The TimeSafe® Configuration Management System

AccuWork Issue Management Manual

Version 4.0.2

May, 2006

AccuWork Issue Management Manual
May, 2006

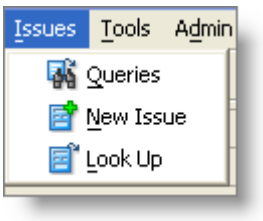
Table of Contents

Working with Issue Records

This chapter discusses usage of AccuWork, the AccuRev issue-management facility from the viewpoint of the day-to-day user. The topics include:

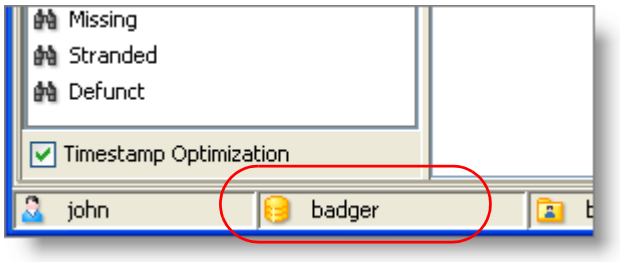
- *Accessing a Particular Issues Database*
- *Creating a New Issue Record*
- *Viewing and Modifying an Existing Issue Record*
- *Printing an Individual Issue Record*
- *Using AccuWork Queries*
- *Printing Query Results*

This chapter covers the commands on the **Issues** menu: **New Issue**, **Look Up**, and **Queries**. See *Designing Issues Databases and Edit Forms: The Schema Editor* on page 23 for a discussion of the AccuWork Schema Editor, the tool for creating issues databases and their edit forms.



Accessing a Particular Issues Database

In the AccuRev GUI window, you can access any number of issues databases, using different tabs. The name of the depot you're using in the current tab is displayed at the bottom of the window.



When you invoke the AccuWork **New Issue** or **Look Up** command, it applies to the current depot — the one being used by the current tab. (If there is no current depot, AccuRev prompts you to select one.) When you invoke the **Queries** command, AccuRev prompts you to select a depot.

Creating a New Issue Record

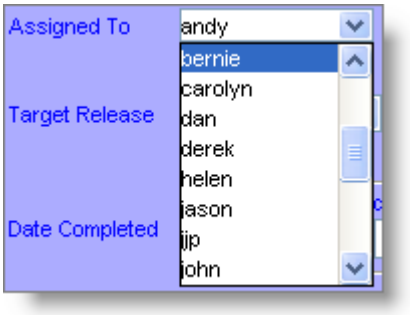
Use the **New Issue** menu command or toolbar button to open an empty edit form. The form is a new tab in the GUI's tabbed display, labeled "New Issue". Depending on the design of the issues database, you'll notice some or all of the following features:

- Certain fields have already been initialized with default values.
- Certain field-labels are in a contrasting color (by default, red), indicating that they are required fields. You cannot save a new issue record until a value appears in all required fields. Some of those fields may have gotten their values by being initialized.
- You can't enter a value in the **Issue** field. This is the unique issue-number for this particular issue record. AccuWork assigns the number when you save the issue record.

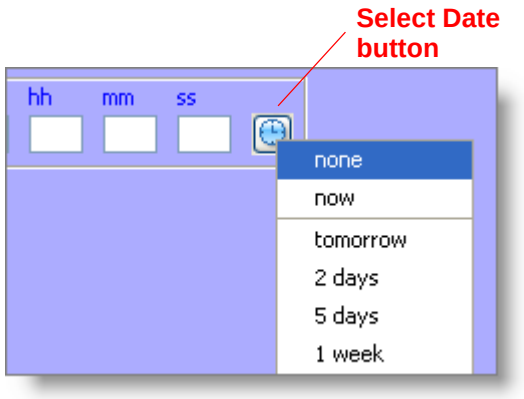
Enter values in the fields, using the mouse or **Tab** and **Shift-Tab** to move from field to field. Within a multiple-line text field, the **Tab** key inserts a TAB character, rather than jumping to the next field.

With several kinds of fields, you can enter a value in some way other than simple typing:

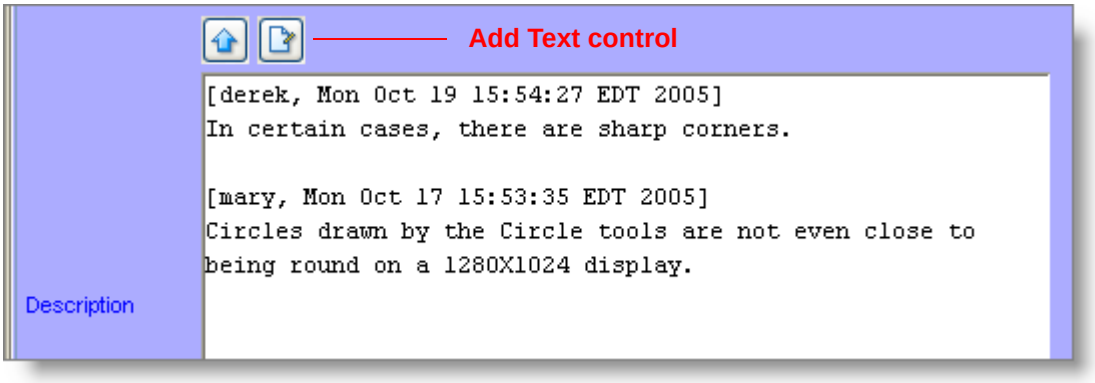
- **list-box** — a multiple-choice box works in the same way as on Web forms. You can click on the arrow control to see some or all the choices (maybe with a scroll bar). You can use arrows to scroll through the choices, without having to open the list. Typing a letter advances to the next item that begins with that letter.



- **timestamp** — You can fill in the various parts of a timestamp field manually, or use the **Select Date** button to display choices that fill in the fields automatically. Once these fields are filled in, you can go back an revise them.

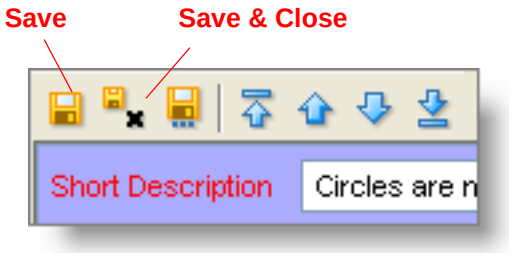


- **log** — A log field is a variant of the multiple-line text field. You can type directly into such a field, or you can click the **Add Text** control, which pops up a separate window. The text that you type (or paste) into this window is added to the current contents of the field, along with a "log stamp" that incorporates your AccuRev username and the current time.

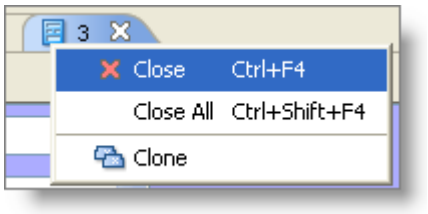


The edit form will have multiple pages, which appear as subtabs within the overall "New Issue" tab. And typically, there's a "header section", which always remains visible as you switch from page to page. Use the mouse to switch among multiple pages.

When you're ready (have you filled in all the required fields?), click the **Save** button or the **Save & Close** button in the toolbar above the New Issue form. AccuWork assigns an issue-number to the new record and stores it in the depot. With **Save**, the new record remains on-screen; the issue-number replaces the "New Issue" label and appears in the **Issue** field. With **Save & Close**, the tab containing the edit form disappears.



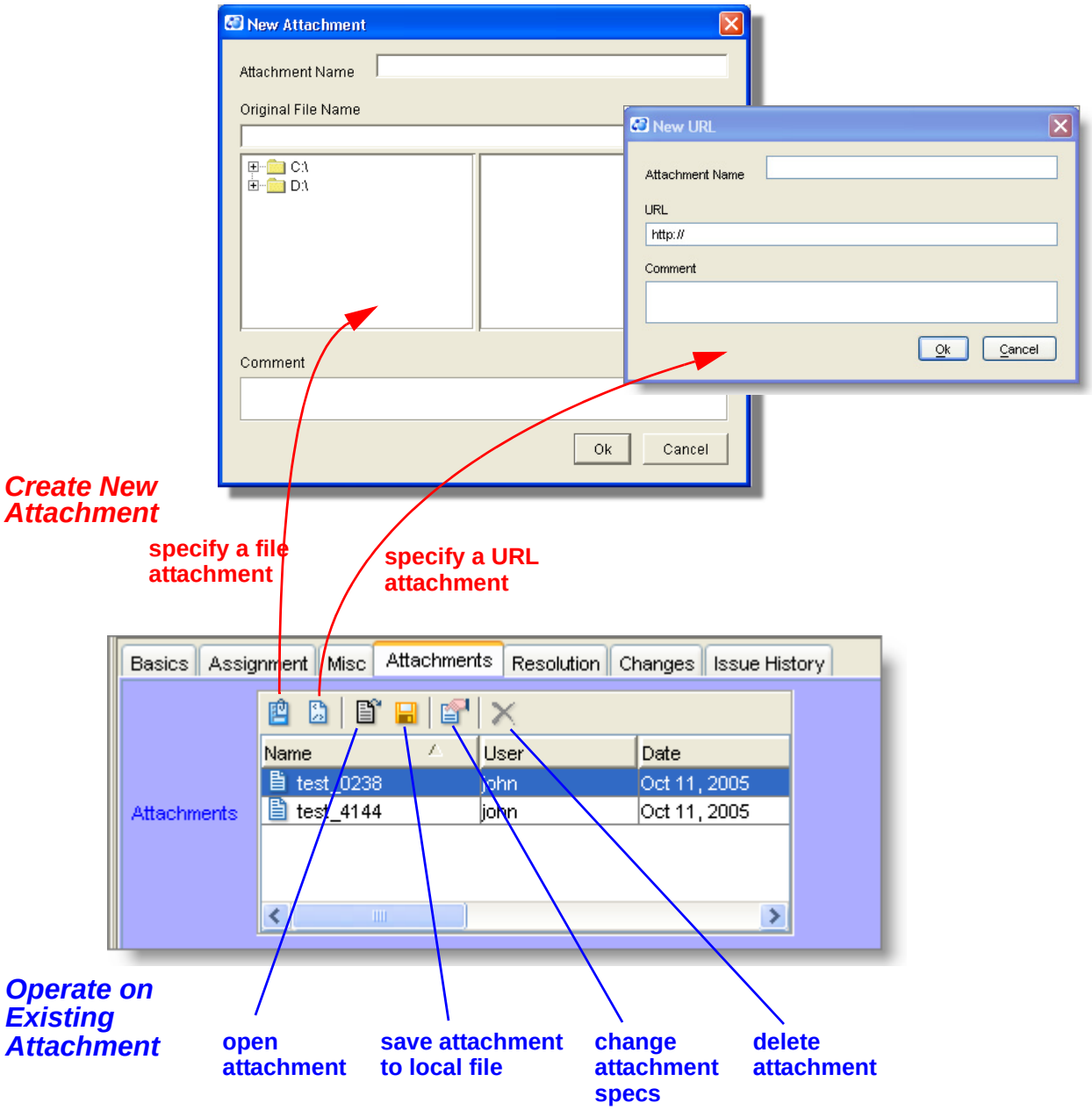
At any time, you can close an edit-form tab as you would any GUI tab: right-click the tab's title, then select **Close** from the context menu. If AccuRev Look And Feel is enabled (**Tools > Preferences**), you can close the tab using the "X" icon on the tab control itself.



If you want to create another issue record, use the **New Issue** menu command or toolbar button again.

Creating Attachments to an Issue Record

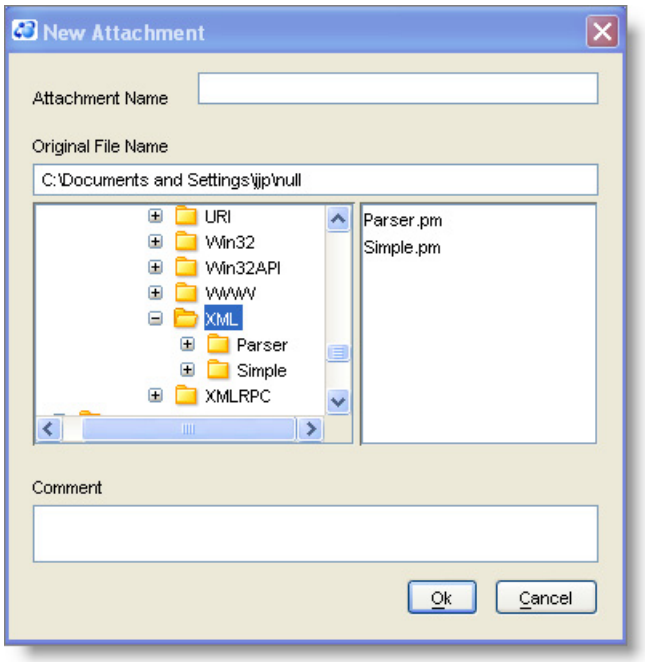
An edit form can contain one or more attachment fields. In each such field, you can specify one or more files and/or Internet addresses (URLs) to be attached to the current issue record. AccuWork displays the attachments data as a table.



In addition to specifying the location of a file or Internet resource, you enter a Name and optional Comment. AccuWork automatically fills in your username, the date, and the size of the attached file. (Internet URLs are assigned a size of 0.) If the edit-form field is not large enough to show all these attachment parameters, use the scroll bar to see all the data. You can also resize and rearrange the columns of an attachments table.

An attachment field includes its own toolbar, with these buttons:

New Attachment



Define a new file attachment for this issue record. A file selection dialog box appears, in which you can specify one or more files, a name for the attachment, and a comment string.

You can specify multiple files at once — each one becomes a separate file attachment. In this case, the Attachment Name input field is disabled; each filename is automatically assigned as the attachment name. The comment string that you specify is assigned to each file attachment.

New URL

Define a new attachment to be the address (URL) of an Internet resource. A dialog box appears, in which you specify a URL, a name for the attachment, and a comment string. AccuWork inserts the string **http://** in the URL field. You can erase this if you want to specify a location accessed by another Internet protocol, such as **ftp://**.

Open Attachment

Open the existing attachment that is currently selected, using the appropriate program.

Save Attachment As

Create a copy of the currently selected attachment on the client machine.

Properties

Launch a Properties window, displaying the definition of the currently selected attachment. You can use this window to change the attachment's Name or Comment value.

Delete Attachment

Remove the attachment from the issue record (and from the depot).

When you save the issue record, each file is copied to the depot, so that the data always remains available through the issue record, even if an original file is deleted.

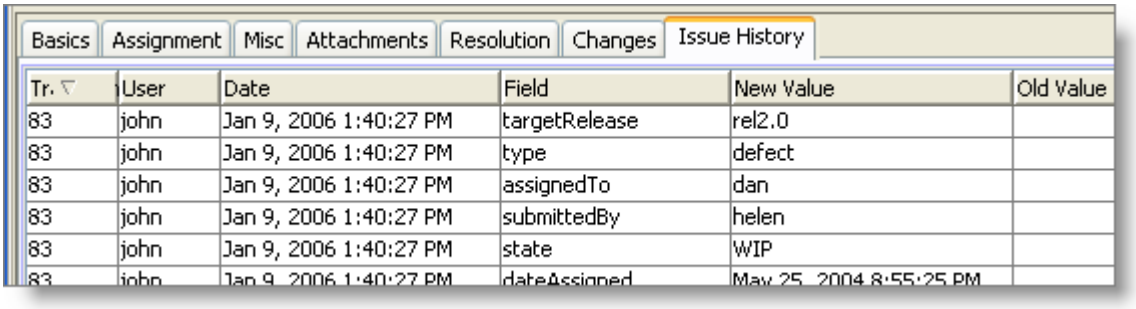
Canceling Creation of a New Issue Record

If you decide not to create a new issue record, just close the edit form (as described above) without clicking the **Save** button.

Issue Records and Transactions — the Issue History Page

When you create a new issue record (or update an existing record), a **dispatch** transaction is written to the depot's transaction log — the same log that records AccuRev transactions, such as **keep** and **promote**. It's important to distinguish the depot's transaction log (which is a database in its own right) from the AccuWork issues database stored within the depot.

AccuWork automatically adds an **Issue History** page to each edit form. This page shows how an issue record has changed over time. It displays all the **dispatch** transactions for an issue record, including the initial **Save**. Each row of the table shows the change to one field made by a subsequent **Save**.



Tr.	User	Date	Field	New Value	Old Value
83	john	Jan 9, 2006 1:40:27 PM	targetRelease	rel2.0	
83	john	Jan 9, 2006 1:40:27 PM	type	defect	
83	john	Jan 9, 2006 1:40:27 PM	assignedTo	dan	
83	john	Jan 9, 2006 1:40:27 PM	submittedBy	helen	
83	john	Jan 9, 2006 1:40:27 PM	state	WIP	
83	john	Jan 9, 2006 1:40:27 PM	dateAssigned	May 25, 2004 8:55:25 PM	

If you double-click any row of the Issue History table, AccuWork opens a read-only edit form, showing the state of the issue record at the time of that particular **Save**.

The server_dispatch_post Trigger

The act of saving a new issue record (or updating an existing record) may cause a user-defined procedure to be performed on the AccuRev Server machine. This is implemented through AccuRev's trigger facility. Typically, a trigger script sends an email message to one or more interested parties — for example, the user to whom a bug has been assigned.

For more information, see *Post-Operation Triggers* on page 42 of the *AccuRev Administrator's Manual*.

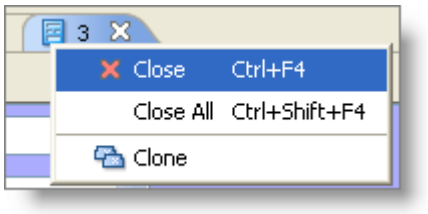
Viewing and Modifying an Existing Issue Record

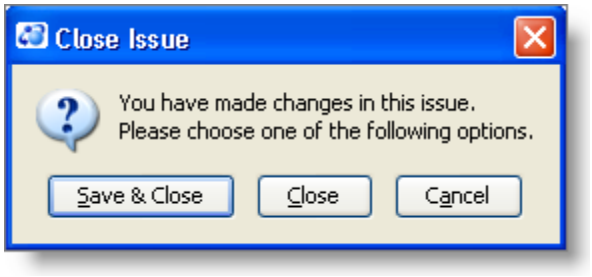
The easiest way to retrieve an existing issue record is through its unique issue-number. Use the **Look Up** menu command or the **Open Issue** toolbar button. AccuWork prompts you to enter an issue-number, retrieves the record, and displays it in the edit form. You can also view and edit issue records that have been selected by a query; see *Query Results Pane* on page 12.

> Note: issue-numbers are unique within a given depot. Several different depots might have an issue-record numbered 382. Make sure you've opened the right depot!

If you wish, modify one or more fields, then click the **Save** or **Save &Close** button. Required-fields restrictions apply when you're modifying an existing issue record, just as they do when you're creating a new one.

If you don't want to modify the issue record, or if you change your mind after modifying some fields, just close the GUI tab containing the edit form without clicking the **Save** button. AccuWork asks for confirmation, to make sure that you don't mistakenly discard work that should be saved:





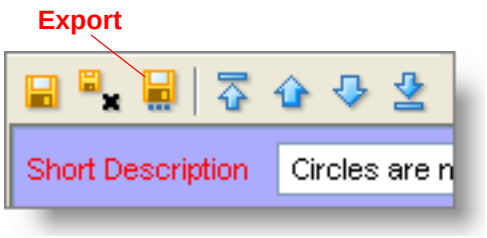
While you're viewing or modifying an issue record that you've displayed with **Look Up**, browse arrows are enabled in the edit form's toolbar. This makes it easy to view a set of consecutive issue records. If you've made some changes to an issue, AccuWork prompts you to save or discard those changes before switching to the previous or next one.

As described in section *Issue Records and Transactions — the Issue History Page* on page 6, the updating of an issue record is recorded as a **dispatch** transaction in the depot's transaction log.

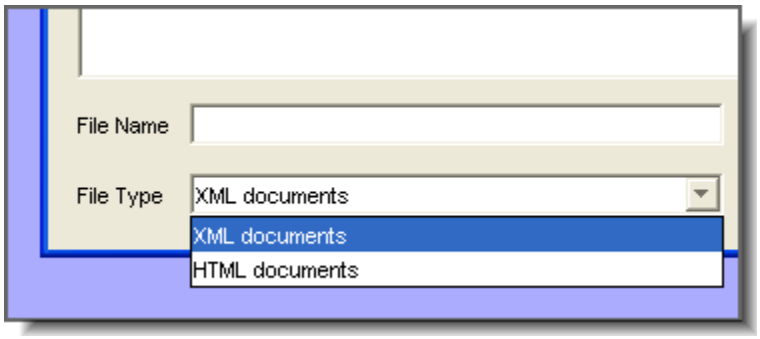
Printing an Individual Issue Record

At any time when you're using an edit form, whether creating a new issue record or modifying an existing one, you can "print" it. For AccuWork, printing means "publish to the Web" — which means either "create an HTML file" or "create an XML file".

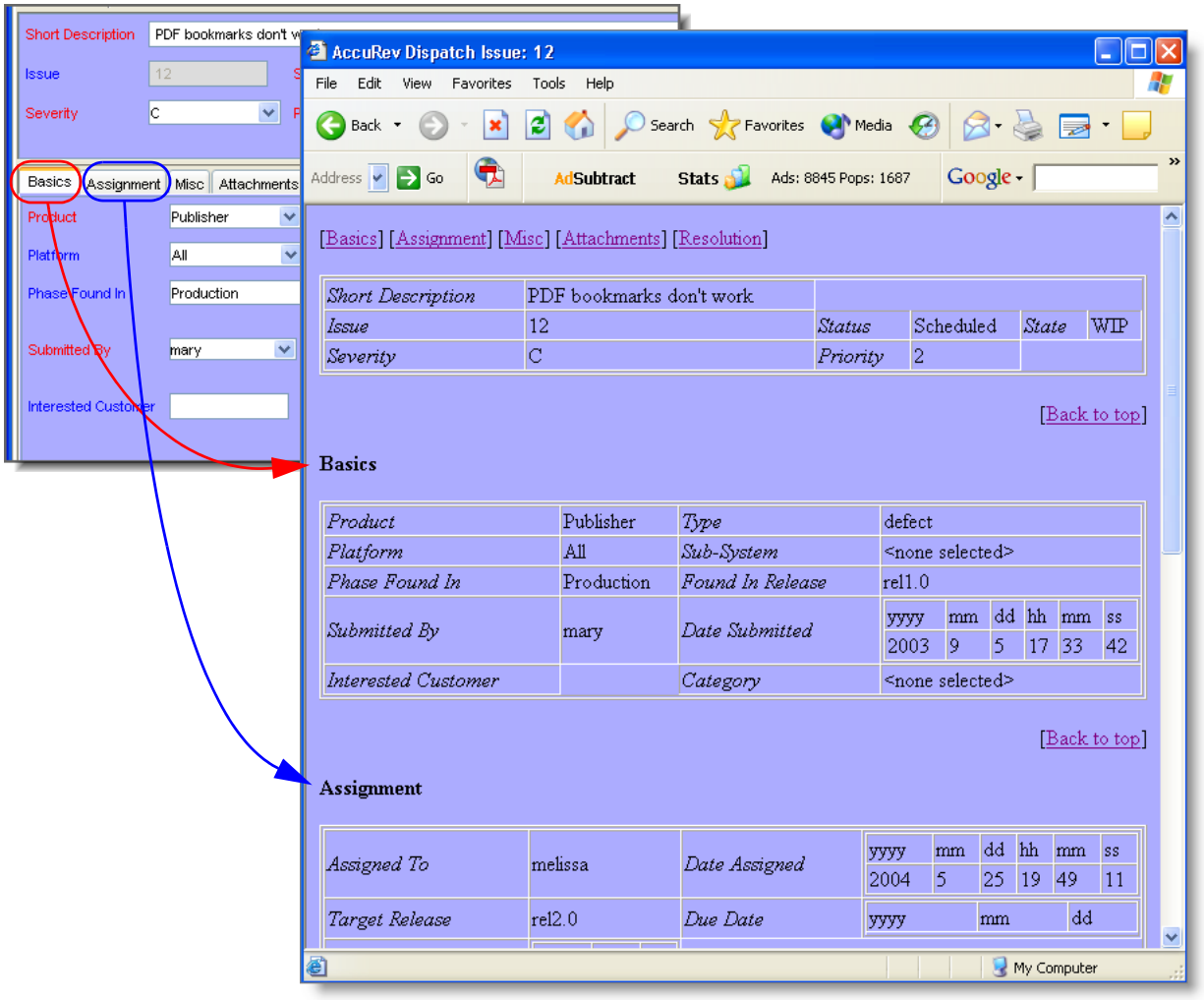
- Click the **Export** button in the form's toolbar.



- Select the file type in the File Chooser window that appears.
- Specify a pathname for the export file, using the navigator and the File Name input field. (You don't need to specify the **.html** or **.xml** suffix — AccuWork adds it automatically.)



When creating an HTML file, AccuWork outputs all the content of the issue record and approximates the form layout, too. Even if the edit form has multiple pages, you need to “print” only once. All of the pages are combined into a single HTML or XML file:



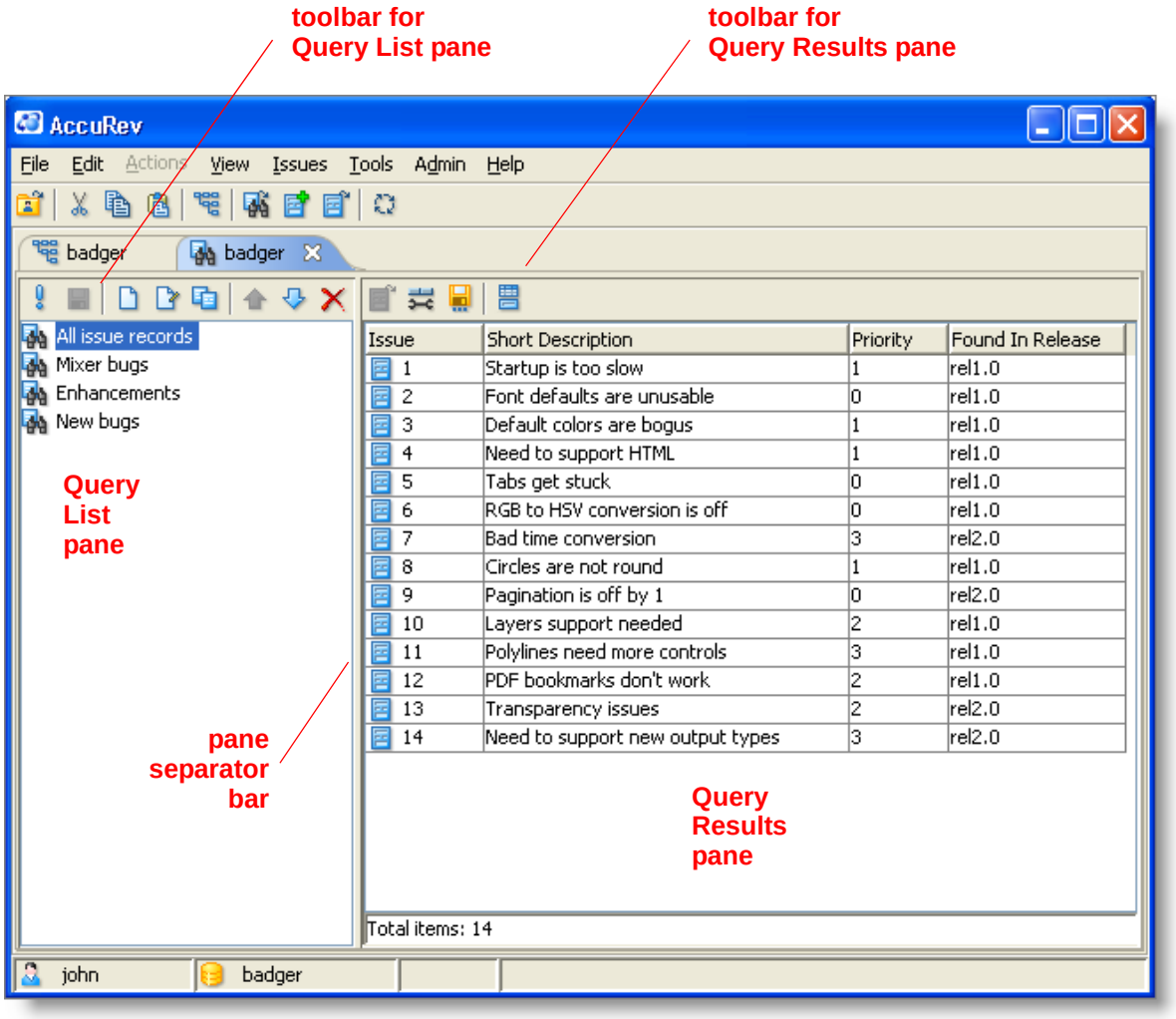
On Windows machines, AccuWork automatically invokes a Web browser on the HTML file it creates. If you wish, resize the browser window to optimize the look of the issue record. HTML documents automatically adjust to changes in window width. Then, use the browser’s print command to create hardcopy of the issue record.

Using AccuWork Queries

Retrieving one issue record at a time is useful in many situations. But robust issue-management systems must also support queries which retrieve groups of records according to user-defined selection criteria. AccuWork has a point-and-click interface for creating and editing queries; it enables you to create simple queries quickly, and to create sophisticated queries in a straightforward, reliable way.

By default, the queries that you create are private queries, which cannot be seen by other users. You can declare any query to be a public query. Such queries are visible to all users, who can use and copy them, but cannot modify your original.

Use the **Issues > Queries** command or the **Queries** toolbar button to open a new Queries tab in the GUI window. The Queries tab includes two panes, each with its own toolbar:



- The **Query List** pane lists the names of all your existing queries. (Each AccuRev user has his own set of queries; there are no “global” queries available to all users.) Using the toolbar in this pane, you compose new queries, view/revise/rearrange/execute existing queries, and delete queries.
- When you execute a query, the set of issue records selected by the query are displayed as a results table in the **Query Results** pane. If you’ve set a default query, it’s executed automatically when you open the Queries tab. AccuWork remembers query results as long as the Queries tab stays open. It’s easy to browse through the results of several queries, switching back and forth instantly between different queries’ results tables.

- If you click the **Show Issue Form** button in the Query Results pane's toolbar, a third pane appears. This pane is a fully functional edit form, which you can use to view and modify the issue record currently selected in the Query Results pane. A user preference (**Tools > Preferences**) causes this pane to appear automatically when you open a Queries tab.

You can use the separator bar between the panes to adjust their relative size. And remember that the GUI window itself is resizable, too.

Typical Workflow

Here's a typical workflow for creating and using a query:

1. Create a new query: click the **New Query** button to open the Edit Query window with a blank query, assign the query a name, and specify one or more clauses that select issue records.
2. In the Query Results pane, select click the **Setup Columns** button and select certain fields to appear in the results table.
3. Click the **Run Query** button to execute the query, loading data from selected issue records into the results table.
4. Optional: save the results table in HTML format; use a Web browser to view and print the table.
5. Use the **Save All** button on the Queries toolbar to save the query in the depot, for future use. Each user's queries are stored separately.

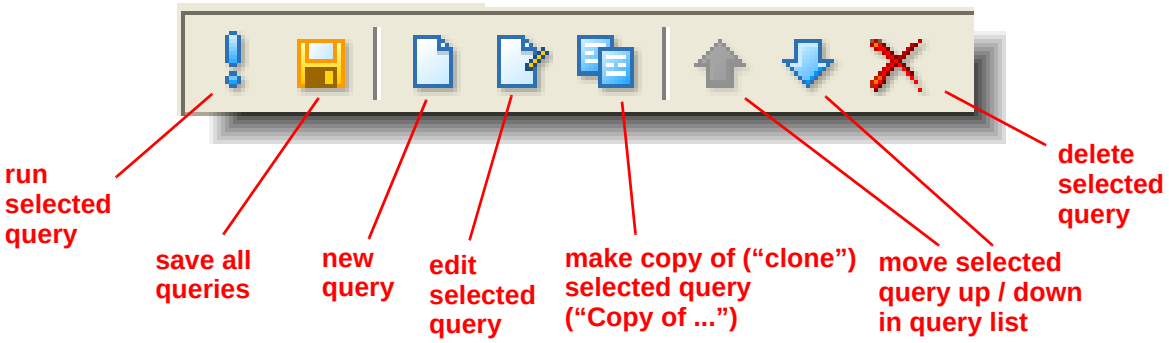
You can refine a query as much as you want, changing the record-selection criteria and the fields that appear in the results table.

The following sections describe the components of the Queries tab in more detail.

Query List Pane

The Query List pane contains a list of the private queries you've defined for the current depot (issues database), along with any public queries defined by you and/or other users. The listing is not alphabetical — whenever you create a new query, it's simply added to the end of the list. You can rearrange the order of the queries using the toolbar buttons and . Your queries (both private and public) are always listed above other users' public queries.

You can invoke the following commands on the currently selected query in the Query List pane — either through its context menu or using the Query List toolbar:



Run query

Executes (or re-executes) the selected query, displaying the results in the Query Results pane.

Whenever you select a particular query in the Query List pane, AccuWork displays in the Query Results pane the most recent results of running that query. (The cache of previous query results is cleared when you close the Queries tab.) You can update a results table to reflect recent AccuWork transactions by clicking the **Run Query** button on the Query List toolbar.

You don't need to click the **Run Query** button when you revise a query in the Edit Query window — AccuWork automatically executes the revised query and updates the results table.

Save all queries

Save all your private queries. (There is no way to save a single query.) Your queries are stored in the AccuRev repository within the depot directory, in an XML-format file:

`.../storage/<depot-name>/dispatch/config/user/<principal-name>/query.xml`

The <*principal-name*> directory in this pathname causes the private queries for each user to be stored separately.

Although your queries are stored in the depot, they are not version-controlled in the way AccuRev files are. For example, there is no command that displays or reinstates your queries as you saved them two days ago.

New query
Edit query

Create a new query or revise an existing one. See *The Edit Query Window: Creating and Revising Queries* on page 13.

Clone query

Create a private query that is a copy of the selected private or public query. The new query is initially named "Copy of ...", but you can change the name.

Move query up
Move query down

Change the position of the selected query in the Query List pane. Note that your own queries (both private and public) are always listed above other users' public queries.

Delete query

Use the **Delete Query** button to remove the currently selected query. The deletion does not take effect until you save your queries. If you close the Queries tab without saving your queries, the "deleted" query will still exist the next time you open the Queries tab. This may or may not be the right thing to do: closing the Queries tab without saving also discards changes you've made to other queries since the most recent save.

Set as Default
Disable as Default

Designates the query to be — or to stop being — the *default query* for the current issues database. The query name is redisplayed in ***bold-italic*** to indicate that it's the default query.

The default query is executed whenever you open a new Queries tab. It is also executed by the transaction-level integration between AccuRev's configuration management and issue management capabilities. See *Integrations Between Configuration Management and Issue Management* on page 54. And it is executed by the **Send to Issue** command

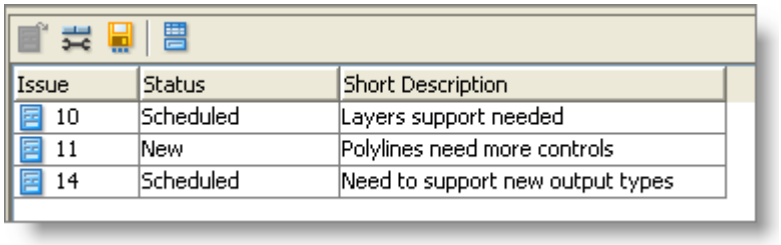
A query loses its status as the default query when you select **Disable as Default** from its context menu, or when you select another query as the default.

Set as Private
Set as Public

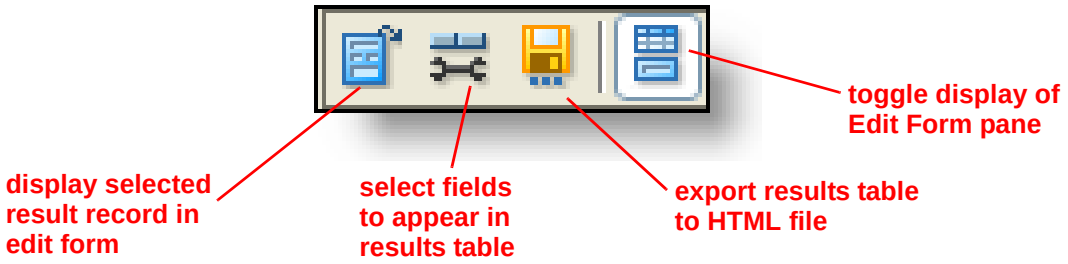
Changes a query that you created from public to private, or vice-versa.

Query Results Pane

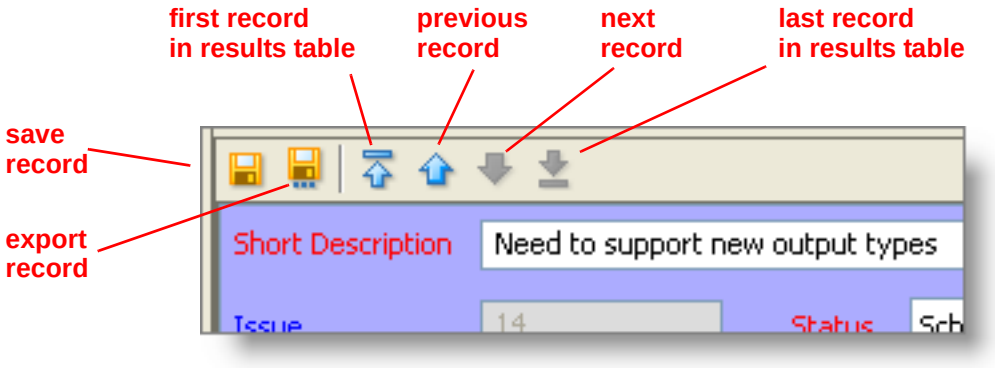
The Query Results pane displays the results of a query as a results table. Each row of the table displays one issue record; each column displays a particular issue-record field.



Using the mouse, you can rearrange and resize the table's columns. You can also specify a single-level or multiple-level sort order for the table's rows. Additional operations are available through the Query Results pane's toolbar:



- **Open Issue**: Open an edit form to view/revise the selected issue record in the results table.
- **Setup Columns**: Change which columns (fields) appear and their order
- **Export Table**: Create an HTML file containing the entire contents of the results table.
- **Show Issue Form**: Toggles whether an Edit Form pane appears below the Query Results pane. The currently selected record in the Query Results pane is loaded into the Edit Form pane. Toolbar buttons above the edit form enable you to browse some or all of the other records in the results table. You can modify and save an issue record, as described earlier in this chapter; likewise, you can export an issue record to an HTML or XML file.

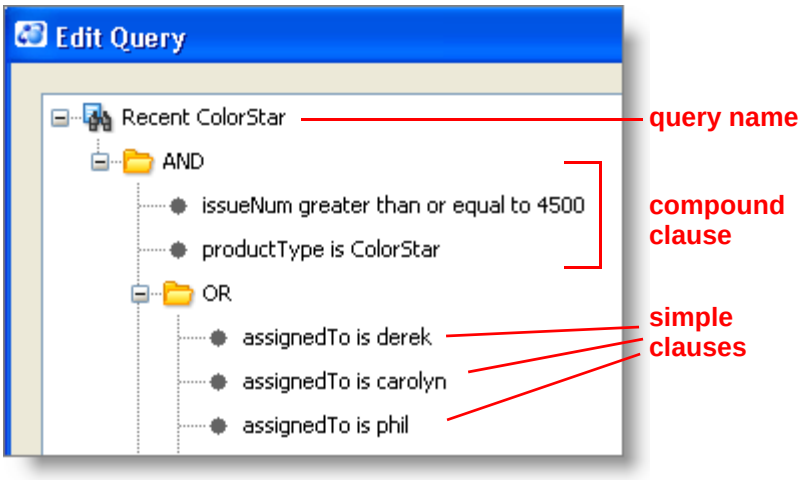


The **Open Issue** and **Setup Columns** commands are also available on the context menu of individual records in the Query Results pane. The context menu also contains the **Remove Column** command, which provides a quick way to revise the structure of the results table: AccuWork removes column that you right-click on.

The Edit Query Window: Creating and Revising Queries

The Query List pane's toolbar includes **New Query** and **Edit Query** command buttons. These commands open an Edit Query window, in which you compose a new query or revise an existing one. A query is displayed as a hierarchy, which you navigate using the familiar expand/collapse controls. The hierarchical organization is a natural fit, because each query is, itself, a hierarchy of simple clauses and compound clauses:

- The first level contains the query's name.
- A simple clause, such as "value of the **assignedTo** field is **mary**", can live at any level.
- A compound AND clause *(cls-a* AND *cls-b* AND *cls-c* AND …) consists of the operator AND at one level of the hierarchy and any number of clauses *cls-a*, *cls-b*, *cls-c*, etc. at the next sublevel.



- Compound OR clauses are structured hierarchically in exactly the same way.

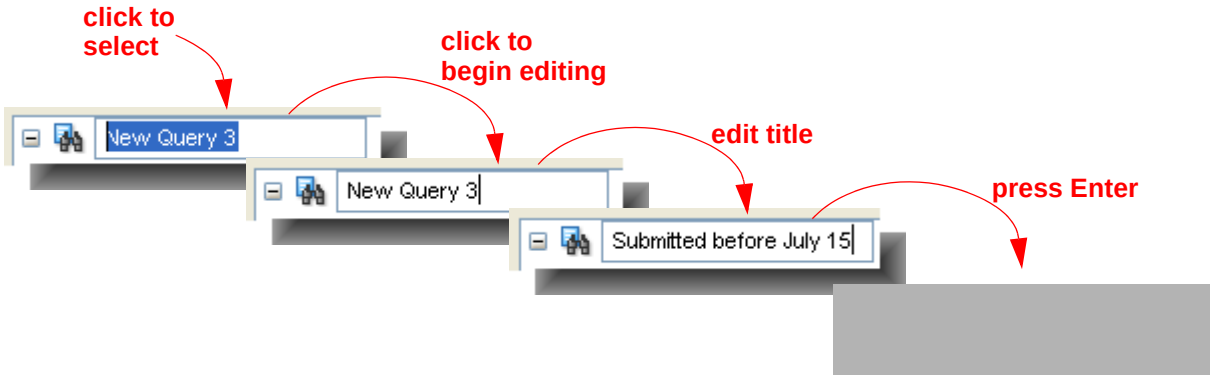
The hierarchy can get deep, because each of the clauses *cls-a*, *cls-b*, etc. can be either simple or compound. For example, the sophisticated query pictured above can be expressed as:

> "Retrieve each bug report that is assigned to either **derek, jjp**, or **mary**, and applies to product **ColorStar**, and is numbered above 4500."

The following sections fill in the details of working in the Edit Query window.

Naming a New Query / Renaming an Existing Query

When you click the **New Query** button to create a new, empty query, AccuWork assigns it a placeholder title ("New Query *nnn*"). You can edit the title now or whenever the Edit Query window is open: click the title once to select it (if it is not already selected); then click a second time to begin editing it. Not too fast! Double-clicking a title is equivalent to using the expand/collapse control.

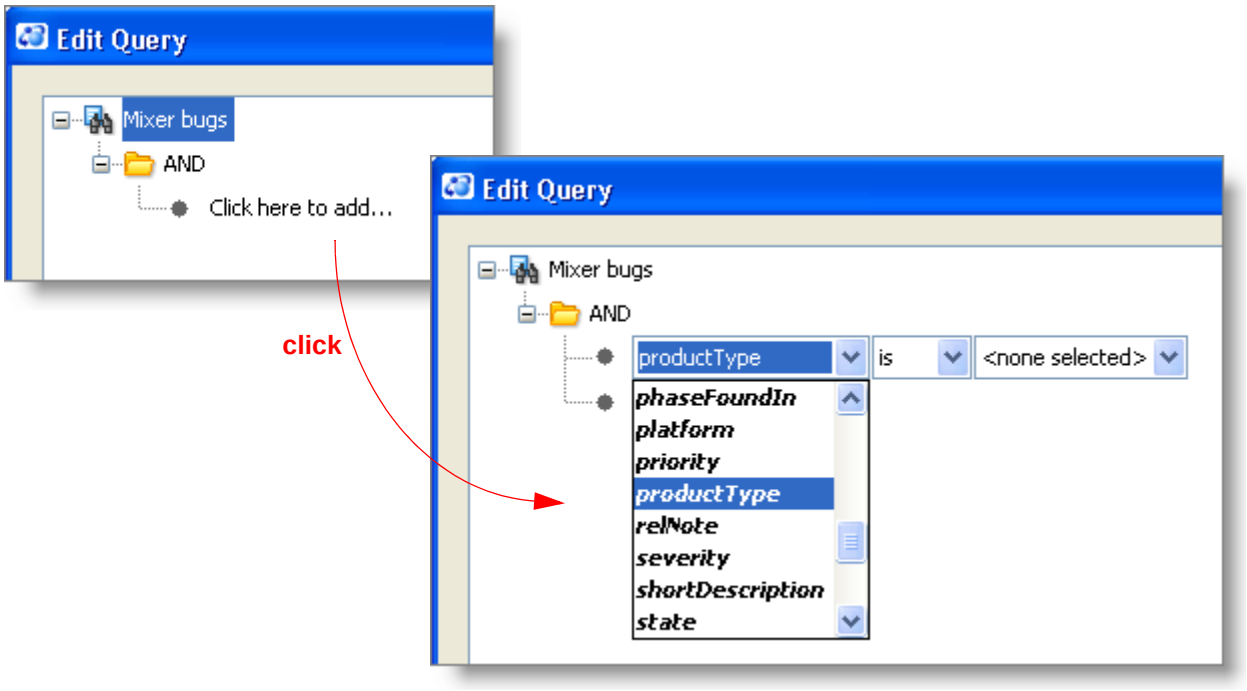


Creating a Simple Clause

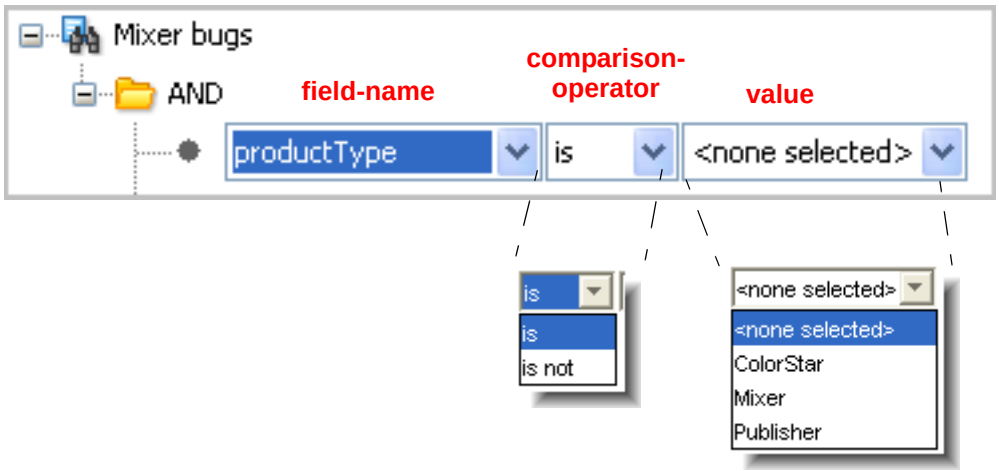
Every query consists at least one simple clause. A simple clause has three parts:

```
<field-name>   <comparison-operator>   <value>
```

The point-and-click interface makes creating a simple clause easy and (almost) foolproof. Start by clicking one of the “Click here to add ...” placeholders in the query:



First, you must select the <*field-name*> part of the clause from the list-box containing all the field-names. When you select a field-name, the query editor automatically adjusts the <*comparison-operator*> and <*value*> parts of the clause, based on the selected field. In the example below, the user has selected field-name **productType**, whose value must be one of these names: **ColorStar**, **Mixer**, **Publisher**.



The table below shows all the AccuWork data types, along with the corresponding choices for the <*comparison-operator*> and <*value*> parts of a simple clause. For more on data types, see *Data Types* on page 27.

Field Type	Comparison Operators	Values
Text	contains matches does not contain does not match equal to not equal to less than less than or equal to greater than greater than or equal to	Any character string. (Do not enclose it in quotes.) The value is always interpreted as a string literal; there is no way to specify the value of some other field here. The comparison is always a string comparison, never a numeric comparison. For example, the value **3** is greater than the value **25**. The **matches** and **contains** operators — and their negations — are case-insensitive.
Timespan	equal to not equal to less than less than or equal to greater than greater than or equal to	A numeric value, representing an amount of time. Note: users specify the value in the edit form as a number of *hours* (e.g. 7.5); an XML-format dump of the issue record created by the **Export** command reports the value as a number of *minutes* (e.g. 450).
Choose	is is not	One of the strings specified in the definition of this field in the Schema Editor.
List	is is not	One of the strings specified in the definition of a particular named list in the Schema Editor.
User	is is not is member of is not member of	One of the principal-names in the user registry maintained by the AccuRev server. Alternatively, a user-group defined in the registry.
Timestamp	is is not is before is after is before or equal to is after or equal to	An AccuRev timestamp.
Attachments	contains	Any character string. This string is compared to the **Name** of each of an issue record's attachments. See *Creating Attachments to an Issue Record* on page 3.
internal	equal to not equal to less than less than or equal to greater than greater than or equal to	An integer, identifying a particular AccuWork issue record (**issueNum** field) or a particular AccuRev transaction (**transNum** field).

As you “fill in the blanks” to create simple clauses, you’ll notice that AccuWork allocates new “Click here to add ...” placeholders. It makes sure that one of these placeholders is always available at each level of the query.

Creating a Compound Clause

A compound clause combines any number of subclauses together, using the same logical operator: AND or OR. (The NOT operator is not supported.) The subclauses to be combined can, themselves, be either simple or compound:

simple AND *simple*
simple AND *compound*
simple OR *compound* OR *simple*
compound AND *compound* AND *simple* AND *compound* AND *compound*
etc.

Note: a compound clause can contain a single subclause. This is logically equivalent to using the subclause by itself. In fact, the most basic query you can create is a compound AND clause (or compound OR clause) with one simple subclause.

What about this?

simple OR *simple* AND *simple*

Well, it depends. This might be a compound OR clause within a compound AND clause:

((*simple* OR *simple*) AND *simple*)

Or it might be a compound AND clause within a compound OR clause:

(*simple* OR (*simple* AND *simple*))

Whichever meaning was intended, AccuWork models the logical statement as one compound clause nested within another.

We saw above that the query editor automatically creates placeholders for simple clauses. But you must explicitly insert a compound-clause placeholder yourself, then fill in the subclauses. We’ll demonstrate with an example:

“Retrieve all **high**-priority bug reports assigned to either **mary** or **jjp**”

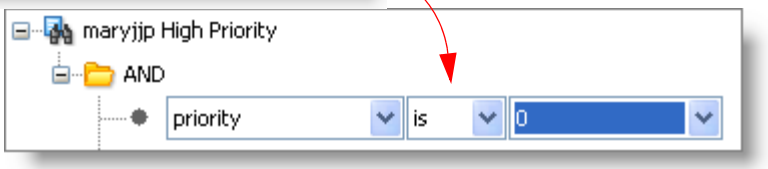
... which becomes a compound OR clause within a compound AND clause:

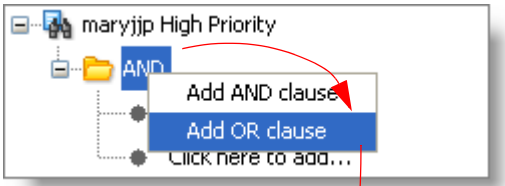
(fix_priority is **high** AND (assign_user is **mary** OR assign_user is **jjp**))

Here’s the procedure for creating this query:

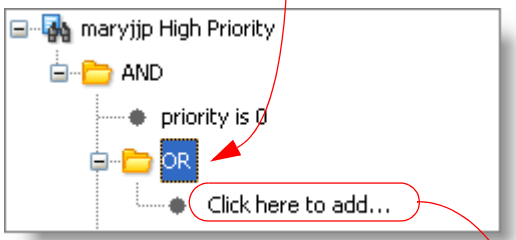
1. create a new query and name it

2. click the placeholder and fill in a simple clause

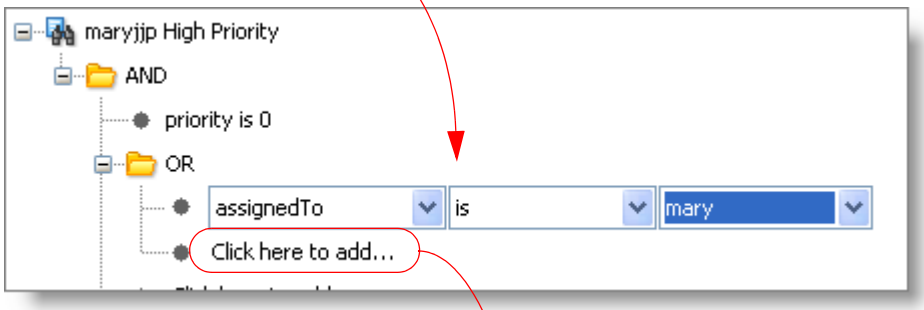




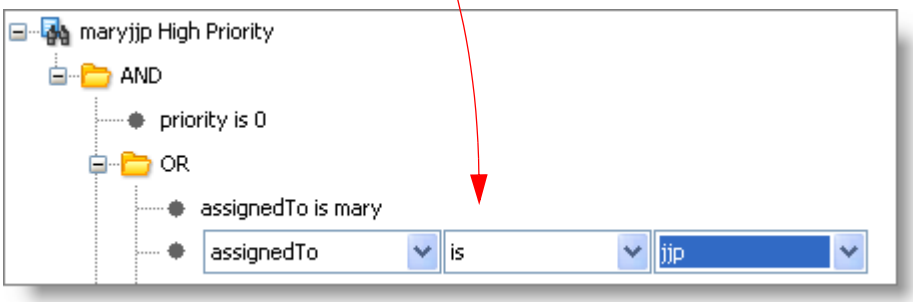
3. right-click at the parent level and create an OR subclause



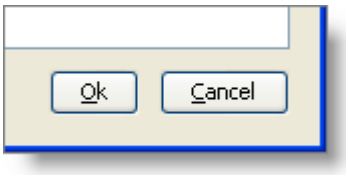
4. click the placeholder and fill in a simple clause



5. click the new placeholder and fill in another simple clause



Closing the Edit Query Window



Whether you're composing a new query or revising an existing one, you end by saving your work (**Ok** button) or discarding it (**Cancel** button). AccuWork automatically executes the query and displays a results table in the Query Results pane.

At this point, you'll often want to work on the design of the results table: specify additional fields to be displayed, and adjust the widths and order of the columns. For more on this, see *Working with a Results Table* on page 19.

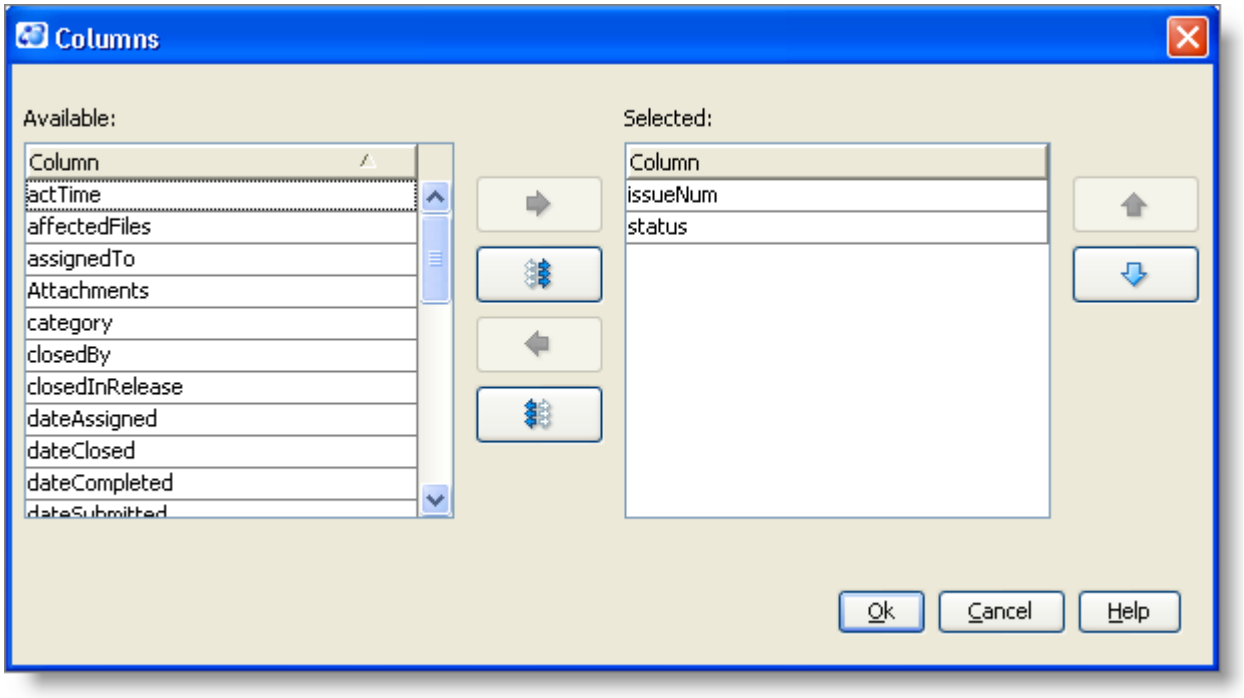
Working with a Results Table

Each query has its own results-table design: a set of columns (fields), in a particular order and with particular column-widths. The results table's design can also include a single-level or multiple-level sort order for the rows (issue records).

When you create a new query, AccuWork automatically starts off the results table with a couple of columns, including **Issue** (issue-number). You can add more columns at any time. You can also remove the **Issue** column from the table.

Selecting Columns to Appear in the Results Table

Use the **Columns** button in the Query Results toolbar to launch a window in which you select the columns to appear in the results table, and their order. (You can also change the column order in a results table using drag-and-drop — see below.)



Note that:

- The names that you work with in the Columns window are the official field-names defined in the issues database schema.
- The names that appear as column headers in the results table are the labels that appear on the database's edit form.

For some queries, you may want to include just a few important columns in the results table; for others (e.g. a complete data-dump), you may want to include all the fields defined in the issues database. Working in the Columns window, you can:

- Select a column label and use the  and  right-arrow buttons to move it between the **Selected** (appears in results table) and **Available** (does not appear in results table) lists.
- Select a column label in the **Available** list and use the  and  buttons to change the order of the columns.
- Use the  and  buttons to move all columns labels to the **Available** or **Selected** list.

When you're done, click **Ok to** apply the changes you've made, or click **Cancel** to discard the changes. You can further adjust the columns of a query results table using the techniques that work with all AccuRev GUI tables. See *Working with Tables* on page 4 of the *AccuRev User's Guide (GUI Edition)*.

All the changes you make, both in the Columns window and in the Query Results pane, are preserved when you invoke the **Save All** command.

It's likely that you won't be able to see the complete text strings entered into multiple-line text fields, no matter how wide you've made the results-table column. This limitation doesn't apply when you "print" the results table to an HTML file: the complete contents of each field is included in the HTML table. (See *Printing Query Results* on page 21.)

If you change the column layout in the Query Results pane after running a query, you don't need to rerun it — even if you add or delete columns. AccuWork automatically updates the results table in this case.

Working with the Records Listed in a Results Table

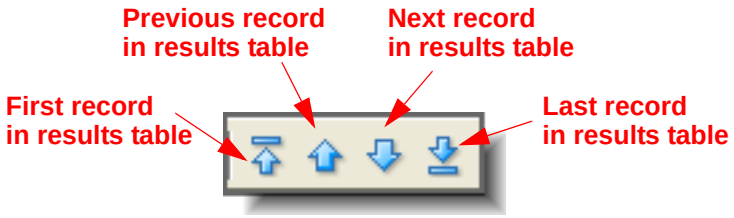
In many cases, browsing the results table produced by running a query may be all you need to do. But in other cases, you may want to see a selected issue record in the context of its edit form:

- The results table is read-only. To modify an issue record, you must access it through its edit form.
- That data you need may be in a field that is not included in the results table. Instead of adding a column to the results table, you can use the edit form to display all the fields.

Use either of the following techniques to retrieve the complete records listed in the results table:

- **Full-size edit form in a new tab** — select a row of the query results table and click the **Open Issue** button in the Query Results toolbar. (Alternatives: right-click the results-table row and select **Open Issue** from the context menu; or double-click the row in the results table.)

This opens a new tab containing When you open an issue record from a results table, browse arrows are enabled in the edit form's toolbar. This makes it easy to view, in the edit form, some or all of the records selected by a query.

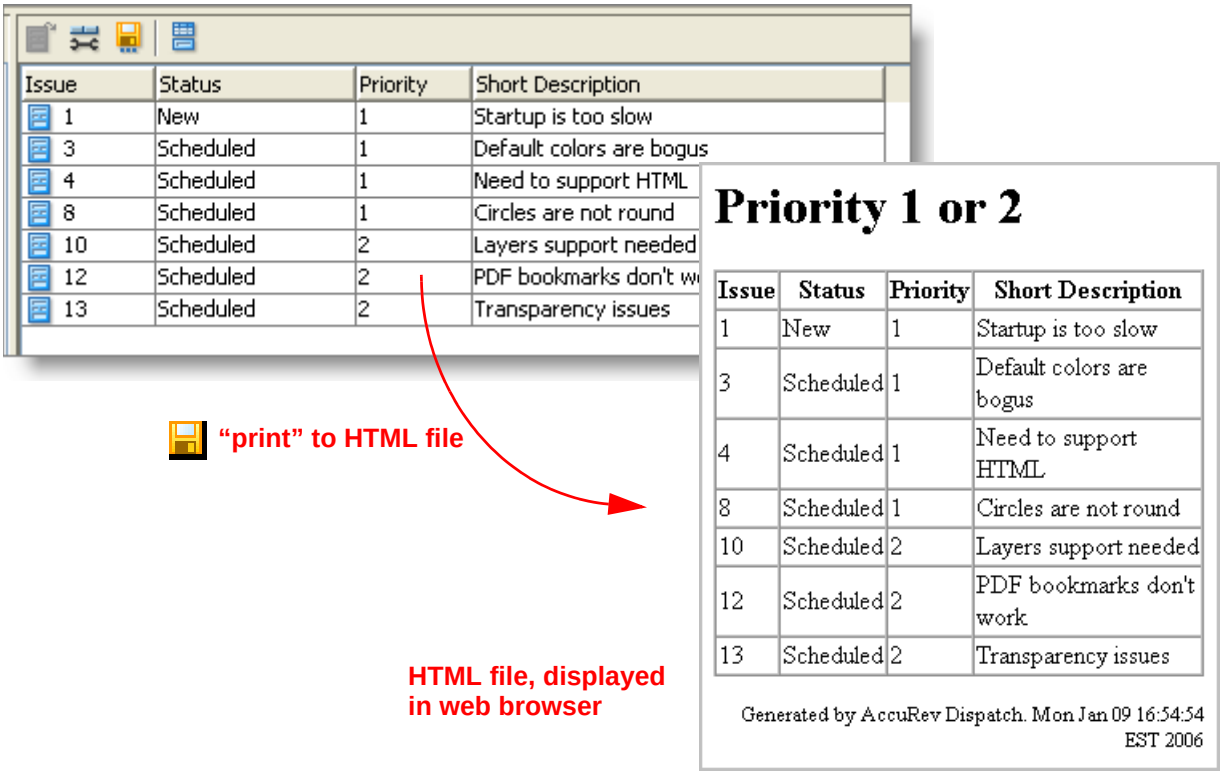


- **Reduced-size edit form pane in the Queries tab** — Click the **Show Issue Form** button in the Query Results pane toolbar. The edit form that appears is fully functional. As you select records in the query results table, they are automatically loaded into the edit form.

If you wish, modify the record as described in *Viewing and Modifying an Existing Issue Record* on page 6. After modifying a record, you may want to rerun the query in order to update the results table; this doesn't occur automatically.

Printing Query Results

Use the **Export Table** button on the Queries toolbar to save the current contents of the results table to an HTML-format file. The file contains an HTML table; each cell changes height and width when you view it with a Web browser and adjust the size of the browser window. This is particularly useful for viewing the contents of multiple-line text fields.

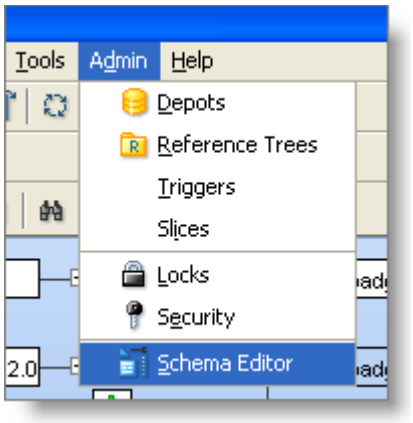


Designing Issues Databases and Edit Forms: The Schema Editor

This chapter describes how to use the AccuWork Schema Editor to define an AccuWork issues database, containing a collection of issue records. Each depot can contain one issues database, along with a custom-designed edit form, through which users create and modify the issue records. You can make the edit form "smart" by defining validations (edit checks) that specify default values, required fields, and interrelationships among multiple fields.

Invoking the Schema Editor

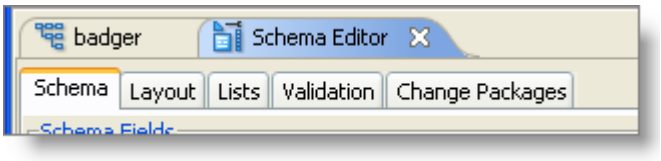
You perform all AccuWork database-design work on the Schema Editor tab, which you display using the **Admin > Schema Editor** command.



The first time you invoke this command in a particular depot, AccuWork offers to use the repository's default schema. Accepting this offer copies a set of XML-format configuration files from the **site_slice/dispatch/config** subdirectory to this depot.

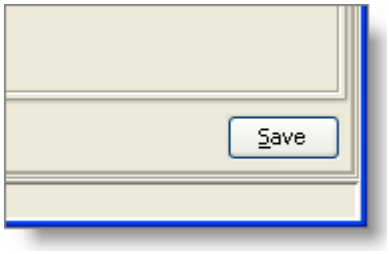
> Note: the default schema does not actually become the schema for this depot until you click the Schema Editor's **Save** button. See *Defining Database Fields* below.

The Schema Editor tab includes these subtabs:



- You define the AccuWork database's fields on the **Schema** subtab.
- You design the database's edit form on the **Layout** subtab.
- You define multiple-choice lists, each of which can constrain the values of one or more fields, on the **Lists** subtab.
- You define field validations, or edit checks, on the **Validation** subtab.
- You monitor and control certain aspects of the integration between AccuRev and AccuWork on the **Change Packages** subtab.

Saving Changes to the Schema

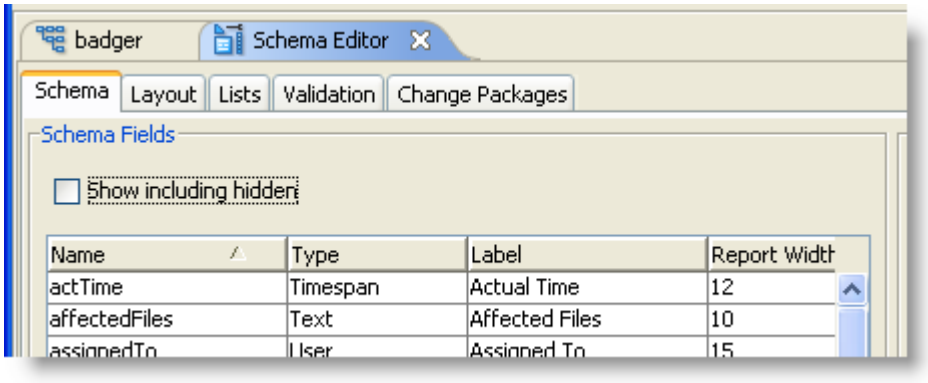


At any time while working in the Schema Editor, you can click the **Save** button in the lower right corner of the Schema Editor tab. This saves the current state of the schema to a set of XML-format files in subdirectory **dispatch/config** of the depot directory (slice) in the AccuRev repository:

- Contents of the **Schema** subtab: **schema.xml**
- Contents of the **Layout** subtab: **layout.xml**
- Contents of the **Lists** subtab: **lists.xml**
- Contents of the **Validation** subtab: **logic.xml**
- Contents of the Change Package Results section of the **Change Packages** subtab: **cpk_fields.xml**
- Contents of the Change Package Triggers section of the **Change Packages** subtab: **cpk_promote_queries.xml**

Defining Database Fields

To begin designing an issues database, open a Schema Editor tab and go to the Schema subtab. (This happens automatically if you accept AccuWork's offer to use the repository's default schema.)



If you are not using the default schema, AccuWork initializes the Schema Fields table with two fields.

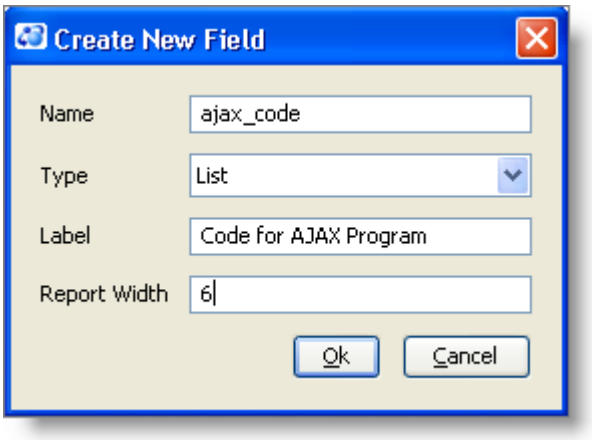
- **issueNum**: An integer-valued field that records the position of the issue record in the depot's issues database. This number is assigned when a user creates the record (i.e. at the first **Save** on the edit form), and it never changes.
- **transNum**: An integer-valued field that records the transaction number of the most recent update to the issue record. The transaction resides in the depot's transaction log — the same log that records AccuRev transactions, such as **keep** and **promote**. The type of the issue-record-update transaction is **dispatch**.

Note that a record's **issueNum** value never changes, but its **transNum** value changes each time a user **Save**s the record. The integration between AccuRev and AccuWork also updates issue records, and so changes the value of the **transNum** field. Also note that **issueNum** and **transNum** are indexes into two different databases.

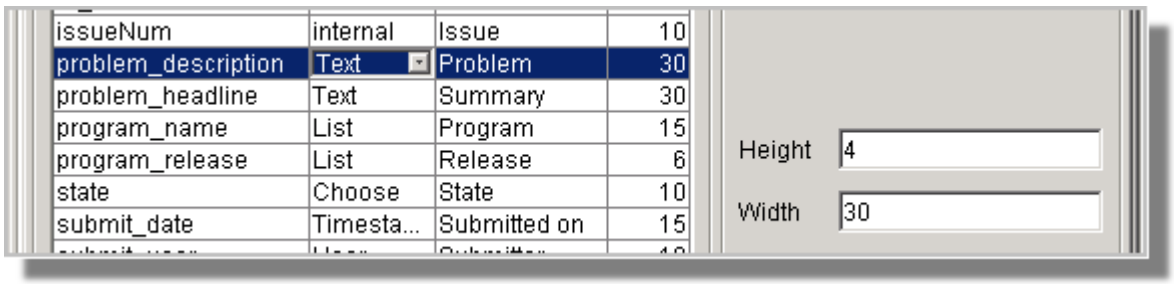
Adding and Removing Fields from the Database Schema

You can define any number of additional fields in the issues database schema. Follow these steps for each new field:

1. Click the **Add** button at the bottom of the Schema subtab.
2. Fill in the **Create New Field** window that appears:



   - **Name**: The official *field-name* of the field. Users of the AccuWork database don't ever need to know this name — they know the field by its label. The field-name must not contain any SPACE characters. Validations must be expressed in terms of the field's name, not its label.
   - **Type**: One of the *data types* supported by AccuWork. See *Data Types* below.
   - **Label**: The field-label character string that identifies the field on the database's edit form. A field-label can contain SPACE characters (e.g. **Last Name**).
   - **Report Width**: An integer that determines the relative width of the field in the HTML table created when the user clicks **Export** on the edit form of an individual issue record.
3. Click **OK** to close the **Create New Field** window.
4. In the Field Values box to the right of the Schema Fields table, specify additional information about the field. The kind of information required varies with the data type (see *Data Types* below).



Repeat the steps above as often as required to create new fields in the AccuWork database.

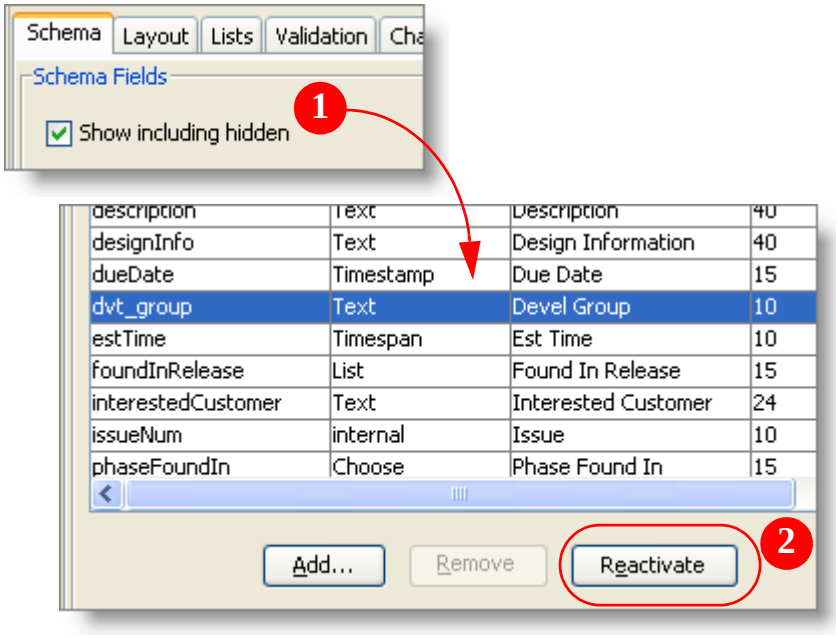
Note: your field definitions are not saved until you click the **Save** button in the lower-right corner of the Schema Editor tab. You cannot save your work until you place at least one field in the database's edit form (Layout subtab).

To remove an existing field (except for **issueNum** and **transNum**, as noted above), select it and click the **Remove** button. The field disappears from the Schema tab and can no longer be used in the edit form (see *Designing an Edit Form* on page 28). But any data stored in existing issue records is preserved.

Note: when you remove a field from the schema., AccuWork automatically checks whether the field is used in the issue database's edit form. If so, it removes the field from the edit form

You can restore a removed field to the database schema:

1. Check the **Show Including Hidden** checkbox. All removed fields appear in the list, with a gray background.
2. Select the field to be restored, and click the **Reactivate** button.



Integrating Configuration Management and Issue Management: the 'affectedFiles' Field and Change Packages

If you wish to enable the integration of a depot's version-controlled files and its AccuWork issue records, define a database field whose name is **affectedFiles**. The field's type must be "text"; see *Data Types* below. You can choose any label for the field. (Such a definition is included in the default AccuWork database schema.)

The integration also depends on the enabling of a built-in AccuRev trigger procedure:

```
accurev mktrig -p <depot-name> pre-promote-trig client_dispatch_promote
```

The integration routine writes the transaction number of each **promote** command to the **affectedFiles** field of a particular issue record. Alternatively, in the AccuRev Enterprise version of AccuWork, the integration routine records each promoted version in the issue record, in a special section named **Changes**. This section is maintained automatically by AccuWork — you don't need to define any database fields to enable this additional aspect of the integration.

For details, see *Integrations with AccuWork* on page 154 of the *AccuRev User's Manual (CLI)*, and *Change Packages and Integrations between Configuration Management and Issue Management* on page 43.

Data Types

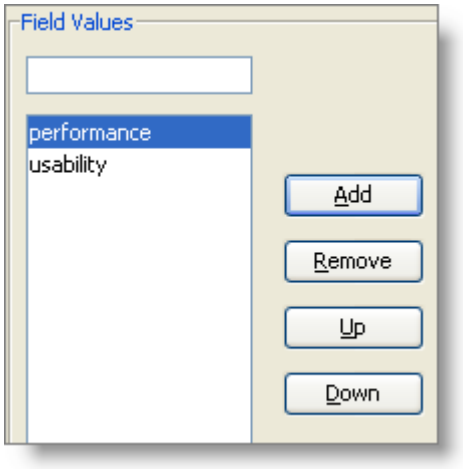
Each field you define in the **Create New Field** window must have one of the AccuWork data types listed in the table below.

Field Type	Possible Values	Required Additional Information in "Field Values" Box
text	Any character string. For multiple-line fields, the string can include line-terminators.	**Height**: number of lines displayed for the field in the edit form (default: 1). For multiple-line fields (height > 1), the edit form includes an expand/contract button to increase/decrease the number of lines displayed. **Width**: relative width of field in the edit form.
choose	One of the strings specified in the Field Values box for this field.	A set of strings. In the edit form, a list-box containing this set of strings is offered to the user.
timestamp	An AccuRev timestamp.	Granularity (year, month, day, hour, minute, or second). In the edit form, the user fills in fields that indicate a time, to the granularity you specify here.
user	One of the principal-names in the user registry maintained by the AccuRev server.	None. In the edit form, a list-box containing all principal-names is offered to the user.
timespan	A numeric value, indicating a number of hours. Decimal values (e.g. 4.5) are allowed.	None.
list	One of the strings specified in the definition of a particular named list.	The name of an existing list, defined on the **Lists** subtab. (See *Defining Multiple-Choice Lists* below.) In the edit form, a list-box containing the set of strings defined in the named list is offered to the user.
log	Any character string (variant of text field type)	In the edit form, an Add Text control appears above the input field. The user can type directly into the input field, or can click the Add Text control and create a "log entry" in the popup window that appears.
attachments	A set of attachment definitions.	**Height**, **Width**: the height (number of lines) and width (approximate number of characters) of the edit-form field that lists the attachment definitions.
internal	A positive integer.	None. You cannot create a field with this data type; it is used only by the built-in fields **issueNum** and **transNum**.

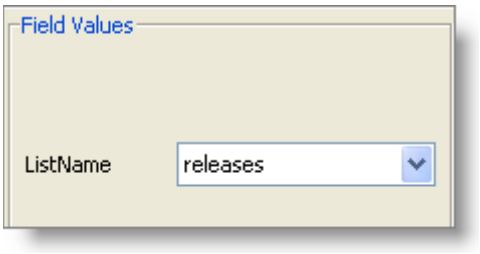
Defining Multiple-Choice Lists

The **choose** and **list** data types are similar:

- For a **choose** field, the set of possible values — an ordered list of strings — is part of the individual field definition. You enter the possible-values list in the Field Values box for that field.



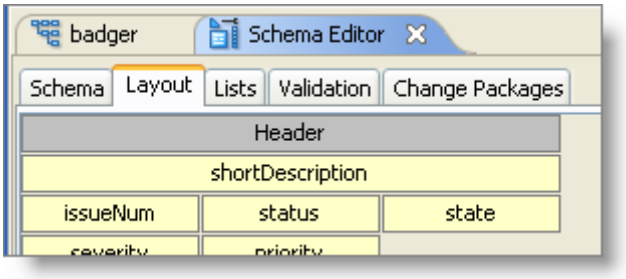
- For a **list** field, the set of possible values is also an ordered list, but it is *not* part of the individual field definition. In the Field Values box, you specify the name of one of the lists created on the Lists subtab.Any number of **list** fields in a AccuWork database can use the same named list:



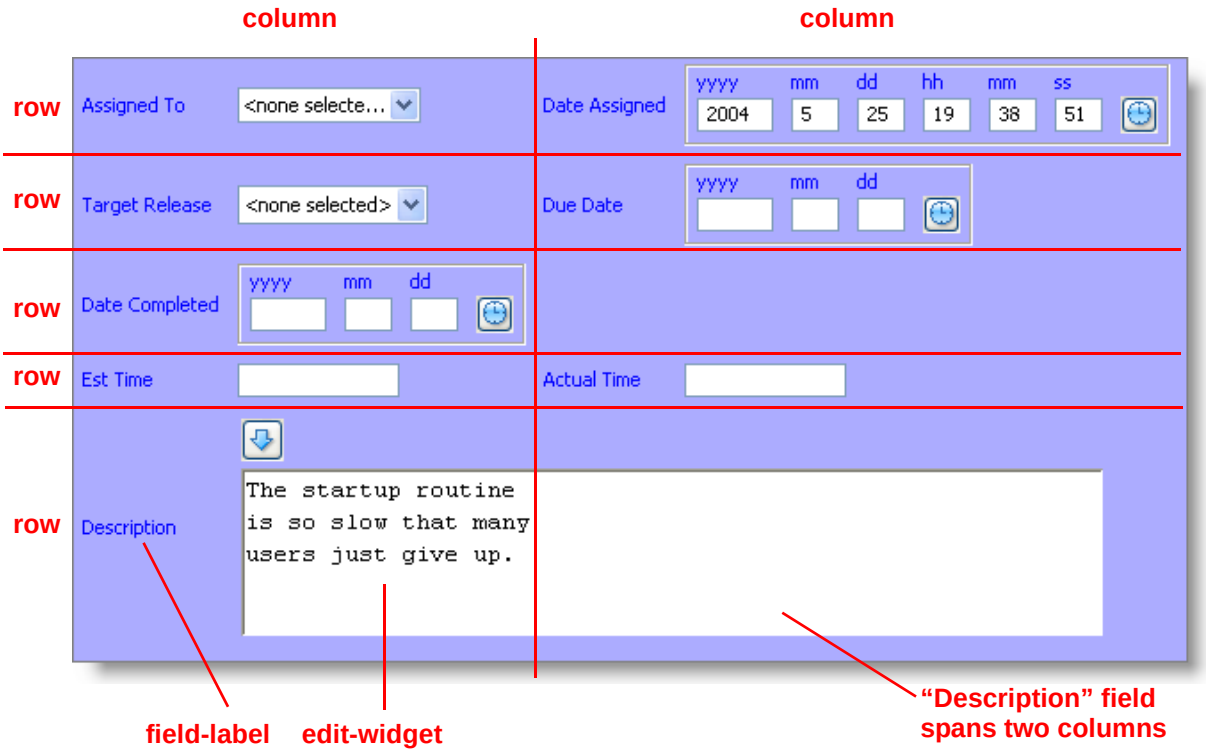
The mechanics of defining the ordered list is similar in the “local” case (for an individual **choose** field) and in the “global” case (on the Lists subtab, for use by any number of **list** fields). On the Lists subtab, you must supply a ListName for the list; for an individual **choose** field, the possible-values list doesn’t need or have a name.

Designing an Edit Form

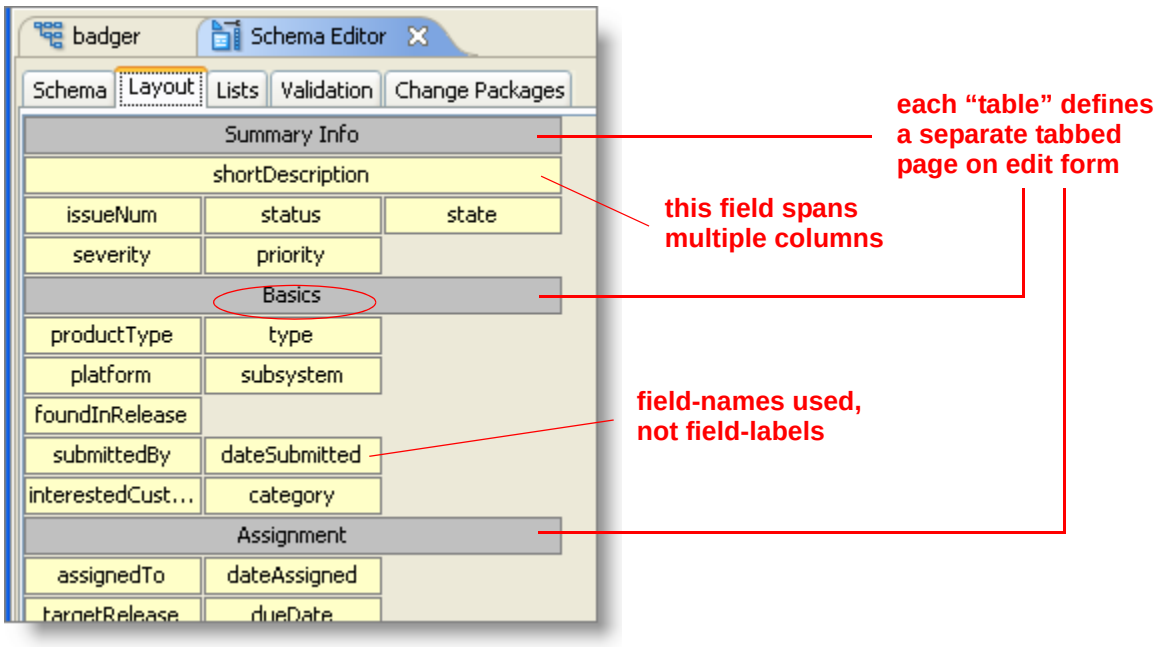
You design the edit form for a AccuWork database on the Layout subtab.



An edit form consists of field-labels and corresponding edit-widgets, arranged in rows and columns. The field-labels come from the Labels column of the Schema Fields table (Schema subtab). The edit-widget for a field depends on its data type and, in some cases, on the additional Field Value information.

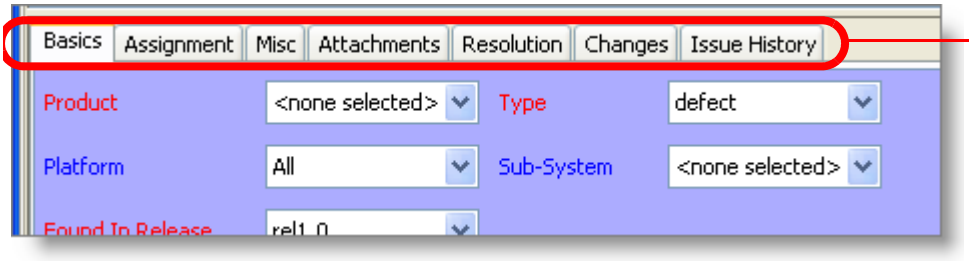


The Layout subtab provides a drag-and-drop canvas on which you design the edit form's rows and columns. On this canvas, each field-label/edit-widget pair is represented by a yellow box. The following screen shot shows the layout that defines the edit form illustrated above:

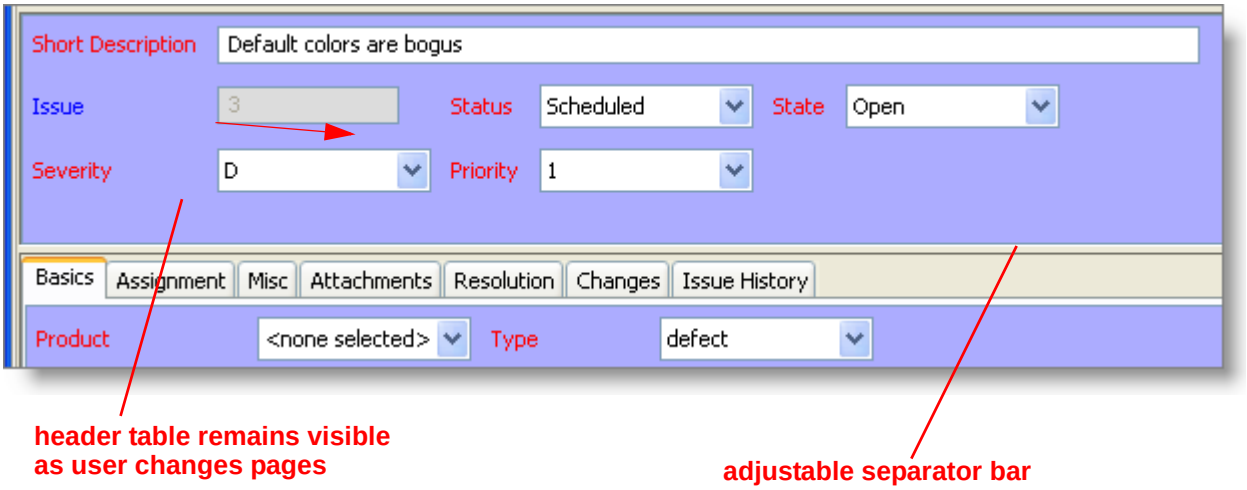


As the screen-shot annotations indicate, the yellow box contains a field-name, not a field-label. On the edit form, the field-label and the corresponding edit-widget will appear, side by side, at this position. You can also expand a yellow box to make it span two or more columns.

You can organize the fields into multiple “tables”, each of which defines a separate tabbed page in the edit form.



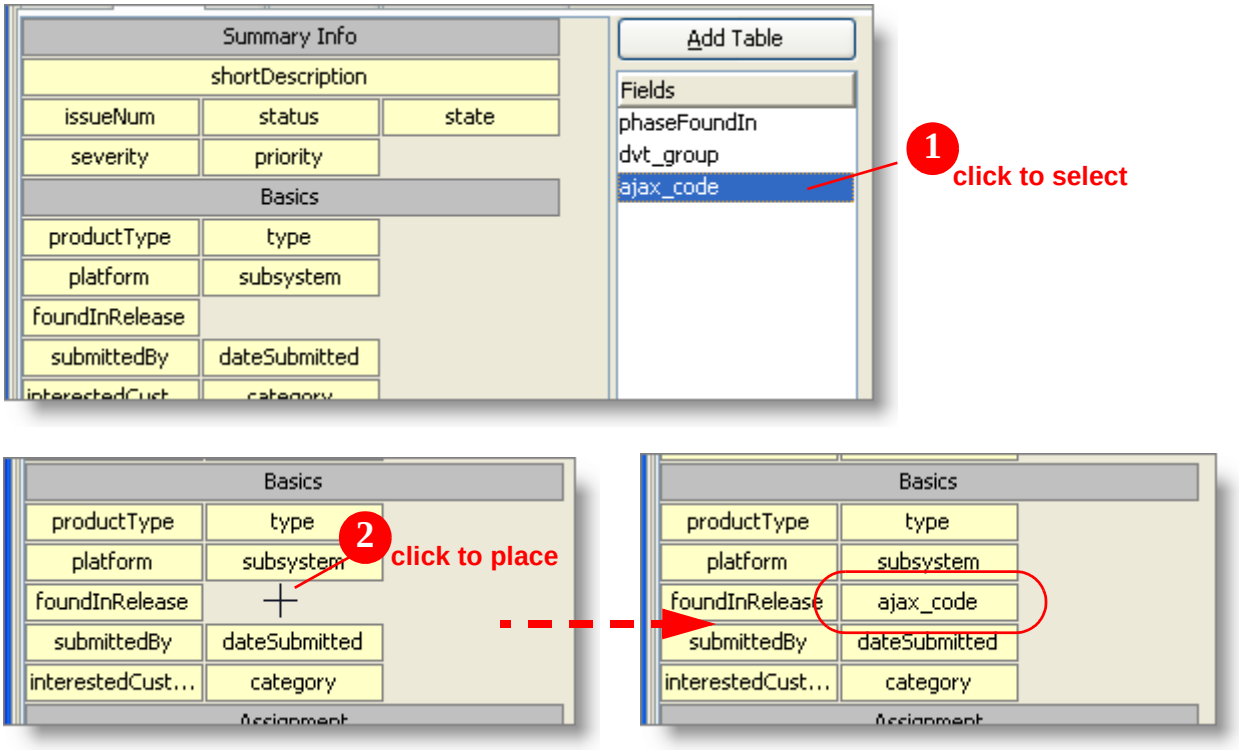
One of the tables can be a header table. Instead of defining a separate tabbed page, a header table defines a set of fields that always appear on the edit form, no matter which tabbed page is currently visible.



Form Layout Operations

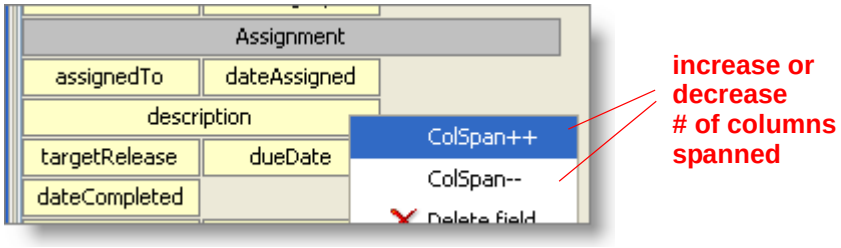
This section describes how to perform the various edit-form design operations on the Layout subtab.

- **Adding a field to the edit form**: The **Fields** list-box offers all fields that do not currently appear in the edit form. Click to select a field in this list-box. Then, click at the empty location on the canvas where you want to place the field.



- **Arranging fields into rows and columns**: Drag-and-drop the yellow boxes to the desired location on the canvas. Drop a box on top of another one to push it rightward or downward.

- **Changing column-spanning**: Right-click a yellow box, and select one of the spanning operations from the context menu.

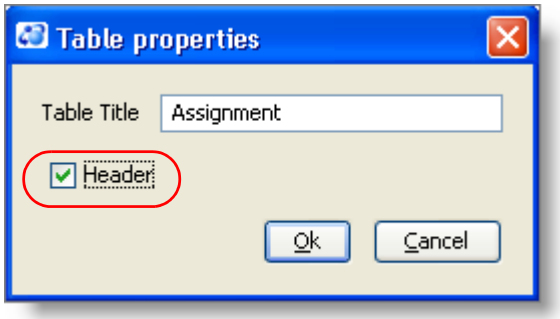


- **Creating a new page in the edit form**: Click the **Add Table** button above the **Fields** list-box. Fill in the Table Title field in the dialog box that appears, then click **OK**. This title appears on the page's tab in the edit form.

  A gray box with the specified title appears at the bottom of the layout. Fields that you place below this box will appear on the new page of the edit form.

- **Creating a header table for the edit form**: When creating a new page for the edit form (see above), check the **Header** checkbox if you want it to be the header table. (If another table is currently selected as the header table, it is automatically deselected.)

  You can also redefine an existing page to be a header table: right-click the gray box and select



Properties. All fields in the header table will appear at the top of the database's edit form, above every tabbed page.

Remember that an edit form can have at most one header table. Its fields always appear at the top of the edit form, even if it is not the topmost table on the Layout tab.

- **Renaming a page**: Right-click on the gray box, select **Properties**, and change the entry in the Table Title box.
- **Arranging fields into multiple pages**: Drag-and-drop the yellow boxes (fields) and the gray boxes (pages). Fields that you place below a gray box will appear on the corresponding page of the edit form.
- **Removing a field or page**: Right-click the yellow box (field) or gray box (page), and select **Delete** from the context menu. Deleting a page doesn't delete the fields currently on the page — it just deletes the gray box, effectively merging the fields into the page above.

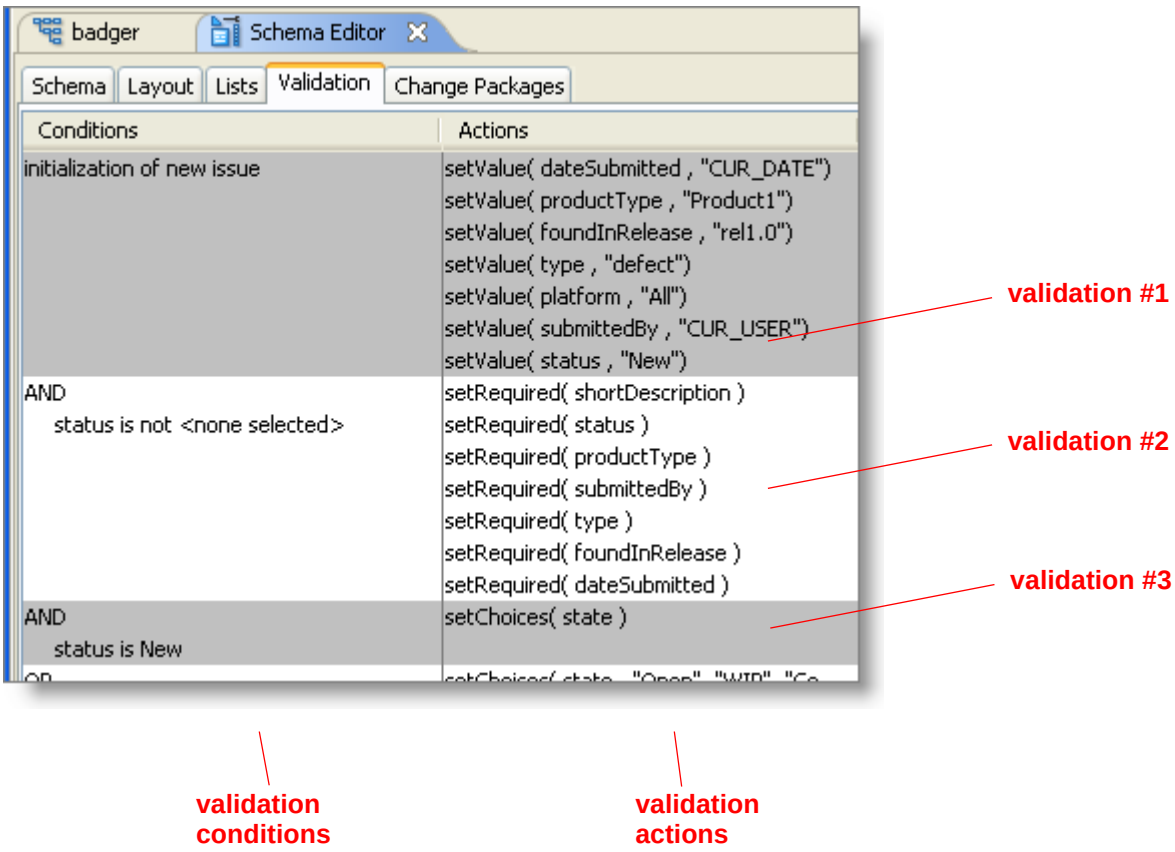
  Note: you cannot delete the first gray box; an edit form must have at least one page!
- **Setting the edit form's colors**: Click one of the colored buttons at the right side of the Layout subtab. Then select a color from the **Pick a color** window. You can set these colors:
  - **Foreground**: the color of the edit form's field-labels.
  - **Required Foreground**: the field-label color for fields where a value must be specified. Required fields are defined on the Validations subtab. See *Defining Edit Form Validations* on page 32.
  - **Background**: the color of the edit form's background.

  Note: The "Foreground" and "Background" settings are currently not used.

Defining Edit Form Validations

Users of a AccuWork database create and modify issue records through the database's edit form. To increase the efficiency and accuracy of this process, you can create a set of validations to be applied as the user works with the edit form. (Validations are sometimes called "edit checks".)

You create and maintain the set of validations using a point-and-click interface. AccuWork displays the current validations in tabular format; each one takes the form "if a certain condition is true, then perform a particular set of actions".



The following kinds of validations are available:

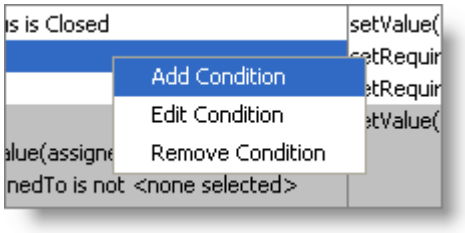
- Initialize a field in a new issue record with a specified value.
- Require a value to be entered in a field.
- Based on a condition, make one or more of these changes:
  - Set a **text** field to a specified value.
  - Set a **user** field to a particular value — for example, to the current user.
  - Set a **timestamp** field to a particular value — for example, to the current time.
  - Make one or more fields into required fields.
  - Change the possible-values list for a **choose** field.
  - Set the entire issue record, a particular page (tab) of the issue record, or a particular field to be read-only.

  The condition you specify is essentially the same as a AccuWork query; if the condition is true for (“selects”) the current record, the specified changes are applied to the edit form.

Initializing Field Values in a New Issue Record

The first entry in the table of validations is special. Its condition is always “initialization of new issue”, so that its actions are performed exactly once: between the time the user invokes the **New Issue** command and the time the new edit form appears.

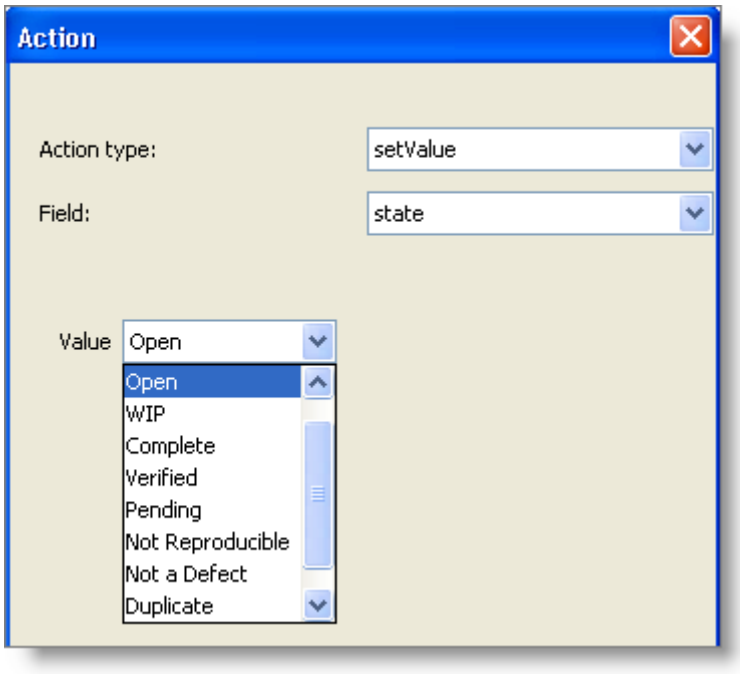
To define an action, right-click in the **Actions** column, then select **Add Action** from the context menu.



An **Action** window appears, in which you define the action. For initialization of a new issue record, the only appropriate action type is **setValue**.

> Note: some releases of AccuWork allow you to include **setRequired** and **setChoices** actions in a record initialization. Such actions will be ignored when a new record is initialized.

After setting the action type to **setValue**, select a field to be initialized and specify the initial value. The **Value** edit-widget adjusts to the selected field. In this example, **state** is a “choose” field, so the list-box offers the field’s predefined choices as the initial value.



Conditional Validations

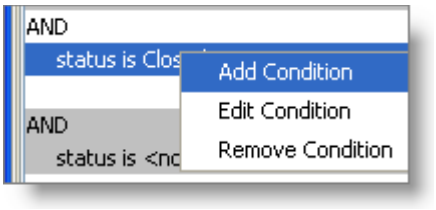
The setting of initial field values is an unconditional validation: it happens every time a **New Issue** command is invoked. All other validations are conditional: a certain set of actions are performed if, and only if, a certain condition is met.

The unconditional setting of initial field values occurs just once; but the conditional validations are performed repeatedly: when an edit form first appears *and* each time the user changes any field value. Each repeat involves:

- Clearing the “required” status of all fields.

- Performing all validations (except for that first one: “initialization of new issue”). If a validation’s condition is true, the corresponding actions are invoked.

Specifying the Condition

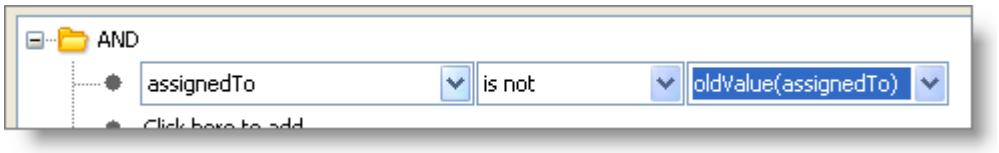


To create a new conditional validation — that is, a new shaded or unshaded row in the table — right-click anywhere in the Conditions column, then select **Add Condition** from the context menu. Then proceed to construct the “if” clause, as described above.

The same context menu provides commands for revising the “if” clause of an existing conditional validation, and for removing a conditional validation.

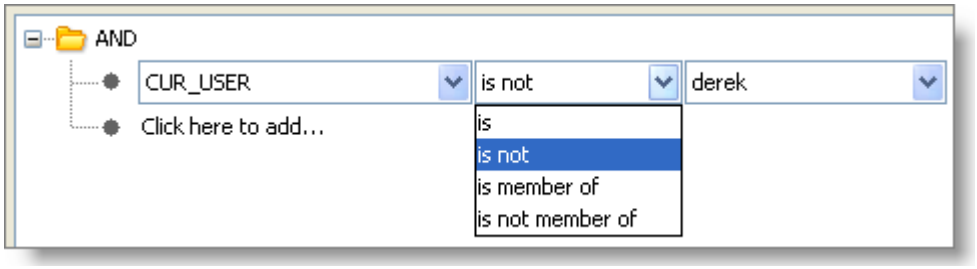
Specifying the condition itself — the “if clause” — of a conditional validation is very much like specifying a AccuWork database query. (Queries are described in section *Using AccuWork Queries* on page 8.) But there are some differences: in a validation condition, you can:

- Compare the current value of a field with its “old” value — that is, compare the value currently displayed for the field vs. the value stored in the AccuWork database by the most recent **Save**.



- Compare the (current or old) value of a field with the (current or old) value of another field in the same record.

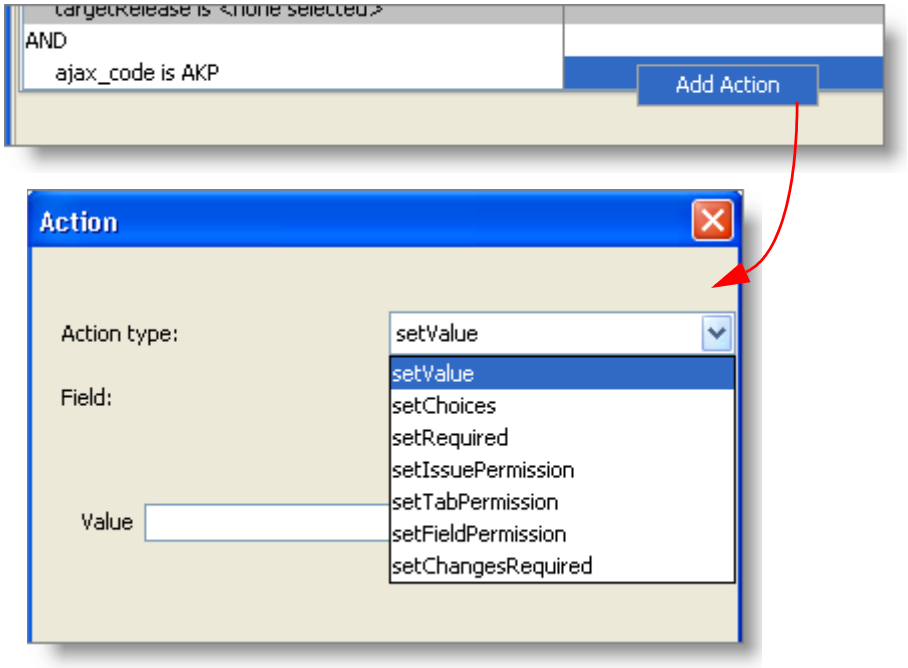
- Test the AccuRev user identity or group membership of the person using the edit form (CUR_USER).



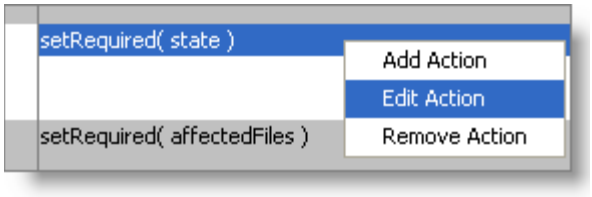
Query conditions cannot make such field-to-field comparisons; they can only compare field values to literal values.

Specifying the Actions

Each conditional validation can include any number of actions. You create an action by right-clicking in the Actions column of the validation, and selecting **Add Action** from the context menu. This displays a window in which you define the action.



After you've created one or more actions for a conditional validation, you can use the same context menu to revise or delete individual actions.



The following sections describe the actions that you can define to be performed if the validation condition is true:

- *Setting a Field Value* (**setValue**)
- *Revising the Choices for a "choose" Field* (**setChoices**)
- *Requiring a Value to be Entered in a Field* (**setRequired**)
- *Setting Permissions on All or Part of the Issue Record* (**setIssuePermission**, **setTabPermission**, **setFieldPermission**)
- *Requiring Change Set Entries* (**setChangesRequired**)

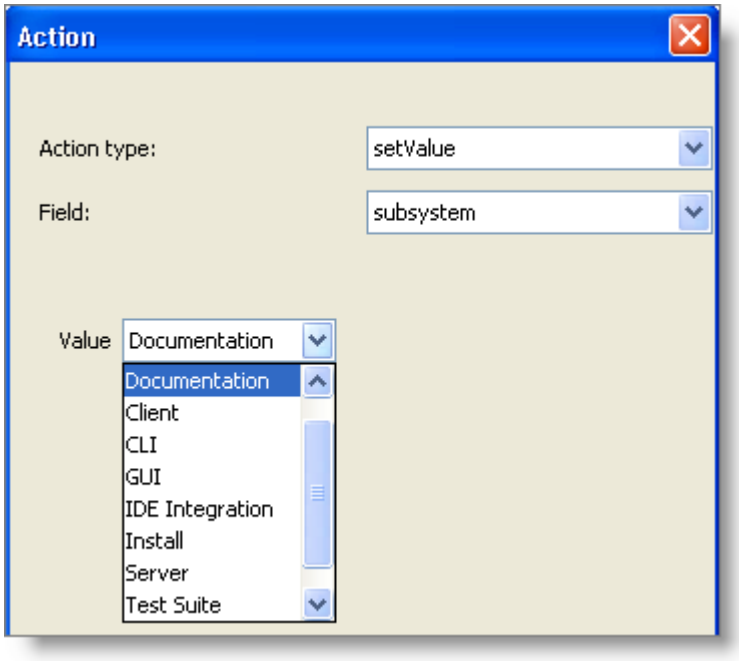
Each validation can invoke any number of actions, of any type.

Setting a Field Value

In addition to initializing field values (see *Initializing Field Values in a New Issue Record* on page 34), you can set the values of fields while the user is working with the record, based on certain conditions. For example, if the user enters **ColorStar** in the **program_name** field, a validation could automatically set the **fix_priority** field to **high**. (Management has mandated rapid improvement in the robustness of the ColorStar application.)

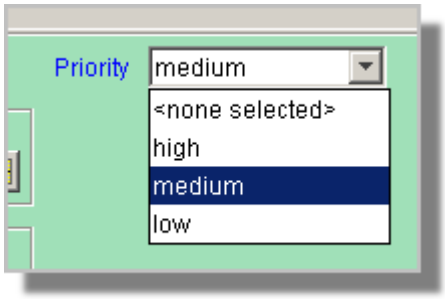
To define a **setValue** action for a validation condition:

1. In the Action window, select **setValue** as the action type, and select the field to be set.
2. The **Value** edit-widget adjusts to the selected field, just as it does when you're initializing a field in a new issue record.



For a "text", "log", or "timespan" field, you enter a text string as the value; for a "date" field, you specify a date in the standard way; for other field types, you use a list-box to specify any of the field's predefined values.

Revising the Choices for a “choose” Field



The definition of each field of type “choose” includes an ordered list of strings. The user fills in this field by choosing one of the strings from a list-box.

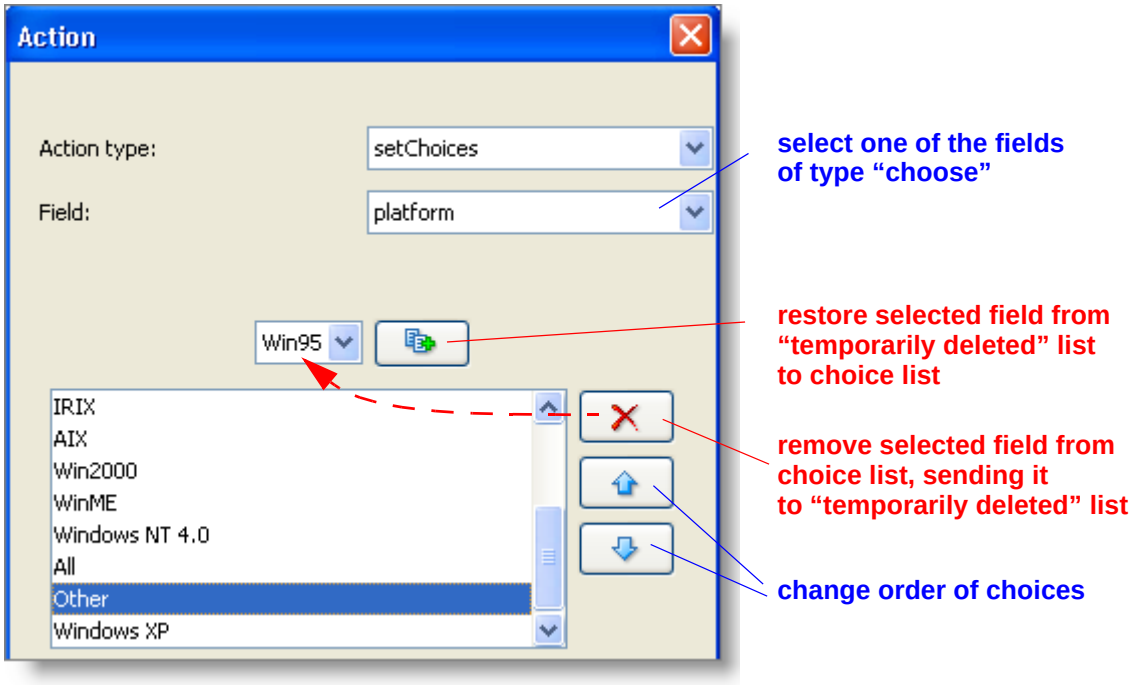
In a validation, a **setChoices** action can change the “choice list” that the user will see when he opens the list-box: The changes to the choice list can include:

- Removing one or more of the existing choices
- Changing the order of the choices

You cannot add new strings to the choice list with a **setChoices** action. The changes made by a **setChoices** action are temporary: they last only until the next **setChoices** action on the same field, or until the user closes the edit form. When an edit form is opened on another issue record (or subsequently on this issue record), the user will see the field’s original choice list, as defined on the **Schema** tab.

To define a **setChoices** action for a validation condition:

1. In the Action window, select **setChoices** as the action type, and use the **Field** list-box to select one of the database’s fields of type “choose”.
2. Use the controls at the bottom of the **Action** window to define the temporary change to the choice list:



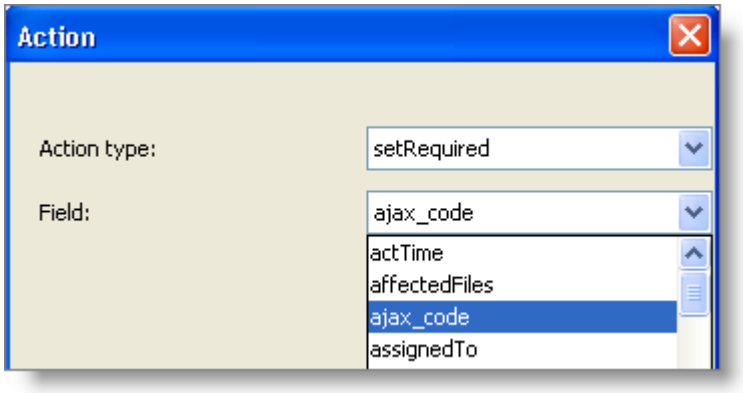
Requiring a Value to be Entered in a Field

Like many other programs that process fill-in-the-blanks forms, AccuWork can treat certain fields as required fields — the user must specify a value in each such field before AccuWork will create or update an issue record. (For a “choose”, “list”, or “user” field, the value must not be the **<none selected>** placeholder.)

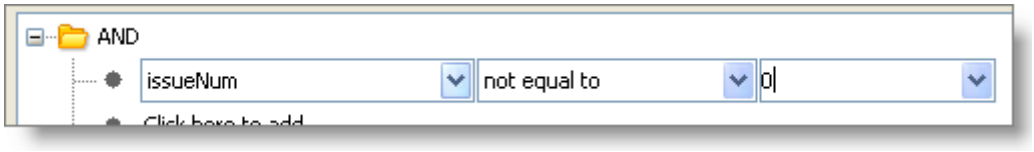
AccuWork affords you great flexibility in controlling required fields. A field’s “required” status is not part of the basic database schema. Instead, it is controlled by the conditional-validation facility. Thus, AccuWork repeatedly redecides whether a field is required: when the edit form first appears *and* each time the user changes any field value on the edit form. Whenever the user clicks the **Save** button, AccuWork uses the current set of required fields to allow or disallow the “save” operation.

To define a **setRequired** action for a validation condition:

1. In the Action window, select **setRequired** as the action type.
2. Select the field to be required.



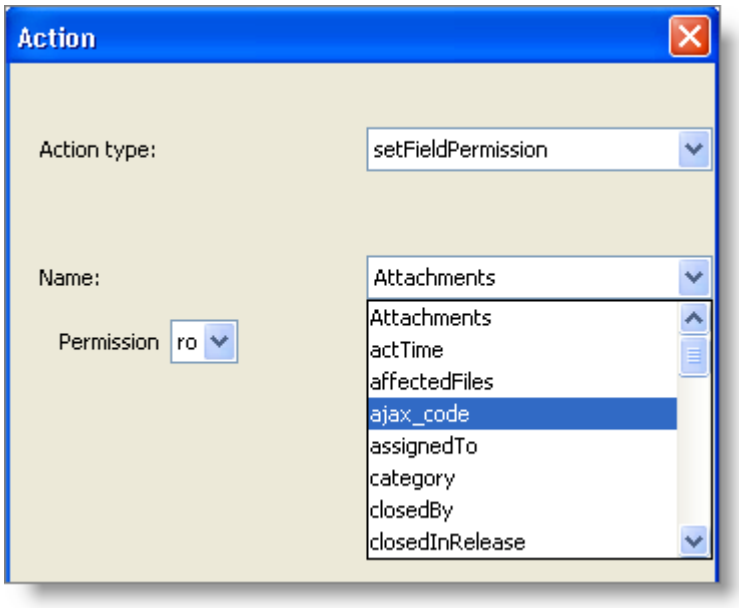
If you want a field to be required in all circumstances, use the **setRequired** action along with a condition that’s always true. For example, you can exploit the fact that every issue record has a positive integer as its issue number:



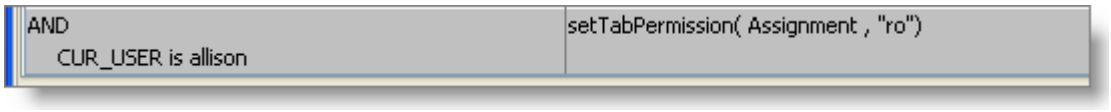
Setting Permissions on All or Part of the Issue Record

The **setIssuePermission**, **setTabPermission**, and **setFieldPermission** actions are essentially similar: they restrict the editability of some component of the issue record:

- **setIssuePermission** makes the entire issue record read-only.
- **setTabPermission** makes a particular page (tab) of the issue record read-only. See *Designing an Edit Form* on page 28.
- **setFieldPermission** makes a particular field of the issue record read-only.



The "read-only" (**ro**) permission is the only one you can set with these commands. This setting affects the user's current access to the issue record; the "read-only" status is not stored in the repository as part of any issue record. For example, user **allison** might find that she cannot change any fields on the Assignment page of any issue record, because of this validation:



This restriction affect **allison**'s access to issue records; other users' access remains unaffected. And if this conditional validation is subsequently removed, **allison** will regain access to the Assignment page of all the issue records.

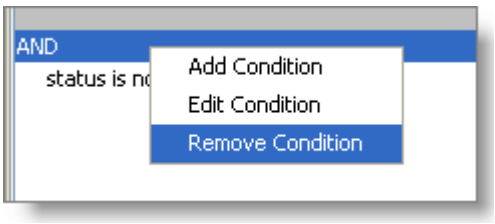
Requiring Change Set Entries

The **setChangesRequired** action specifies that the user cannot **Save** an issue unless there is at least one entry in the issue record's change set (Changes tab).

> Note: be careful *not* to specify this action with the "initialization of new issue" condition. See *Initializing Field Values in a New Issue Record* on page 34.

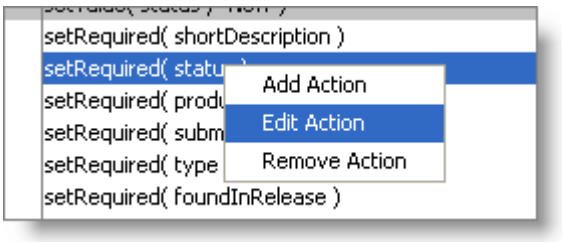
Revising and Removing Validations and Actions

Each conditional validation consists of a condition (left column) and a set of actions (right column). To revise or remove an entire validation, right-click anywhere within the condition, and select the appropriate command, **Edit Condition** or **Remove Condition**, from the context menu.



For the first (unconditional) validation, "initialization of new issue", you cannot remove or revise the condition. You can only work with the validation's actions.

To revise or remove an individual action from a validation, right-click on that action and select the appropriate command, **Edit Action** or **Remove Action**, from the context menu.



The Change Packages Tab

The AccuRev Enterprise version of AccuWork supports an enhanced integration between AccuRev and AccuWork: any AccuWork issue record can act as a change package, keeping track of which versions were created to fix the bug (or implement the new feature) described in that issue record.

In the Schema Editor, the Change Packages subtab provides controls for certain aspects of this facility. For details, see *Viewing Stream Contents/Differences in Terms of Change Packages* on page 51.

Change Packages and Integrations between Configuration Management and Issue Management

Any version-control system must be able to keep track of the changes that developers make to individual files. A full-fledged configuration management system, like AccuRev, should be able to handle questions like these:

> "What were all the changes made to source files in order to fix bug #457?"
>
> "Have all the changes made to fix bug #457 been handed off to the QA Group" (That is, have the appropriate versions been promoted to the QA stream?)

AccuRev can handle such questions through its change package facility. (Change packages are available in the AccuRev Enterprise product only.)

Structure of a Change Package

A change package is a collection of element versions; for example:

> version **kestrel_dvt_jjp/13** of element **/./src/brass.c**
> version **kestrel_dvt_jjp/14** of element **/./src/brass.h**
> version **kestrel_dvt_jjp/16** of element **/./src/commands.c**

The basic idea is that this set (or "package") of versions contains all the changes required to implement a certain development project. But we need to refine this idea. Consider that version 14 of **brass.h** probably contains *more* than just the changes for that development project. For example:

- Versions 1-7 might have been created years ago, when the product was first developed
- Versions 8 and 9 might have been minor tweaks, performed last month
- Versions 10-14 are the only versions with changes for the development project in question

So we need a way to express the idea that only the "recent changes" to **brass.h**, those in versions 10-14, are to be included in the change package. AccuRev accomplishes this by defining each change package entry using two versions: a user-specified head version and an older, automatically-determined basis version. The "recent changes" to be included in the change package were made by starting with the basis version (version 9 in this example) and **Keep**'ing one or more new versions (versions 10, 11, 12, 13, and 14 in this example).

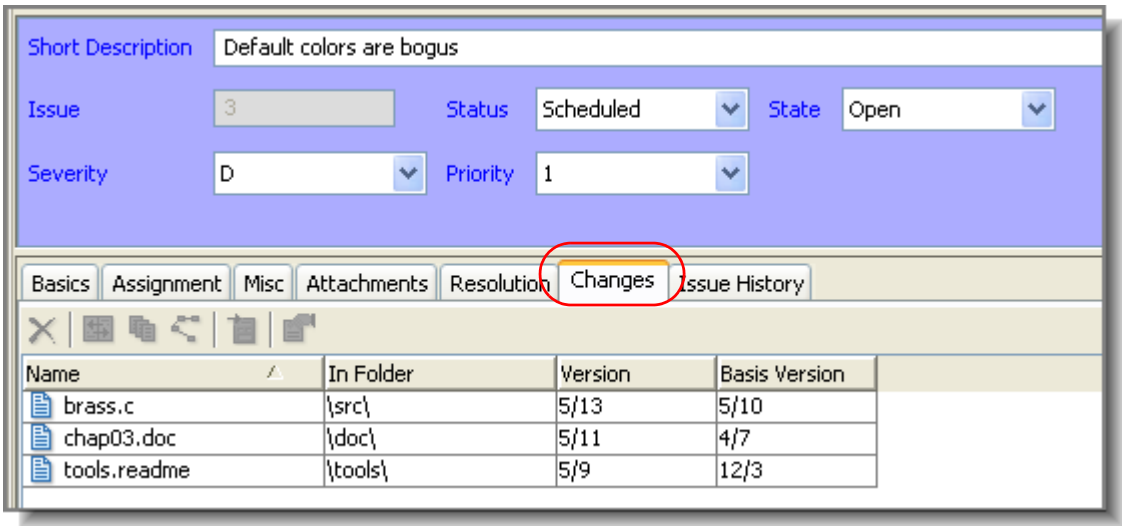
In the AccuRev GUI, the head version of a change package entry is usually identified simply as the "Version".

> Note: the **Patch** command uses the same "recent changes" analysis to determine which changes in the "from" version are to be incorporated into the "to" version.

Where should the change package entry for **brass.h** be recorded? AccuRev already provides a mechanism for keeping track of development projects: the AccuWork issue-management facility.

Each project — fixing a bug, creating a new feature, etc. — is tracked by a particular AccuWork issue record. So it makes sense to implement change packages using issue records.

Each issue record includes a **Changes** section that acts as an "accumulator" for versions' changes. Here's how the above example of a change package would appear in an issue record's edit form:

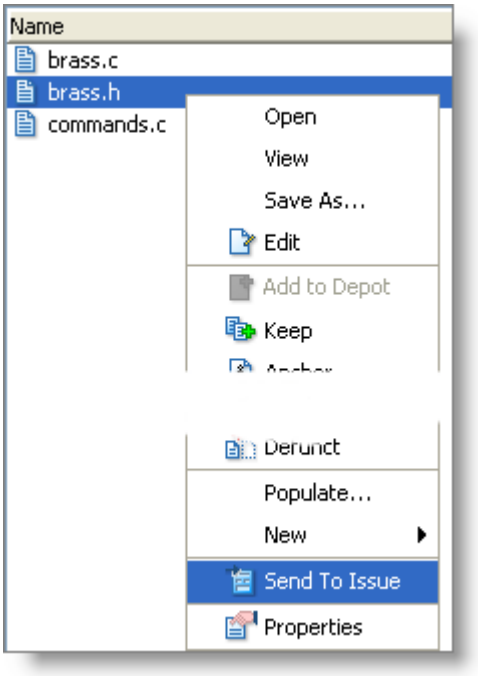


This change package has entries for three elements:

- **brass.c**: The basis version, 5/10 was created in the user's own workspace. This indicates that the user promoted 5/10 to the backing stream. AccuRev assumes that this change was for another task, not the one covered by this issue record. Then, the user turned his attention to the current task, creating additional versions up to and including 5/13.
- **chap03.doc**: This change began when the user updated his workspace, bringing in version 4/7 of the element (which had originally been created in another workspace, then was promoted to the backing stream). Then, the user created one or more versions in his own workspace, up to and including version 5/11.
- **tools.readme**: Similarly, this change began when the user updated his workspace, bringing in version 12/3, originally created in another workspace. The user created one or more versions in his workspace, ending with version 5/9.

Each change package can include at most one entry for a given element. This rule helps to ensure that the changes in a given change package are consistent with each other. See *Updating Change Package Entries* on page 48.

Creating Change Package Entries



You can add entries to a change package manually: right-click a version in the File Browser, Version Browser, or History Browser, and then select the **Send to Issue** command from the context menu. The selected version becomes the head version of the change package entry; AccuRev automatically determines the corresponding basis version. As the examples above suggest, AccuRev uses an algorithm that determines the set of "recent changes" to the element, made in a single workspace.

In the Version Browser, a variant command, **Send to Issue (specifying basis)**, enables you to pick the basis version, rather than allowing AccuRev to determine it automatically.

You can also invoke the **Send to Issue** command on the Changes tab of an issue record. This copies an existing change package entry to a different change package (issue record).

AccuRev can record change package entries automatically, whenever the **Promote** command is invoked in a workspace. For example, suppose issue record #3 represents a particular bug (and its fix). Whenever a developer promotes one or more versions whose changes address that bug, he specifies issue #3 at a prompt. AccuRev automatically creates a change package entry in issue #3 for each promoted version.

Automatic recording of change package entries is enabled through an integration between configuration management and issue management. This change-package-level integration is separate from the transaction-level integration that has long been an AccuRev feature. For more on both these integrations, see *Integrations Between Configuration Management and Issue Management* on page 54.

Complex Change Package Entries

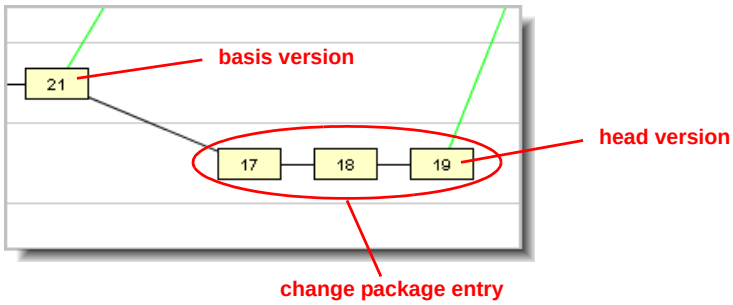
All change package entries are recorded in terms of real versions (those created in users' workspaces), even though there may be corresponding virtual versions (created by promoting the real versions from workspaces to dynamic streams). In all the examples shown above, each change package entry is a series of consecutive real versions created in the same workspace — that is, each change package entry records a particular patch to the element.

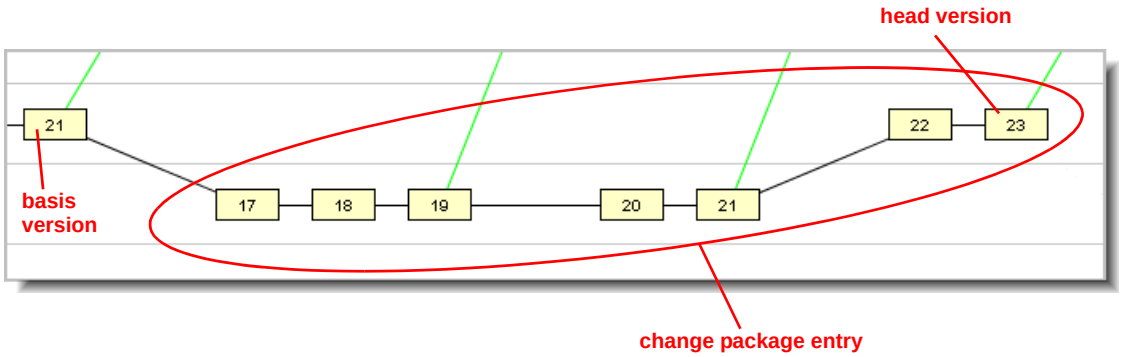
But the change package facility can also track *ongoing* changes to elements — changes made at different times, and in different workspaces. To support this capability, AccuRev defines a change package entry in a more general way than a patch:

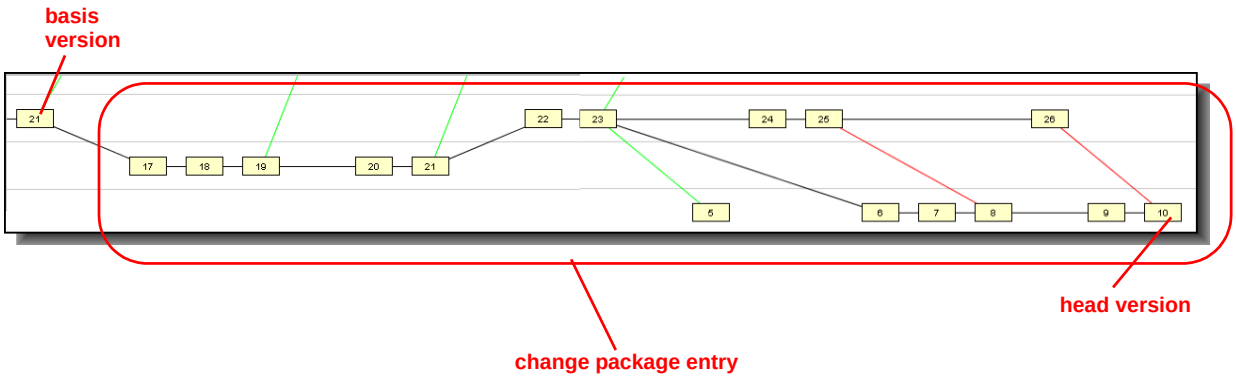
> A change package entry for an element consists of all the real versions in the element's version graph between a specified basis version and a specified head version. Between-ness is determined both by direct predecessor-successor connections (created, for example, by **Keep**)

and by merge connections (created by **Merge**). Patch connections are *not* considered in this determination; the basis version itself is not part of the change package entry.

The following Version Browser excerpts show the range of complexity that a change package entry can have. In fact, these excerpts show how the same change package entry can change over time, becoming more complex. See *Updating Change Package Entries* on page 48.





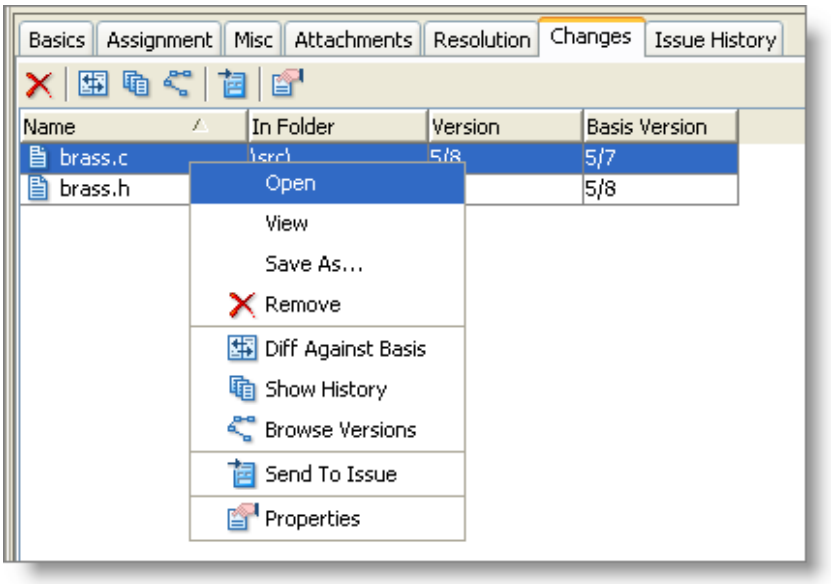


These illustrations suggest the following definition for a change package entry, which is equivalent to the definition above:

> A change package entry for an element consists of the element's entire version graph up to the specified head version, *minus* the entire version graph up to the specified basis version. For these purposes, the version graph includes direct predecessor-successor connections and merge connections, but not patch connections.

Operations on Change Package Entries

You can perform several operations on a change package entry, using its context menu. Most of these operations concentrate on the head version, which contains all the changes in the change package entry. (The head version appears in the “Version” column.)



Open

Run the appropriate command on the head version of the change package entry, according to its file type.

View

Open a text editor on a temporary copy of the head version (text files only).

Save As

Copy the head version to another filename.

Remove

Delete the selected entry(s) from the change package. This doesn’t affect an entry’s membership in other change packages, or the status of the element in the depot’s stream hierarchy.

Diff Against Basis

Use the graphical Diff tool to compare the head version with the basis version.

Show History

Open a History Browser on the selected element.

Browse Versions

Open a Version Browser on the selected element.

Send to Issue

Copy the selected entry(s) to another issue record, which you specify in a pop-up window. With this command, you can make the same entry(s) can appear in two or more change packages.

The pop-up window offers the set of issue records selected by the default query of the issues database, but you can enter the number of any issue record. If the other issue record already has an entry for the element, AccuRev attempts to combine them. See *Updating Change Package Entries* on page 48.

Properties

Displays information about the selected element: pathname, file type of current version, and element-ID.

The **Remove** and **Send To > Issue** commands enable you to maintain and fine-tune the contents of change packages. select the entry(s), invoke the **Send To > Issue** command, and specify another issue record. To remove one or more entries from a particular change package, just select them and invoke the **Remove** command.

Updating Change Package Entries

Change packages are designed to track ongoing changes to elements, not just a single set of changes. This means there will be times when you want to add a change package entry for a particular element, but an entry for that element already exists in the change package. In such situations, AccuRev attempts to combine the new entry with the existing one, producing an updated entry that includes all the changes. (Recall that there can be at most one entry for a given element in a given change package.)

A Little Bit of Notation

To help explain how AccuRev performs “change package arithmetic” to combine and update entries, we’ll use a simple notation. Suppose a change package entry contains the set of an element’s versions defined by these specifications:

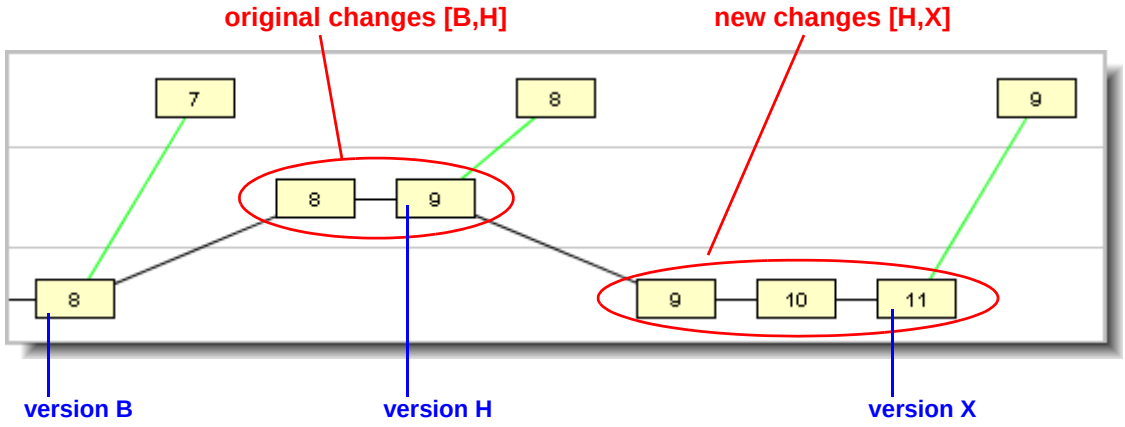
the head version is **H**
the basis version is **B**

We’ll use the ordered pair **[B,H]** to indicate this change package entry.

Combining Two Change Package Entries

Now, suppose a new change is to be combined with the existing change package entry **[B,H]**. There are several cases, each handled differently by AccuRev:

- **Case 1: [B,H] + [H,X]** — This simple case typically arises when you think you’re done with a task and record your work as change package entry **[B,H]** — but it turns out that more work on the same element is required. So you (or a colleague) start where you left off, with version **H**, and make changes up to version **X**. Then, you want to incorporate the new set of changes **[H,X]** into the same change package.

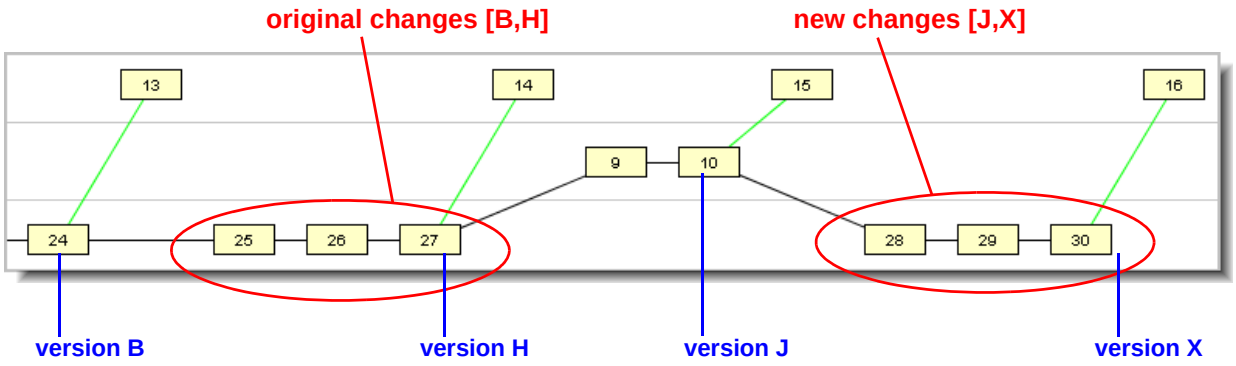


In this case, it's clear that the two series of changes can be viewed a single, uninterrupted series — starting at version B and ending with version X. That is:

[B,H] + [H,X] = [B,X]

Accordingly, AccuRev updates the change package entry automatically — keeping **B** as the "Basis Version" and changing the "Version" from **H** to **X**.

- **Case 2: [B,H] + [J,X]** (where H is an ancestor of J: "change package gap") — This case typically arises when you do work on a task at two different times, and someone else has worked on the same element in between.

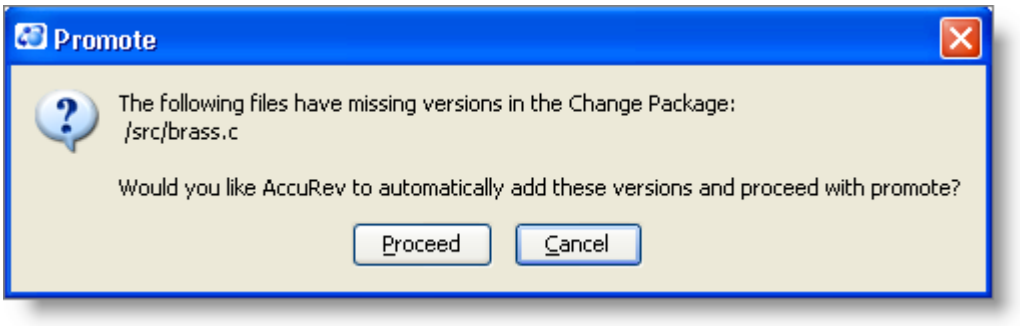


In this example, a colleague updated her workspace to bring in your original changes, created versions 9 and 10 in her workspace, and promoted her changes. You then updated your workspace to bring in her changes, and made a new set of changes.

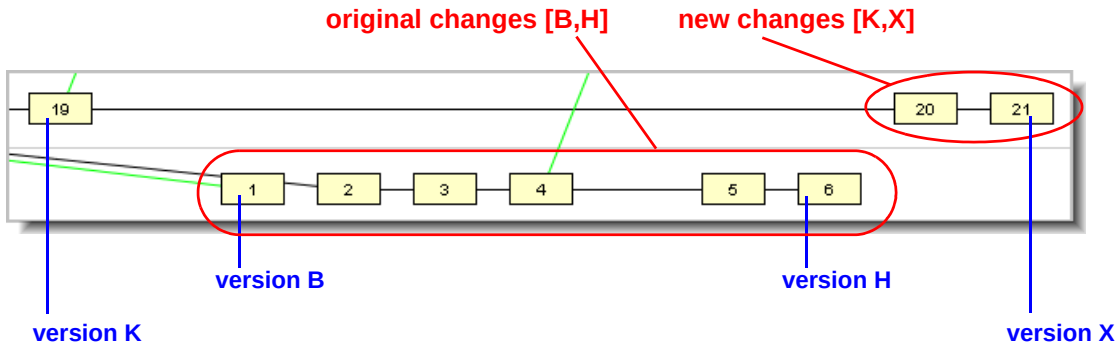
When AccuRev tries to combine the change **[B,H]** and the change [**J,X]** into a single change package entry, it detects that version **H** and version **J** are not the same, but that **H** is a direct ancestor of **J**. Thus, there is a simple "gap" in the potential combined change package entry (in this example, consisting of your colleague's versions 9 and 10).

Probably, your colleague was not working on the same task when she made her changes. (If she had been, she would have added her changes to the same change package, as in Case 1.) On the other hand, it's probably OK to include the entire, uninterrupted series of versions **[B,X]** in your change set — this includes both your original changes and your new changes (and, harmlessly, your colleague's changes, too).

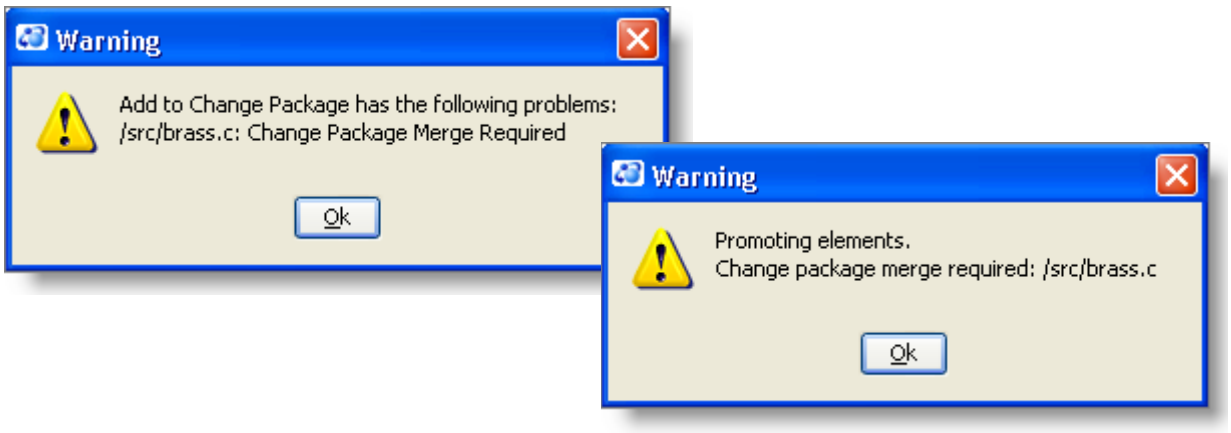
Accordingly, AccuRev prompts you to approve this “spanning the gap” between the two change set entries, in order to create a single, combined entry:



- **Case 3: [B,H] + [K,X]** (where H is *not* an ancestor of K: “change package merge required”) — This case typically arises when developers in workspaces that do not share the same backing stream try to use the same change package. There is no simple “gap” between the existing change package entry and the new one — which means there is no way to combine them into a single change package entry, according to definitions in *Complex Change Package Entries* on page 45.



AccuRev signals this situation with a “change package merge required” message, and cancels the current operation.



You can remedy this situation by performing a merge at the element level. (There is no merge operation defined at the change package level.) In the example above, merging version H and

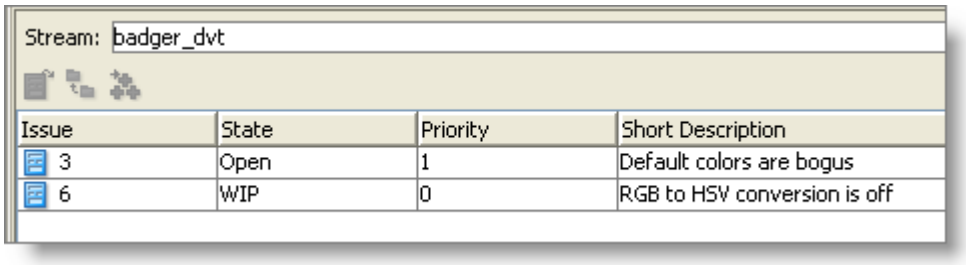
version X would create a new version; a change package entry with the new version as its head can be combined with the existing entry.

Viewing Stream Contents/Differences in Terms of Change Packages

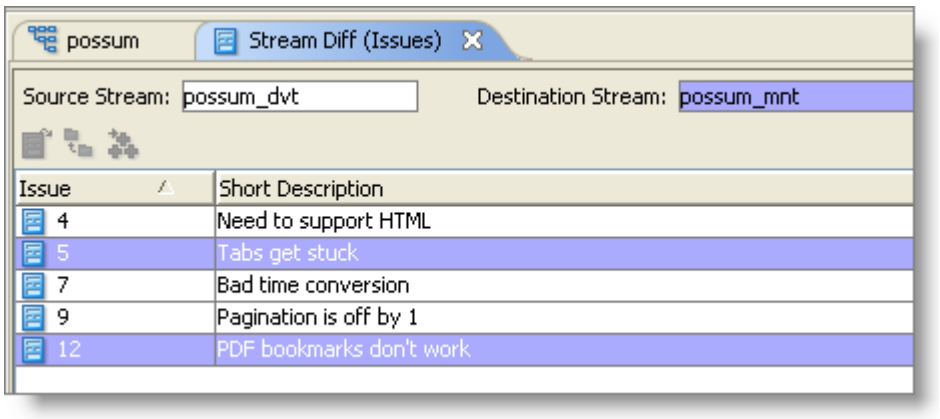
You have always been able to view the contents of a stream or workspace in terms of individual elements and versions (File Browser details pane). Likewise, you've been able to compare two workspaces/streams in terms of individual elements and versions (StreamBrowser **Diff** command).

With the advent of change packages, you can now view the contents of a single workspace/stream — or compare two workspaces/streams — in terms of change packages. The commands, available in the StreamBrowser, are **Show Issues** and **Show Difference > By Issues**. For both these commands, the result is a set of issue records. AccuRev displays the results similarly to the way it displays the results of a AccuWork query: a table in which each row is one issue record and each column is one field:

result of **Show Issues**

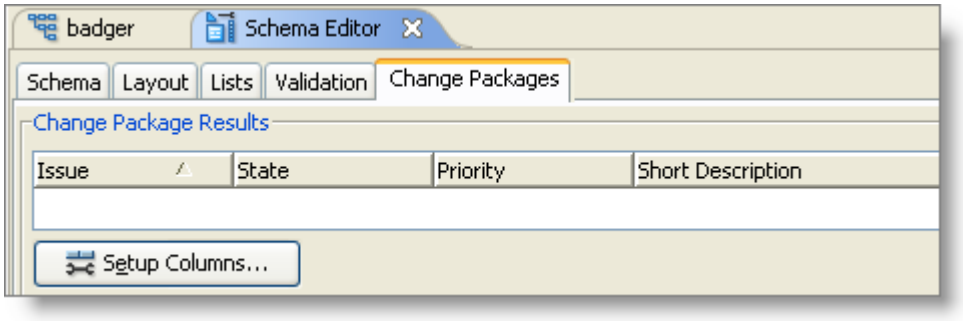


result of **Show Diff By Issues**



Formatting a Change Package Table

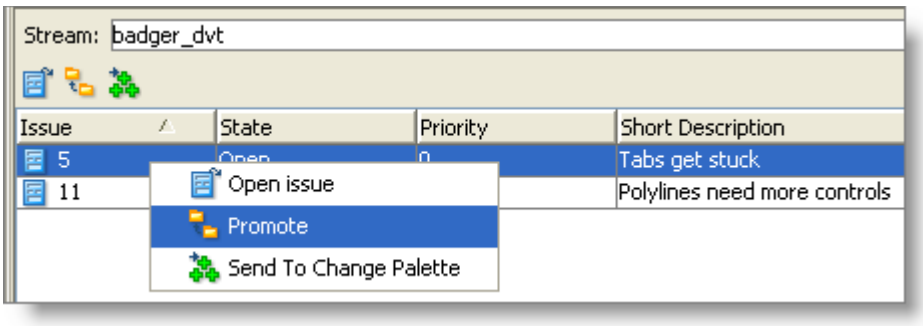
To specify the format of the tables produced by **Show Issues** and **Show Differences by Issues** commands, go to the Schema Editor's Change Packages subtab. In the top section, "Change Package Results", you can determine which fields appear as columns, the column widths, the order of the columns (fields), and the sort order of the rows (records).



For more on these commands, see the *AccuRev User's Guide (GUI)*: *Streams and Change Packages* on page 100, and *Comparing Streams Using Change Packages* on page 108.

Promote by Change Package

You can invoke the **Promote** command on one or more of the issue records in a **Show Issues** table. This promotes all the versions in the issue record's change package to the parent stream.

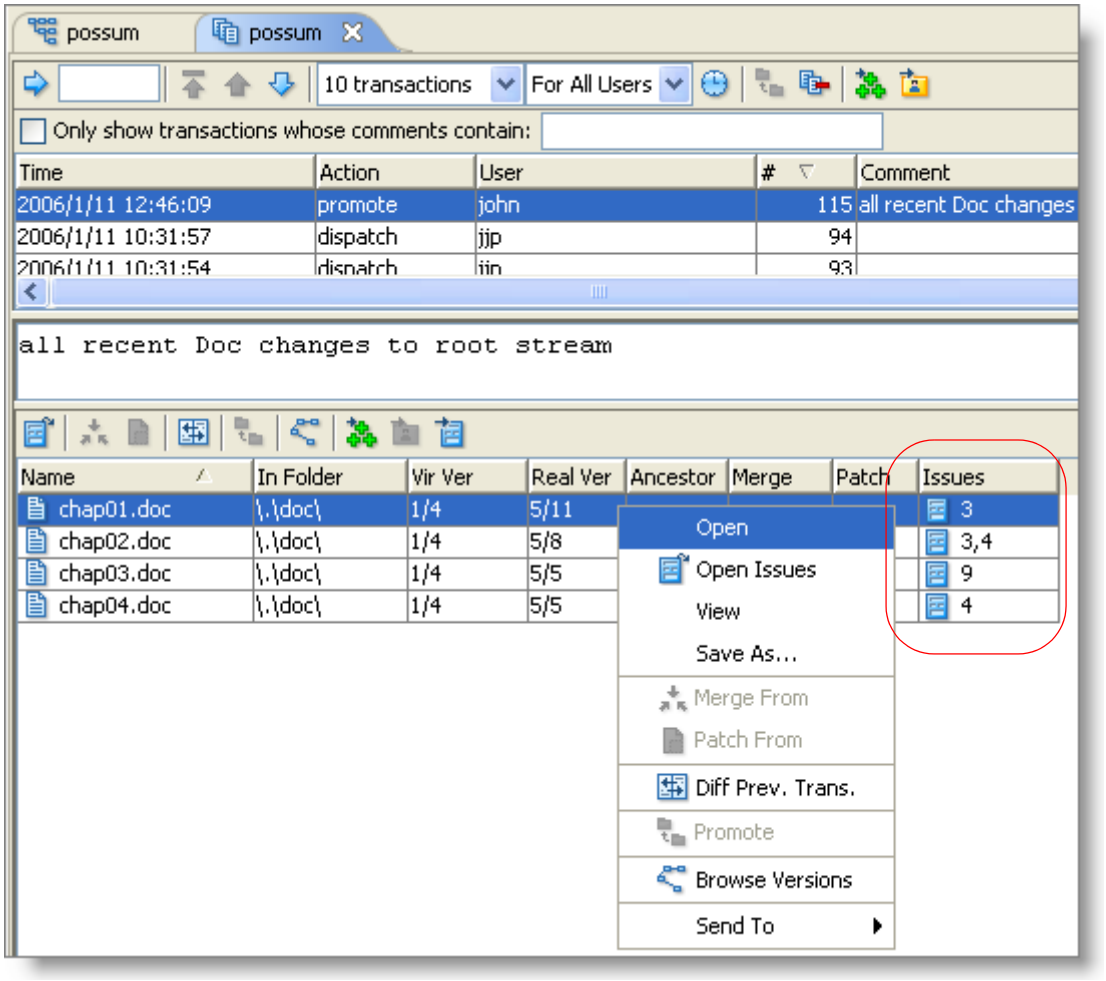


Promotion makes sense only for versions that are currently active in the stream. Accordingly, this command is restricted to issue records that are listed when **Show Active** is checked and **Show Incomplete** is not checked. A version in a change package won't be promoted if AccuRev determines that the changes in that version have already been propagated to the parent stream.

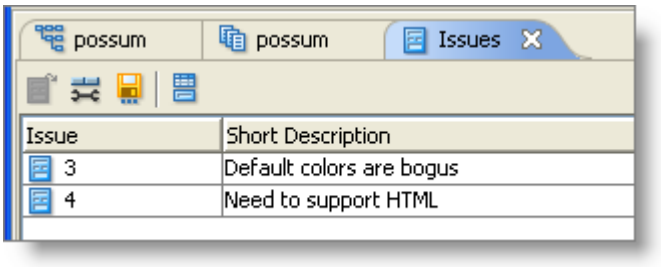
The **Promote** command is not enabled for issue records displayed by **Show Difference By Issues**.

Viewing a Transaction in Terms of Change Packages

The Versions pane of the History Browser shows information on all the versions involved in a selected transaction. This pane includes an Issues column, which indicates the change package(s) to which each version belongs.



The **Open Issues** command, on the context menu of a version, enables you to view the entire change package(s) — that is, the issue record(s). If you select a version that belongs to a single change package, an edit form opens on that one issue record. If you select a version that belongs to multiple change packages, an Issues tab opens, listing all the issue records.



“Locking” a Change Package

You can use AccuWork’s edit form validation facility to implement locking of change packages. This scheme relies of these facts:

- Each change package is the Changes tab of a particular issue record.
- Any tab of an issue record can be set to read-only status with the **setTabPermission** action in a validation.

For example, a validation could specify that the Changes tab's permission be set to **ro** when an issue record's Status changes to **Closed**. See *Defining Edit Form Validations* on page 32 and *Setting Permissions on All or Part of the Issue Record* on page 39.

Integrations Between Configuration Management and Issue Management

This section describes two similar, but separate facilities that integrate AccuRev's configuration management functionality with its issue management (AccuWork) functionality. One of them uses change packages as the point of integration. The other uses a particular issue-record field as the point of integration. Both integrations record information about the **Promote** transaction in a user-specified AccuWork issue record.

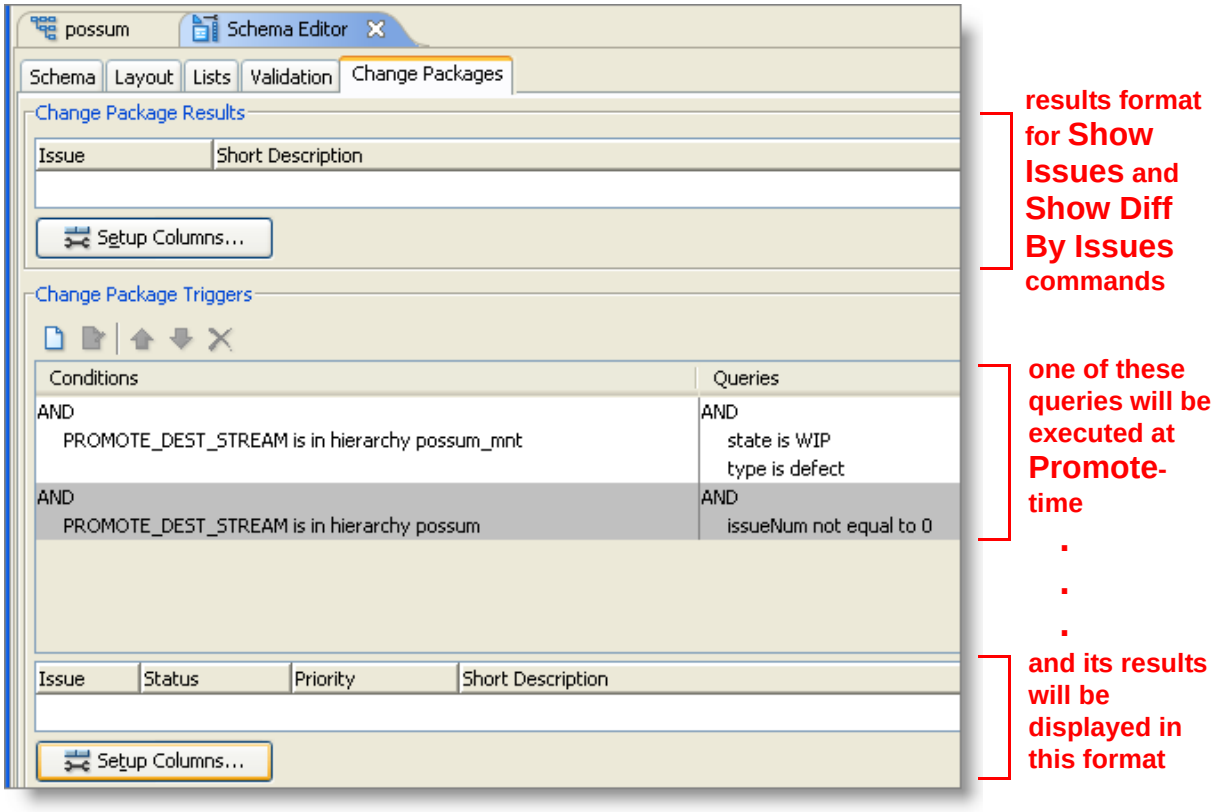
Change-Package-Level Integration

When a **promote** command is executed a user's workspace (but not in a higher-level dynamic stream), the change-package-level integration records all the promoted versions on the Changes subtab of a user-specified AccuWork issue record.

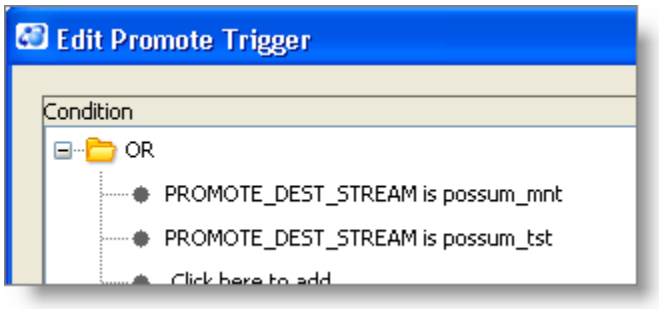
Enabling the Integration

The change-package-level integration is enabled on a depot-by-depot basis. Open the AccuWork Schema Editor for a particular depot's issue database, and go to the Change Packages subtab. Filling in the lower section, "Change Package Triggers", enables the integration for that particular depot.

The Change Package Triggers section is structured as a set of condition/query pairs. One of the queries will be selected for execution at **promote**-time. If the first condition is satisfied, the first query will be executed; otherwise the second condition will be evaluated, and if it's satisfied, the second query will be executed; and so on.



Each clause of a condition performs a test on the **Promote** destination stream. For example, this condition is satisfied if the user is promoting to either of the streams **possum_mnt** or **possum_tst**:



The query corresponding to each condition can be any AccuWork query, which selects a set of issue records. See *Using AccuWork Queries* on page 8.

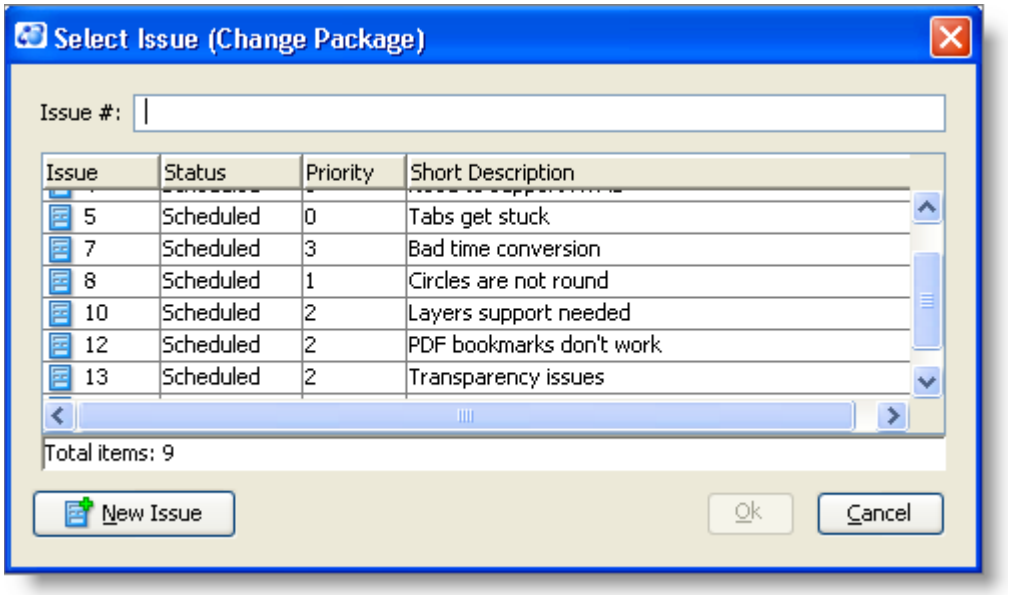
The Change Package Triggers section also specifies the format of each query's results table — the fields to appear as columns, the column widths, the order of the columns (fields), and the sort order of the rows (records).

Triggering the Integration

Once the integration is enabled for a depot, it is triggered whenever a user performs a **promote** command in a workspace associated with that depot.

If the **Promote** command is invoked through the AccuRev GUI:

1. One of the AccuWork queries specified in the Change Package Triggers section is executed, selecting a certain set of issue records.
2. Those records are displayed to the user in the results table format specified in the Change Package Triggers section.
3. The user selects one or more of the issue records. There is also a **New Issue** button, which enables the user to create a new issue record “on the fly”.



4. The command completes its work.
5. The versions involved in the command are recorded in the change package section (Changes page) of the selected issue record(s).

If the **promote** command is invoked through the AccuRev CLI:

1. Just as with the GUI, one of the AccuWork queries specified in the Change Package Triggers section is executed, selecting a certain set of issue records.
2. The user is prompted to specify an issue record:

   ```
   Please enter issue number ?
   ```

   Users can bypass this prompt by specifying an issue number with the **–I** option:

   ```
   > accurev promote -I 4761 chap01.doc
   Validating elements.
   Promoting elements.
   Promoted element \.\doc\chap01.doc
   ```

   Note: attempting to update an existing change package entry might cause a “change package gap” or “change package merge required” situation, either of which cancel the **promote** command. For example:

   ```
   Promoting elements.
   Change package gap: /doc/chap01.doc
   ```

You can handle a change package gap by adding the **–g** option to the **promote** command. This combines the new and existing change package entries by "spanning the gap":

```
> accurev promote -I 4761 -g chap01.doc
Validating elements.
Promoting elements.
Promoted element \.\doc\chap01.doc
```

There is no way to have the **promote** command automatically handle a "merge required" situation. You must either perform a merge on the element to be promoted, or remove the existing change package entry for that element.

For more on "change package gap" and "change package merge required" situations, see *Updating Change Package Entries* on page 48 above.

3. Assuming the user-specified issue record is one of those selected by the query, the command completes its work, and the promoted versions are recorded in the change package section of the selected issue record.

What happens if the user specifies no issue record or a non-existent issue record, such as 99999 or 0? In both the GUI and CLI cases:

- If the user enters "0" (or equivalently, makes a blank or non-numeric entry), AccuRev checks whether issue record #0 is among the issues selected by the query executed in Step 1.

  Note: the query *can* select issue record #0, even though it doesn't exist — for example with this clause:

  ```
  issueNum    equal to    0
  ```

  - If the query does select issue record #0, the user's command completes but no information is sent to the issues database. This provides a way for the user to bypass the integration when performing the **promote** command.
  - If the query does not select issue record #0, the user's command is cancelled, and no information is sent to the issues database.
- If the user specifies a non-existent issue record, such as "99999", the command is cancelled, and no information is sent to the issues database.

Transaction-Level Integration

The integration between configuration management and issue management at the transaction level records the number of each **promote** transaction in a particular field of a user-specified issue record.

Enabling the Integration

The transaction-level integration is enabled on a depot-by-depot basis, by setting the depot's **pre-promote-trig** trigger. For example:

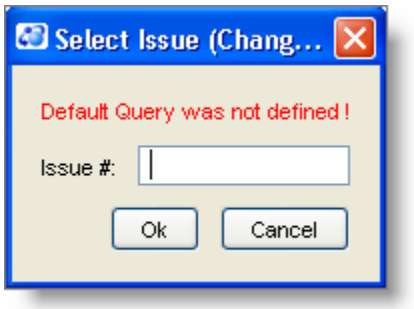
```
accurev mktrig -p kestrel pre-promote-trig client_dispatch_promote
```

Note that “client_dispatch_promote” is simply a keyword, not the name of a script file. The integration is implemented by two cooperating routines, one built into the AccuRev client software, one built into the AccuRev server software.

Triggering the Integration

Once the integration is enabled for a depot, it is activated whenever a user executes the **Promote** command in any workspace or dynamic stream.

1. The depot’s default query, as defined on the Queries tab (**Issues > Queries**), is executed and the results are displayed to the user.
2. The user selects one of the issue records. Note that if no default query is defined for the depot, the user is prompted to type an issue record number.
3. The **promote** command completes its work, propagating the versions to the backing stream.
4. The **promote** transaction number is recorded in the **affectedFiles** field of the selected issue record. (This change to the issue record is, itself, recorded as a transaction, of kind **dispatch**.)



If the user enters “0” or makes a blank entry, the **promote** command completes but no change is made to any issue record. This provides a way for the user to bypass the integration.

Over time, the **affectedFiles** field of a given issue record can accumulate a SPACE-separated list of **Promote** transaction numbers.

Implementation and Customization of the Transaction-Level Integration

When the transaction-level integration is activated, processing takes place on both the AccuRev client machine and the AccuRev server machine:

- The client-side processing — querying the AccuWork issues database and prompting the user to specify an issue record — is structured as a **pre-promote-trig** trigger routine built into the AccuRev client software.
- The server-side processing — updating of the AccuWork issue record — is structured as a **server-post-promote-trig** trigger routine built into the AccuRev server software.

You enable the integration by setting the **pre-promote-trig** trigger with the “client_dispatch_promote” keyword, as described above. You don’t need to explicitly set a **server-post-promote-trig** trigger script. If you do, the script runs instead of — not in addition to — the server-side built-in routine.

In most cases, you’ll want to avoid setting a **server-post-promote-trig** trigger script, just letting the built-in routines do their work. But suppose that after a **Promote**, you want the server machine to perform operations in addition to those defined in the transaction-level integration — for example, updating reference trees and sending email messages. In such cases:

1. Create a script that performs the server-side part of the transaction-level integration, along with the desired additional processing. Start with the sample script **server_dispatch_promote_custom.pl**, which is located in the **examples/dispatch** subdirectory of the AccuRev installation directory.
2. Place the script in the AccuRev **bin** directory.
3. Use a **mktrig** command to make the script the depot's **server-post-promote-trig** trigger script.

Further customizations of the transaction-level integration are possible. For example, you might want the user to be able to specify several issue records, not just one. Or you might want to link **promote** commands in one depot with the AccuWork issues database in another depot. Or you might want to update an issue record field other than **affectedFiles**. In such cases, you'll want to dispense with the built-in "client_dispatch_promote" routines altogether:

1. Start with the sample script **client_dispatch_promote_custom.pl** (in the **examples/dispatch** subdirectory), and create your own script for use as a **pre-promote-trig** script to execute on the client.
2. As described above, start with the sample script **server_dispatch_promote_custom.pl** (in the **examples/dispatch** subdirectory), and create your own script for use as a **server-post-promote-trig** script to execute on the server.

If Both Integrations are Enabled

Both the change-package-level and transaction-level integrations can be enabled for a given depot at the same time. In this case, a user performing a **Promote** command in a workspace is prompted to specify an issue record just once, not twice. The prompting for an issue record by the change-package-level integration takes place as usual. That issue record is then updated by both integrations.

Note that even if both integrations are enabled, a **Promote** command performed in a dynamic stream (not a workspace) activates just the transaction-level integration, not the change-package-level integration.

AccuWork Command-Line Interface

This chapter describes aspects of the command-line interface relevant to the AccuWork issue management system.

> Note: the information herein is accurate as of Version 4.0, but this interface might change or be discontinued in a future release.

Overview

The AccuWork CLI is implemented through a single command, **accurev xml**. The **xml** command is a non-interactive general-purpose command dispatcher; it reads a specified file to determine the AccuRev operation — in this case, an AccuWork command — to be invoked. For example:

```
accurev xml -l mycmd.xml        (“dash-ell” not “dash-one”)
```

Here, the **xml** command’s input comes from a file, **mycmd.xml**, which must contain an XML document. (The filename is irrelevant, and need not have a **.xml** suffix.) The XML document might contain this specification of a AccuWork query:

```
<queryIssue
        issueDB="UserReportedBugs"
        expandUsers="true">
21 == "rel2.0"
</queryIssue>
```

This example specifies the command, “find all issue records in depot **UserReportedBugs** whose value in field #21 (the **targetRelease** field) is the string **rel2.0**”. The results of an **xml** command are sent to standard output, also in the form of an XML document. For example, this query might retrieve two issue records, producing this output:

```
<issues>
  <issue>
    <issueNum
        fid="1">2</issueNum>
    <transNum
        fid="2">3</transNum>
    <targetRelease
        fid="21">rel2.0</targetRelease>
    <type
        fid="7">defect</type>

      ... additional fields ...

    <platform
        fid="12">All</platform>
  </issue>
  <issue>
```

```
      <issueNum
          fid="1">3</issueNum>
      <transNum
          fid="2">26</transNum>
      <targetRelease
          fid="21">rel2.0</targetRelease>
      <type
          fid="7">enhancement</type>

    ... additional fields ...

      <platform
          fid="12">Linux</platform>
    </issue>
  </issues>
```

This output provides the correspondence between field-ID numbers (e.g. `fid="21"`) and field-names (e.g. `targetRelease`). This correspondence is important, since you must specify a query using field-IDs, not field-names (e.g. `21 == "rel2.0"`, not `targetRelease == "rel2.0"`).

AccuWork CLI Operations

You can perform the following AccuWork operations through the command-line interface:

- Query an issues database (`<queryIssue>` document) — Retrieve the contents of all issue records in a particular depot (issues database) that match a specified query.
- Create a new issue record (`<newIssue>` document) — Enter a single new issue record in a particular depot.
- Modify an existing issue record (`<modifyIssue>` document) — Change the contents of a single issue record that already exists in a particular depot.

The sections below provide guidelines for performing each of these operations. But there's an important prerequisite step to perform first

Determining the Field-ID / Field-Name Correspondence

In the AccuWork CLI, you identify a field by its field-ID, not by its field-name. Thus, before doing any real AccuWork CLI work, you must determine the correspondence between field-IDs and field-names in your depot (issues database). This information is stored in the schema configuration file on the AccuRev Server host:

```
<AccuRev-inst-dir>/storage/depots/<depot-name>/dispatch/config/schema.xml
```

You can retrieve the contents of the **schema.xml** file from an AccuRev client machine with this command:

```
accurev getconfig -p <depot-name> -r schema.xml
```

Extract the **name=** and **fid=** text lines from this data, and store them for future reference. You'll need to refer to this information often as you work with the AccuWork CLI. Let's call this extracted data the field-ID definitions.

For example, the field-ID definitions for the default schema look like this:

```
name="issueNum"
fid="1">
name="transNum"
fid="2">
name="status"
fid="3">
name="shortDescription"
fid="4">
name="state"
fid="5">
name="productType"
fid="6">
name="type"
fid="7">
name="severity"
fid="8">
name="priority"
fid="9">
name="submittedBy"
fid="10">
name="dateSubmitted"
fid="11">
name="platform"
fid="12">
...
```

This data shows that field **submittedBy** has field-ID **10**, field **productType** has field-ID **6**, etc.

> Note: all examples in the remainder of this document will use the field-ID/field-name correspondence in the above example.

The contents of these XML elements are the field values for issue record #1. A couple of them, **submittedBy** and **dateSubmitted**, have values that might be a bit surprising — numbers instead of strings. We'll discuss these kinds of values in section *Selecting Issue Records with a Query* below.

Selecting Issue Records with a Query

The simplest kind of query selects one or more issue records by comparing the value of one field with a literal value (a constant) — for example, "is the value of the issueNum field equal to 415?" or "does the shortDescription field contain the string 'busted'?".

For such simple queries, you can adapt the one we used in section *Determining the Field-ID / Field-Name Correspondence* above. This query finds all issue records whose **productType** value is **Frammis**.

```
<queryIssue issueDB="XXXXX">
  6 == "Frammis"
</queryIssue>
```

The output of a query is an XML document whose top-level `<issues>` element contains zero or more `<issue>` sub-elements:

```
>>> accurev xml -l query-filename
<issues>
  <issue>
      ... individual field-name elements ...
  </issue>
  <issue>
      ... individual field-name elements ...
  </issue>

  ...

</issues>
```

More Complex Queries

You can compose and run arbitrarily complex queries. If the query goes just a bit beyond the simplest, you can probably compose it manually. For example, this query finds all issue records whose **productType** value is **Frammis** or **Widget**:

```
<queryIssue issueDB = "dpt38" useAltQuery = "false">
  <OR>
    <condition>6 == "Frammis"</condition>
    <condition>6 == "Widget"</condition>
  </OR>
</queryIssue>
```

With more complex queries, be sure to include the `useAltQuery = "false"` attribute in the `<queryIssue>` start-tag.

But to minimize the chance of getting lost in the syntax of more complex queries, we strongly recommend that you compose the complex query graphically, then "export" the query to a text file:

1. In the AccuRev GUI, enter the command **Issues > Queries** to enter the Query Editor.
2. Create a new query, give it a name, and specify the query logic.
3. Click the **Save All Queries** button.

4. Place an XML-format dump of *all* your queries in a text file:

   ```
   accurev getconfig -p depot-name -u user-name -r query.xml > myqueries.txt
   ```

   This **getconfig** command retrieves the contents of the configuration file that stores your queries:

   AccuRev-install-dir/storage/depots/*depot-name*/dispatch/config/user/*user-name*/query.xml

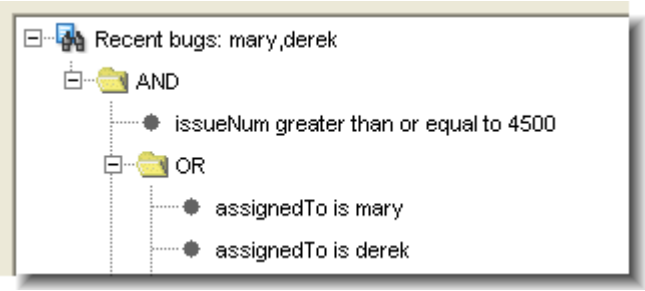
5. Use a text editor to extract the XML `<queryIssue>` element that defines the query. (Don't extract the entire `<query>` element, which includes the query's name and output field definitions.)

6. Store the `<queryIssue>` element code in a separate file, say **frammis_or_widget.xml**.

You can now invoke the query with the CLI:

```
accurev xml -l frammis_or_widget.xml
```

Example: This GUI-composed query ...



... is represented by this XML code:

```
<queryIssue issueDB = "dpt38" useAltQuery = "false">
<AND>

<condition>
1 >= &quot;4500&quot;

</condition>
<OR>

<condition>
14 == &quot;2&quot;

</condition>

<condition>
14 == &quot;24&quot;

</condition>
</OR>
```

```
</AND>
</queryIssue>
```

This code is perfectly good XML, even though it's not "pretty-printed". Also note the use of the XML entity reference **"** which is equivalent to a double-quote character.

Special Field Types

Each field in a AccuWork issues database has a field type: Text, Choose, List, etc. For a field of type User (such as the **submittedBy** field in the example in section *Determining the Field-ID / Field-Name Correspondence* above), a query defaults to reporting users by their integer user-IDs, rather than by their usernames (principal-names):

```
  ...
<submittedBy
    fid="10">40</submittedBy>
  ...
```

In this case, you could use the output of the **accurev show users** command to determine that user **jjp** has user-ID **40**. Alternatively, you can modify the query to set the attribute `expandUsers` in the `<queryIssue>` start-tag:

```
<queryIssue issueDB = "dpt38"
            useAltQuery = "false"
            expandUsers = "true">
             ...
```

Setting `expandUsers` causes the query to report the values of User fields with usernames instead of user-IDs:

```
  ...
<submittedBy
    fid="10">jjp</submittedBy>
    ...
```

For a field of type Timestamp (such as the **dateSubmitted** field in the example in section *Determining the Field-ID / Field-Name Correspondence* above), a query always reports the timestamp as a large integer, representing the number of seconds since Jan 1, 1970 UTC:

```
...
<dateSubmitted
    fid="11">1083606273</dateSubmitted>
...
```

(This is the standard Unix timestamp scheme.) You can use Perl to convert the integer into a human-readable string:

```
>>> perl -e "print scalar localtime(1083606273)"
Mon May  3 13:44:33 2004
```

Creating a New Issue Record

To create a new issue record in a particular issues database, execute the command

```
accurev xml -l my_datafile
```

... where **my_datafile** contains an XML document in this format:

```
<newIssue issueDB="XXXXX">
  <issue>
    ... individual field-value specifications ...
  </issue>
</newIssue>
```

As always, replace **XXXXX** with the name of the depot that stores the issues database. For the individual field-value specifications, adapt the output of a query that retrieves a single issue record. The complete contents of **my_datafile** might be:

```
<newIssue issueDB="UserReportedBugs">
  <issue>
    <type
        fid="7">defect</type>
    <submittedBy
        fid="10">5</submittedBy>
    <foundInRelease
        fid="20">rel2.0</foundInRelease>
    <productType
        fid="6">Widget</productType>
    <shortDescription
        fid="4">Names are sometimes trunca</shortDescription>
    <dateSubmitted
        fid="11">1083606275</dateSubmitted>  </issue>
</newIssue>
```

Some DOs and DON'Ts:

- You must specify the value of a User field with a user-ID (**5**), not a username (**derek**).
- You must specify the value of a Timestamp field with a number-of-seconds integer, not a string. You can use the **timelocal()** function in the Perl module **Time::Local** to generate these integers:

```
use Time::Local;
$sec = 35;   # range = 0 .. 59
$min = 22;   # range = 0 .. 59
$hr  = 14;   # range = 0 .. 23
$dte = 8;    # range = 1 .. 31
$mth = 5;    # range = 0 .. 11  (January is the zero'th month!)
$yr  = 2004; # play it safe: use a 4-digit number
$numseconds = timelocal($sec, $min, $hr, $dte, $mth, $yr);
print $numseconds, "\n";
```

- Don’t include specifications for the **issueNum** and **transNum** fields. AccuWork assigns these values automatically.
- The field initialization and validation code defined in the issues database schema will not be executed when the issue record is created. It’s up to you to specify the appropriate values for the appropriate fields.

  The validations *will* be invoked when the issue record is subsequently opened in the AccuWork GUI. In particular, you can create an issue record with a List field whose value is not in the field’s list of possible values. But when the AccuWork GUI opens the issue record, it replaces the bogus value with `<none selected>`.
- Be sure that the top-level and second-level XML elements are named `<newIssue>` and `<issue>`. The names for the individual-field elements are irrelevant — only the **fid** attributes count. The following specifications are equivalent:

```
<status fid="20">New</status>
<myfield fid="20">rel2.0</myfield>
```

When you submit the **<newIssue>** data structure to the AccuWork CLI, it creates the record and reports the new record’s contents. This report includes the automatically assigned **issueNum** and **transNum** values:

```
>>> accurev xml -l my_datafile
<issue>
  <issueNum
      fid="1">11</issueNum>
  <transNum
      fid="2">154</transNum>
  <type
      fid="7">defect</type>
  <submittedBy
      fid="10">24</submittedBy>
  <foundInRelease
      fid="20">rel2.0</foundInRelease>
  <productType
      fid="6">Frammis</productType>
  <shortDescription
      fid="4">Refuses to fram</shortDescription>
  <dateSubmitted
      fid="11">1062787292</dateSubmitted>
</issue>
```

See also *Using ‘modifyIssue’ to Create New Issue Records* on page 69 below.

Modifying an Existing Issue Record

Use the following procedure to modify an existing issue record:

1. Create a query to select the desired issue record, as described in *Selecting Issue Records with a Query* on page 63. For example, to select issue record #472 from the **Problems** issues database, create this XML document:

```
<queryIssue issueDB="Problems">
  1 == "472"
</queryIssue>
```

2. Run the query, storing the results in a text file:

```
accurev xml -l myquery.xml > issue472.xml
```

3. Edit the text file, making these changes:
   - Change the top-level XML **<issues>** start-tag to **<modifyIssue>**. Include the **issueDB** attribute in the start-tag:

     ```
     <modifyIssue issueDB = "Problems">
     ```

   - At the end of the file, change the **</issues>** end-tag to **</modifyIssue>**.
   - Remove the entire **<transNum>** element.
   - Change the values of one or more existing fields.
   - If you wish, add new field values, using the discussion in *Creating a New Issue Record* on page 67 as a guide.
4. Submit the edited text file:

```
accurev xml -l issue472.xml
```

When you submit the **<modifyIssue>** data structure to the AccuWork CLI, it modifies the specified issue record. As when you create a new issue record with the AccuWork CLI, field validations are not applied.

To verify the new record's contents, submit the original query again:

```
accurev xml -l myquery.xml
```

Using 'modifyIssue' to Create New Issue Records

Instead of using a **<modifyIssue>** document to change an existing issue record, you can use it to create a new issue record. This is useful for copying issue records from one issues database to another. For example, you might use a **<queryIssue>** document, as described earlier, to retrieve the contents of an issue record from database **Problems**, in the form of an **<issues>** document. As described in Step 3 above, when you change the **<issues>** tag to a **<modifyIssue>** tag, specify a different database:

```
<modifyIssue issueDB = "Problems_Public">
```

In this example, the issue record will effectively be copied from the **Problems** issues database to the **Problems_Public** issues database. Make sure the target database exist and has the same schema as the source database.

The **<modifyIssue>** technique is also useful for making copies of the issue records that act as change packages, and for replicating issue records between different AccuRev sites.

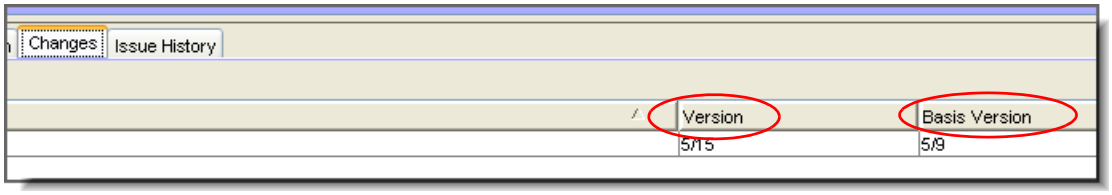
Note that a new issue record created with **<newIssue>** always gets assigned the next available issue number in the database; by contrast, a new issue record created with **<modifyIssue>** gets the issue number specified by the **<issueNum fid="1">** subelement:

```
<issueNum
      fid="1">4197</issueNum>
```

An issue record with this number must not already exist in the target database, but there is no other restriction. It's perfectly OK to have a "sparse" issues database, in which most issue numbers are unallocated.

Interface to the Change Package Facility

AccuWork issue records are used to implement the change package facility. The set of changes in a change package is indicated by a set of "Versions", listed on the Changes subtab of an issue record, each with a corresponding "Basis Version":



Each Version / Basis Version pair defines a set of changes to the element: the changes made since the Basis Version was created, up to and including the Version. The change package consists of such Version / Basis Version "change intervals" for any number of elements.

Various user commands and triggers maintain the contents of a change package: adding new versions and removing existing versions. There are also commands for comparing the contents of a change package to the contents of a stream, enabling you to easily answer the question, "Have all the changes made for Task A been propagated to Stream B?"

The following sections describe the CLI to the change package facility.

Adding Entries to a Change Package

The following XML document requests the adding of two entries to the change package of issue record #433: for the element with element-ID 3, record the series of versions between Basis Version 4/3 and Version 4/6; for the element with element-ID 7, record the series of versions between Basis Version 4/2 and Version 4/9.

```
<acRequest>
  <cpkadd>
    <user>jjp</user>
    <depot>etna</depot>
    <stream1>etna_dvt</stream1>
    <issues>
```

```
        <issue>
          <issueNum>433</issueNum>
          <elements>
            <element
               id="3"
               real_version="4/6" basis_version="4/3">
            <element
               id="7"
               real_version="4/9" basis_version="4/2">
            </element>
          </elements>
        </issue>
      </issues>
    </cpkadd>
  </acRequest>
```

Removing Entries from a Change Package

The following XML document requests the removal of the change-package entry for the element with element-ID 3 from issue record #433.

```
  <acRequest>
    <cpkremove>
      <user>jjp</user>
      <depot>etna</depot>
      <stream1>etna_dvt</stream1>
      <issues>
        <issue>
          <issueNum>433</issueNum>
          <elements>
            <element
               id="3"
               real_version="4/6">
            </element>
          </elements>
        </issue>
      </issues>
    </cpkremove>
  </acRequest>
```

Listing the Contents of a Change Package

The following XML document requests the listing of the change packages of issue records #433 and #512.

```
  <acRequest>
    <cpkdescribe>
      <user>jjp</user>
```

```
    <depot>etna</depot>
    <stream1>etna_dvt</stream1>
    <issues>
      <issueNum>433</issueNum>
      <issueNum>512</issueNum>
    </issues>
  </cpkdescribe>
</acRequest>
```

Listing Transactions that Affected Change Packages

The following XML document requests the listing of transactions in the range 488–569, annotated with the numbers of the change packages (issue records) modified by those transactions.

```
<acRequest>
  <cpkhist>
    <user>jjp</user>
    <depot>etna</depot>
    <stream1>etna_dvt</stream1>
    <transaction1>569</transaction1>
    <transaction2>488</transaction2>
  </cpkhist>
</acRequest>
```