipse.datatools.connectivity.ui.searchExpressions.properties"/>
</extension>
```

The <contextIds> element identifies a helpKey properties file that defines help context IDs, and the UI plug-in for which it is provided.

The <searchExpressions> element identifies a helpKey properties file that defines context-specific help search expressions, and the UI plug-in for which it is provided.

To accommodate context-sensitive help plug-ins that serve more than one UI plug-in, the <extension> element can contain any number of <contextIds> and <searchExpressions> elements.

5.3. Context-Sensitive Help Plug-in IDs

Context-sensitive help plug-ins can be supplied as dedicated plug-ins, which point to related help topics that are contributed by other, online documentation plug-ins, but do not contribute any help topics themselves.

Dedicated context-sensitive help plug-ins can “mirror” the architecture of the online documentation plug-ins. For each plug-in that contributes topics related to help contexts, a corresponding context-sensitive help plug-in can be delivered. In that case, the context-sensitive help plug-in should be named to match its corresponding online documentation plug-in. For example:

• Online documentation plug-in:	org.eclipse.datatools.doc.user
• Context-sensitive help plug-in:	org.eclipse.datatools.doc.user**.contexts**

6. Team Responsibilities and Collaboration

Successful delivery of dynamic context-sensitive help requires a close coordination of UI components and UA components. It imposes responsibilities on both Development teams and Documentation teams, and it requires ongoing collaboration among those teams.

Development teams must provide lists of helpKey constants to Documentation teams, and they must provide information about the UI control associated with each helpKey constant. Development teams must also provide timely updates to Documentation teams when any change occurs in the helpKey constants.

Documentation teams must rely on the information provided by Development teams to locate the UI controls, analyze the help contexts, and create appropriate UA content in various formats.

Since a context-sensitive help plug-in can contribute helpKey properties files and context XML files that support several UI plug-ins, Documentation teams must carefully evaluate the "component level" with which online documentation plug-ins and their corresponding context-sensitive help plug-ins are associated.

Both Development teams and Documentation teams must rely on the project's architectural specifications to determine the appropriate capabilities and features with which to coordinate UI and UA components.

6.1. Development Team Responsibilities

Development teams are primarily responsible to:

- Implement the Eclipse classes and methods necessary to enable dynamic context-sensitive help for all appropriate UI controls.
- Implement the interface classes that declare helpKey constants for each UI plug-in.
- Provide lists of the helpKey constants to Documentation teams in a timely manner.

Development teams should provide lists of the helpKey constants in the form of Java source files for the helpKey constants interface classes.

Note: Java source files provided as helpKey lists must include the help context comments to provide information about each associated UI control.

6.2. Documentation Team Responsibilities

Documentation teams are primarily responsible to:

- Develop context-specific help content and context-specific help search expressions for appropriate UI help contexts.
- Define the concrete help context ID strings that associate abstract help contexts with context-specific help content.

- Create the Java properties files that define the mapping of abstract help contexts to concrete help context IDs.
- Create the context-sensitive help UA plug-ins that contribute Eclipse context XML files and the Java properties files.

Documentation teams should rely on the Java source files (for the helpKey constants interface classes) as the original and definitive sources of all helpKey constant strings.

Creating the Java properties files can be somewhat automated (e.g., by processing the Java source files with a simple perl script).

Most of the other artifacts required for a context-sensitive help UA plug-in can be generated by XSL transformations of a DITA-XML map document. For more information, see Defining Context-Sensitive Help Plug-ins with DITA.

7. Documentation Team Workflow

Documentation teams should refer to the product's architectural specifications to define an appropriate architecture (i.e., granularity, content partitioning, plug-in naming, etc.) for online documentation plug-ins. With that information, Documentation teams can begin the process to create context-sensitive help plug-ins.

Note: The Documentation team workflow described in this section assumes that dedicated context-sensitive help plug-ins will be produced from DITA-XML map documents. The workflow can be modified for other help content source formats.

The following list summarizes the overall Documentation team workflow to create context-sensitive help plug-ins:

1. Get the helpKey list (provided as a Java source file) from the UI Development team for each UI plug-in.
2. Analyze the helpKey list and associated UI controls to define the help contexts. (For more information, see Defining Help Context IDs.)

   The Documentation team must determine whether:

   - The helpKey constants alone are sufficient to identify actual help contexts, and thus, helpKey constants could map directly to a concrete help context ID, with the same string value as the helpKey constant.
   - Distinct help context ID strings must be defined to combine groups of helpKey constants into common help contexts.

Note: It may be preferable to combine help contexts in the helpKey properties file, instead of defining the mapping for multiple help contexts to a single topic in a DITA map. This is a judgment call for the Documentation team lead, and the IAs responsible for maintaining DITA maps. For more information, see Defining Related Topics Associated with Help Contexts.

3. Analyze the help contexts and existing (or planned) help topics to define context-specific help search expressions. (For more information, see Defining Help Search Expressions.)
4. Create helpKey properties files, based on the content of each Java source file.

   - Define the mapping of helpKey constants to concrete help context IDs and context-specific help search expressions, based on results of the help context analysis and help topic (content) analysis.
   - Save the helpKey properties files in source control, as appropriate.

Note: The helpKey properties files are flat ASCII text files, so the authors (or IAs) responsible for defining the help context IDs and context-specific help search expressions should use a suitable ASCII text editor to create and edit those files. For more information about the properties file format, refer to the example in helpKey Properties Files.

5. Modify existing DITA maps (if used to produce online documentation plug-ins) to add the markup for context-sensitive help. For more information, see Defining Context-Sensitive Help Plug-ins with DITA.

7.1. Defining Help Context IDs

Since helpKey constants are abstract entities in the Java code, and independent of the actual help context IDs, Documentation teams are responsible for defining the help contexts, which the Eclipse help system uses to locate context-specific help content, and associating each helpKey constant with an actual help context ID.

Additional considerations:

- Each help context ID represents a single help context, but any one help context could occur more than once in a single UI component (i.e., a UI plug-in). Therefore, any help context ID can be associated with any number of abstract helpKey constants, but
    - Each helpKey constant must be associated with exactly one help context ID.
- Each help context within a UI component should be associated with a unique help context ID, however
    - Help context IDs need not be unique across all plug-ins. For example, a particular help context could be declared by more than one UI plug-in, if the Documentation team determines that the actual help context is the same for multiple UI plug-ins.

7.2. Defining Help Search Expressions

Each help context should be associated with a context-specific help search expression, which the Eclipse help system will use to provide Dynamic Help search results in the Help view.

Documentation teams are responsible for defining the context-specific help search expressions for each help context.

Additional considerations:

- Each help search expression can be associated with multiple help context IDs, but
    - Each helpKey constant must be associated with exactly one help search expression.
- Each help context within a UI component (i.e., a UI plug-in) can be associated with a unique help search expression, however
    - It may be more practical to define a "generic" help search expression for each UI component, which would limit search results to product-specific content, rather than including all of the help content contributed by any doc plug-in.

Note: For each helpKey constant, the help context should be defined *before* the help search expression is defined. If the Documentation team chooses to combine multiple helpKey constants in a single help context, all of the helpKey constants associated with that help context should be associated with the same search expression.

7.3. Defining Context-Sensitive Help Plug-ins with DITA

Any DITA map document that defines an online documentation (doc) plug-in can be modified to *also* define a corresponding context-sensitive help plug-in by inserting the appropriate context-related markup.

Note: The DITA map markup described in the following sections does **not** require DITA specializations; it relies entirely on standard DITA-XML elements and attributes.

7.3.1. Identifying UI Plug-ins Associated with Help Contexts

DITA maps for context-sensitive help plug-ins must contain a <topicmeta> element as the first child of the <map> element, and for each UI plug-in whose help context IDs are identified in the map, that <topicmeta> element must contain one <othermeta> element that identifies the UI plug-in.

Note: The <othermeta> elements that identify UI plug-ins should be the last child elements in the <topicmeta> element. The <topicmeta> element may contain any other valid child elements (preceding the <othermeta> elements that identify UI plug-ins).

The <othermeta> element's *name* and *content* attribute values will be used to identify UI plug-ins in the org.eclipse.help.contexts extension, which is declared in the plug-in manifest (plugin.xml file) of the context-sensitive help plug-in. For example:

```
<map id="org.eclipse.datatools.ui.doc">
   <topicmeta>
      ...
      <othermeta name="ui-plugin"
         content="org.eclipse.datatools.connectivity.ui"/>
      <othermeta name="ui-plugin"
         content="org.eclipse.datatools.connectivity.ui.dse"/>
   </topicmeta>
   ...
</map>
```

The *name* attribute value **`"ui-plugin"`** is a fixed, literal string. The *content* attribute value is the Eclipse plug-in ID of a UI plug-in.

Note: Nested maps that contribute context-related help topics must include the same markup to identify each UI plug-in whose context IDs are identified in that map.

7.3.2. Defining Related Topics Associated with Help Contexts

DITA maps for context-sensitive help plug-ins contain <resourceid> elements, each of which identifies a concrete help context ID associated with a DITA topic.

The <resourceid> element is a child of a <topicmeta> element. For example:

```
<topicmeta>
   <resourceid id="help_context_ID_string"/>
</topicmeta>
```

For each DITA topic associated with a help context ID, the <topicref> element that points to that topic contains a <topicmeta> element (with a <resourceid> child element). This defines the association of a help context ID with that topic. For example:

```
<topicref navtitle="label attribute in contexts.xml topic element"
   href="path/to/topic.xml">
   <topicmeta>
      <resourceid id="help_context_ID_string"/>
   </topicmeta>
</topicref>
```

Any DITA topic can be mapped to multiple help context IDs by inserting as many <resourceid> child elements as necessary in the <topicmeta> element. For example:

```
<topicref navtitle="label attribute in contexts.xml topic element"
   href="path/to/topic.xml">
   <topicmeta>
      <resourceid id="help_context_ID_1"/>
      <resourceid id="help_context_ID_2"/>
      <resourceid id="help_context_ID_3"/>
   </topicmeta>
</topicref>
```

Note: It may be preferable to combine help contexts in the helpKey properties file, instead of defining mapping for multiple help contexts to a single topic in the DITA map. This is a judgment call for the Documentation team responsible for maintaining the DITA map. For more information, see Defining Help Context IDs.

Other considerations for DITA maps that define context-sensitive help plug-ins:

- Markup for context-to-topic mapping is needed only for the DITA topics that will be context-sensitive help targets (i.e., Related Topics shown in the Help view).
- When a single DITA topic appears more than once in a map, only the first instance of a <topicref> that points to that topic needs the context-sensitive help markup.
- Nested maps that contribute DITA topics related to help contexts must include the same context-sensitive help markup.

7.3.3. Context-Specific Help Content

Each <topicmeta> element that wraps a <resourceid> element may optionally contain one <searchtitle> element or one <shortdesc> element, or both (one of each):

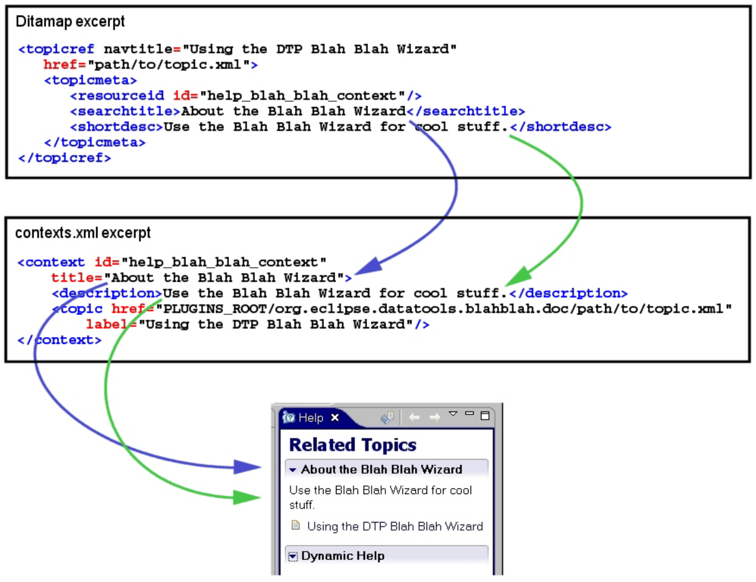
- The <searchtitle> element is used to supply the value of the *title* attribute on the <context> element in the Eclipse context XML file.
- The <shortdesc> element is used to supply the content of the <description> element in the Eclipse context XML file.

For example:

```
<topicref navtitle="label attribute in contexts.xml topic element"
   href="path/to/topic.xml">
   <topicmeta>
      <searchtitle>Optional text to override the help About title.
      </searchtitle>
      <shortdesc>Text for context description.</shortdesc>
      <resourceid id="help_context_ID_string"/>
   </topicmeta>
</topicref>
```

Note: The <searchtitle>, <shortdesc>, and <resourceid> elements must appear in the <topicmeta> element in the order shown above.

The following illustration shows how the <searchtitle> and <shortdesc> elements in a DITA map can be transformed into markup in the Eclipse context XML file, and how those items in the context XML file provide the context-specific content rendered in the Help view.



Ditamap excerpt

```
<topicref navtitle="Using the DTP Blah Blah Wizard"
   href="path/to/topic.xml">
   <topicmeta>
      <resourceid id="help_blah_blah_context"/>
      <searchtitle>About the Blah Blah Wizard</searchtitle>
      <shortdesc>Use the Blah Blah Wizard for cool stuff.</shortdesc>
   </topicmeta>
</topicref>
```

contexts.xml excerpt

```
<context id="help_blah_blah_context"
    title="About the Blah Blah Wizard">
    <description>Use the Blah Blah Wizard for cool stuff.</description>
    <topic href="PLUGINS_ROOT/org.eclipse.datatools.blahblah.doc/path/to/topic.xml"
        label="Using the DTP Blah Blah Wizard"/>
</context>
```

Help
Related Topics
About the Blah Blah Wizard
Use the Blah Blah Wizard for cool stuff.
Using the DTP Blah Blah Wizard
Dynamic Help

8. Testing Dynamic Context-Sensitive Help

Dynamic context-sensitive help in Eclipse involves complex interactions between several components, so testing it requires a properly configured runtime platform, with both user interface (UI) components and user assistance (UA) components installed.

Like any other functionality, dynamic context-sensitive help should be tested to assess its quality.

Note: This document deals with testing the functionality of dynamic context-sensitive help, not the quality (or usefulness) of the help content.

8.1. Test Objectives

For projects that use the DTP help-helper, there are two primary objectives in testing dynamic context-sensitive help:

- Determine if all abstract helpKey constants are mapped to concrete help context IDs (e.g., by comparing the contexts XML file with helpKey properties files)
- Determine if help context IDs work correctly, providing the correct context-specific help content in the Help view or dialog tray

8.2. Test Procedures

For each UI control associated with a help context, test procedures should verify the following:

- Keyboard shortcut (e.g., F1) opens the Help view or dialog tray, with correct context-specific content
- Help icon (if available) opens the Help view or dialog tray, with correct context-specific content
- Changing focus on the UI control invokes refresh in the open Help view or dialog tray (dynamic context-specific content)

8.3. Plug-in Spy

The Plug-in Spy tool (provided in **org.eclipse.pde.runtime**, since Eclipse 3.4) shows various information about the workbench, including help context IDs associated with the current (in focus) UI control. This is particularly useful for testing and troubleshooting context-sensitive help.

9. Legal Notices

Sybase, Inc.
One Sybase Drive
Dublin, CA 94568

http://www.sybase.com