CA DataMinder

iConsole Search Definition Reference Guide

Release 14.6

CA Technologies Product References

This document references the following CA Technologies products:

- CA DataMinder™

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: About iConsole Search Definitions

The CA DataMinder iConsole has been designed to support flexible interrogation of the CA DataMinder database. Predefined searches are provided, but you can modify, supplement or replace these to meet your organization's particular search requirements.

This document provides guidelines for creating and modifying iConsole searches, and includes reference information for all supported facilities.

Chapter 2: Search Architecture

iConsole searches and reports comprise a number of components:

Database Stored Procedures (SPs)

Define the query logic that is executed and the results that are returned.

Search Definitions

XML documents that define the properties of the search that form the inputs that the SP requires and the layout of the results from the SP. The Search Definition can also include scripting to define behavior for the properties or results displays.

Script Files

In addition to inline script in the XML, you can load script from separate script files. This can be useful when the script is used by several searches or reports, reducing duplication.

Help Files

You can provide help files for both the properties and results pages to guide the user on how to provide the required inputs, and how to interpret the results.

Report Format Files

You can produce custom reports by specifying an XSL file to transform the raw results into the required report format. This can be further processed into a document format (such as PDF) if required.

This section contains the following topics:

Architecture

The iConsole is a Web Application that you access using a web browser (such as Internet Explorer). The program logic is distributed across four logical systems:

- The **Browser** provides the presentation layer of the application. The iConsole display is primarily generated using javascript on the user's workstation. Requests are made to the Web Server using AJAX.
- The **Web Server** manages the user's session, delivering resources such as web pages, script files and images, and manages requests for CA DataMinder information through a SOAP interface.
- The **Application Server** provides Web Services that encapsulate the business logic of the iConsole. The Web Services are called upon by the Web Server to deliver information from the CA DataMinder database via the CA DataMinder Infrastructure.
- The **CMS** manages the CA DataMinder database and responds to the requests from the Application Server.

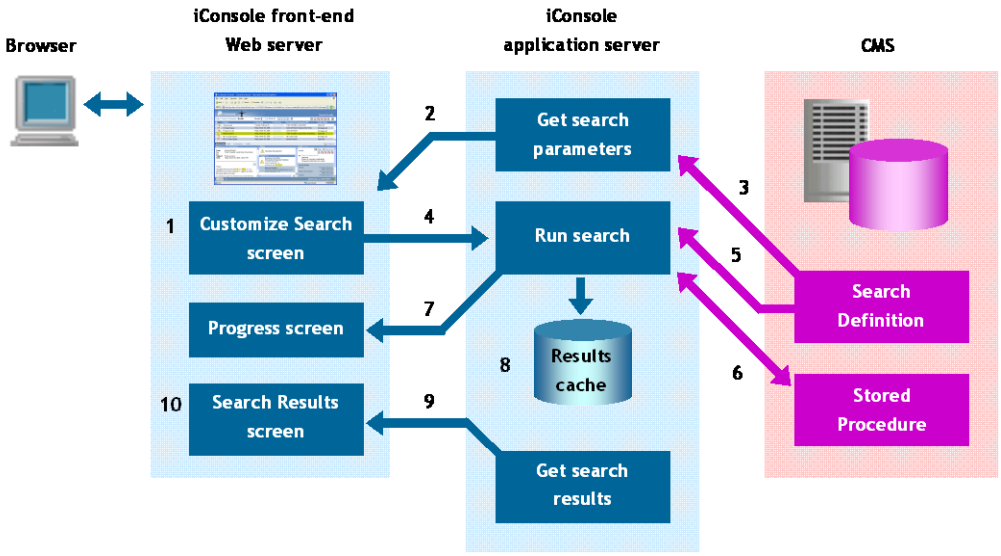
Each of these systems can be hosted on separate computers, all installed on the same computer or any other configuration to meet the performance needs of the business.

As far as iConsole search components are concerned:

- The Database Stored Procedures and Search Definitions are installed on the CMS.
- Script Files and Help Files are installed on the Web Server
- Report Format Files are installed on the Application Server.

Search Execution

The following diagram illustrates the typical search execution:



1. The user selects the search to be run. An AJAX request is made to the Web Server for the search details. The Web Server sends a SOAP request to the Application Server for the Search Definition. The Search Definition is retrieved from the CMS and delivered to the Web Browser. The Browser renders the parameters defined in the Search Definition for the user to edit.
2. The user enters values for the parameters and clicks Run. The browser sends the parameters to the Web Server which makes a request to run the search to the Application Server. The Application Server starts a thread to run the search and returns immediately. The browser displays a progress dialog, periodically polling to see if the search is complete and updating the progress.
3. The Application Server thread merges the supplied parameter values with the Search Definition and runs its stored procedure by making a request to the CMS. When the search is complete, the results are retrieved and packaged for return to the browser, along with results display details from the Search Definition. The results are temporarily stored in the Application Server cache.
4. When the progress indicates that the results are ready, the Web Browser requests the results from the Web Server and it makes a request to get the results from the Application Server cache. The results are returned to the Browser which renders them, using the display details, for the user to review. Paging, filtering and sorting of results is all performed in the browser (with the exception of unlimited searches).

More information:

Supporting Searches of Unlimited Size (see page 94)

Chapter 3: Versioning Scheme

Searches installed in the iConsole support a versioning scheme to facilitate upwards compatibility of saved searches wherever possible. This section describes how this versioning scheme operates.

As an illustration of the problem, suppose we have installed and published a search definition in the iConsole, and installed the corresponding stored procedure on the CMS. This search is being used every day and users have saved searches based upon it. We now want to add an extra feature to the search. Without versioning we have two options:

- **Update the existing search, keeping the name of the stored procedure the same.**

  The problem with this option is that we must test the updated (unpublished) search and fix any problems. While we are doing this, the published search, and any user searches based on it may fail or give incorrect results because they are accessing the modified stored procedure, which may have errors or have incompatible parameters.

- **Create a new search with a new stored procedure name**

  The problem with this option is that saved searches of users do not benefit from improvements in the search when we publish it. You will have to create a completely new set of saved searches based on the new search.

Search versioning is designed to overcome both of these problems.

When a search definition is created, major and minor version numbers (generally 1 and 0) are defined. These attributes, along with stored procedure name, are used to identify the stored procedure that the search definition will access. When a search needs to be modified, minor and/or major version should be updated based on the impact of the change.

This section contains the following topics:

Minor Updates

A minor update is one that does not affect the upwards compatibility of saved searches. In general this means that:

- The parameter names have not been changed – new names may not cause the search to fail, but parameters from the saved search using the old names are discarded, and incorrect results are obtained.
- The parameter types have not been changed.

However, the new version of the search can:

- Define additional parameters – default values are used for these parameters when an old saved search is merged with the new search definition.
- Change the parameter order – the new search definition arranges the saved parameters in the correct order for the new stored procedure.
- Change the content or layout of the results.

Incrementing the minor version is usually appropriate for most changes involving bug fixing, performance tuning and adding extra capabilities.

Minor updates are automatically picked up by saved searches when the new version is published.

Major Updates

An increment of the major version is only required when the changes to the search are such that saved searches can no longer be expected to function correctly. For example this occurs if the search was completely revised, using a completely different set of input parameters.

Major updates are not picked up by saved searches based on previous major versions.

Saved Searched Update Automatically

When a user saves a search, only the parameter values for the search are saved along with a reference to the search it was saved from.

When the saved search is run, the iConsole retrieves the search definition with the most recent minor version for the same major version. This means that if a custom search was saved against Standard_Search 1.0 and the current, published version of the standard search is now 1.5, the saved search uses version 1.5 of the search. However, if version 2.0 of the search is produced, the saved search does not use this as it is considered to be incompatible, and continues to use 1.5.

The iConsole merges the saved parameters with the latest search definition, using defaults for any new parameters that have been added since the search was saved. In this way, saved searches continue to function, but benefit from updates to improve performance or correct errors and also new features using the default parameter values.

Search Visibility

Only the most recent version of a search with a given major version is displayed on the Review page as users only need to see the latest one. Several minor versions of a search may still be present, however and may be seen in the Administer Searches | Searches page allowing the administrator to remove old versions as and when required.

Example: If versions 1.0, 1.1, 1.2, 1.3, 2.0 and 2.1 of a particular search have been installed, only versions 1.3 and 2.1 would be visible in the Review page Searches folder.

Note: The reason that old versions of searches are not automatically removed when the new version is published is that large installations may have multiple iConsole server installations which may be at different iConsole version levels (for example, part way through an upgrade project). If new version of a search requires a particular version of the iConsole (both web and application servers) to function (see the 'requires' attribute in Search Definition Syntax), it is only displayed on the upgraded iConsole servers; older installations continue to show an earlier version of the search.

Search Definitions

The search definition XML defines the search to the iConsole and includes the name of the stored procedure (spname) and the major and minor version numbers. The search definition defines the interface to the stored procedure so they must be kept in step.

The iConsole has a publishing process allowing new searches to be tested before they are made available to users. After a new version of a search definition is published, the new version may be used automatically by any saved searches based on the original search.

Note: The publication process in the iConsole only affects the search definition; the stored procedure is not moved or renamed. Access to a different version of a stored procedure is controlled through publication of its search definition.

Stored Procedures

The name of the stored procedure (SP) is constructed like this:

```
<spname>_V<major>_<minor>
```

For example, with an spname of Standard_Search, major version 1 and minor version 0 the stored procedure name would be "Standard_Search_V1_0".

When the search is modified, the major or minor version should be incremented so that a completely new stored procedure is referenced. This allows the new version of the search to be tested from the unpublished folder of the iConsole, accessing the new SP, while the original, published search and any saved searches can continue to use the original SP, unaffected by the changes.

Note: On a SQL Server database, the stored procedures are all independent and so the new version can be installed independently. However, on Oracle the stored procedures are defined within a package. In this case all versions of the stored procedure must be retained in the package as the complete package is replaced during SP installation.

Chapter 4: Stored Procedures (SPs)

This chapter includes SP implementation and naming details.

This section contains the following topics:

SP Implementation

Although the detailed implementation of the SPs differs depending on the database engine, the SPs must always conform to a standard interface convention to be compatible with the iConsole.

Oracle

Searches defined for Oracle databases must be implemented as **functions**, taking positional parameters and returning a **refcursor**. The columns returned using the refcursor must match the search definition in terms of number, name and type.

SQL server

Searches defined for SQL Server databases must be implemented as **procedures** taking positional parameters. In addition:

- The final SELECT statement in the procedure must yield columns that match the search definition in terms of number, name and type.
- If a procedure contains multiple SELECT statements the directive NO COUNT ON must be included in the procedure.

Note: You can specify a search with no results column definitions. In this case the results are displayed based using the name and type of the returned column. The restriction that columns must match the search definition in terms of number, name and type does not apply in this case.

SP Naming Conventions

The SP name must include a suffix defining the SP version. This must be in the format _Vx_y, for example MyEmailSearch_V2_1.

where:

x

Is the major version number. The major version is updated when a significant change is made to the function or procedure that requires an updated search definition. Existing customized searches will not use this new version.

y

Is the minor version. The minor version is updated when the procedure or functions is changed but the interface with the search definition stays the same. That is, the parameter and results definitions remain the same. This ensures that there is change control, but users' customized searches continue to function.

Database Views

CA DataMinder databases include various database views that you can use when writing database queries. These views provide access to underlying database tables and ensure that Row level security (RLS) is not bypassed.

RLS ensures that reviewers cannot see events associated with users outside of their management groups or the policies that they manage (depending on configuration) when searching the CMS database for events. RLS is primarily applied to events, users, groups and triggers, with the underlying tables (Wgn3Event, Wgn3User, WgnGroup and Wgn3Trigger) being inaccessible directly to search users.

For more information on the supported views, see the *Database Views Guide*.

To maintain database integrity and insulate the SPs from database schema updates, the data in the database must be accessed using the documented views, where available, rather than accessing tables directly.

Example SPs

These examples define a simple search for e-mail events whose title contains a specific word or phrase. The SP takes one parameter—the string that you want to match—and returns the event ID and title.

Example: Stored Function for Oracle

```
CREATE OR REPLACE FUNCTION MyEmailSearch_V2_1(MyMatch VARCHAR2)
RETURN SYS_REFCURSOR IS
     MyCursor SYS_REFCURSOR;
BEGIN
     OPEN MyCursor FOR
     SELECT event.eventUID AS "event_uid", event.EventText2 AS "title"
          FROM Wgn_V_Event_2 event
     WHERE event.EventText2 LIKE '%' || MyMatch || '%'
          AND (event.EventMajorType=2 AND event.EventMinorType<>18);
     RETURN MyCursor;
END;
```

Note the following:

- The MyMatch parameter passes the required phrase to the stored function.
- The WHERE clause also includes the constraint on event major and minor types to ensure the search only yields e-mail events.
- SYS_REFCURSOR is used to return the result.

Example: Stored Procedure for SQL Server

```
CREATE PROCEDURE MyEmailSearch_V2_1(@MyMatch NVARCHAR(255))
AS
BEGIN
     SET NOCOUNT ON
     SELECT event.eventUID AS "event_uid", event.EventText2 AS "title"
          FROM Wgn_V_Event_2 event
     WHERE event.EventText2 LIKE '%' + @MyMatch + '%'
          AND (event.EventMajorType=2 AND event.EventMinorType<>18)
END
```

Note the following:

- The @MyMatch parameter passes the required phrase to the SP.
- SPs for SQL Server **must** include the SET NOCOUNT ON directive.

Chapter 5: Standard SQL Helper Functions

To assist with the development of iConsole searches a library of helper functions is provided. These functions are grouped into collections of related functions with names beginning with WgnWC_.

On Oracle systems these are implemented as Oracle packages so a function call has the form WgnWC_Group.FunctionName

On SQL Server systems there is no equivalent of the Oracle package, so the package name is added as a prefix to the function name along with an underscore. A function call has the form WgnWC_Group_FunctionName.

This section contains the following topics:

Utilities

Typical functions in this collection include functions to do the following:

- Return an icon definition for the specified event
- Return a string for a LIKE clause with special characters substituted
- Return the audit status of an event

Logging

The logging functions are used for debugging and problem solving. Because logging requires permission to write to the file system on the database server, additional configuration is required as follows:

Oracle

Logging uses DIRECTORY objects to define the location of the log files. These objects must pre-exist, or the privilege CREATE ANY DIRECTORY must be granted to the CA DataMinder user accounts.

SQL Server

OLE Automation must be enabled. This requires two steps:

a. Enable OLE Automation (as sa):

```
exec sp_configure 'show advanced options', 1
reconfigure exec sp_configure 'OLE Automation Procedures', 1
reconfigure
```

b. Grant access to OA methods (as sa):

```
use master
grant execute on [sys].[sp_OACreate] to [public]
grant execute on [sys].[sp_OAMethod] to [public]
grant execute on [sys].[sp_OADestroy] to [public]
grant execute on [sys].[sp_OAGetErrorInfo] to [public]
```

Chapter 6: Constraints on Custom Searches

The following constraints apply when defining a custom search:

- Queries are normally executed by a special database user WgnQueryUser. For the query to run successfully, you must grant this database user access to all customized procedures or functions. To do this, run:

  ```
  GRANT EXECUTE ON MyEmailSearch_V2_1 to "WGNQueryUser"
  ```

- **Oracle**
  - When you insert data into global temporary tables, these must be committed to the database in the query to release UNDO space.
  - Use a package to simplify the granting of EXECUTE access to a collection of procedures or functions. The standard queries delivered with the iConsole are contained in the Wgn_iConsole package.
  - If you do use an Oracle package, prefix all referenced custom functions with the package name, for example, myPackage.myFunction.
  - All references to secure views (see the list below) must be prefixed with the current user name. This means you will need to use Native Dynamic SQL to construct the query, for example:

    ```
    OPEN mycursor FOR 'SELECT * FROM ' || myUser || '.Wgn_V_Event_1';
    ```

    You can retrieve the current user name using this command:

    ```
    SELECT sys_context('userenv','current_user') INTO myUser FROM dual
    ```

    Secure views at the time of writing:

    ```
    Wgn_V_Event_1
    Wgn_V_EventPartpntUser_1
    Wgn_V_User_1
    Wgn_V_Group_1
    Wgn_V_Trigger_1
    ```

- **SQL Server**
  - If temporary tables are needed, use local temporary tables (prefixed with #). Any constraints on the table (that is, the primary key) cannot be named as these names are global.
  - SQL Server stored procedures must include the SET NOCOUNT ON directive to ensure correct operation, particularly where the procedure includes more than one SELECT.

Chapter 7: Search Definition

The search definition is described in an XML document that must conform to the supplied XML schema. It defines the attributes of the search and has two major sections defining the search parameters and the search results. The search definition schema supports multiple search definitions in one file.

This section contains the following topics:

Example Search Definition

A search definition to support the MyEmailSearch SPs in the Example SPs section looks as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<searches>
     <search
          spname="MyEmailSearch"   (1)
          major="2"
          minor="1"
          label="My First Search"
          description="Finds e-mails with specific text in the title."   (2)
          help="help/htm/MyEmailSearchHelp.htm"> (3)
         <parameters>   (4)
               <parameter
                    name="txtMatch"
                    type="text"
                    label="Match this text in e-mail title"
                    argpos="1"
               />
         </parameters>
         <results>
               <column   (5)
                    name="event_uid"
                    type="id"
                    label="Event ID"
               />
               <column   (5)
                    name="title"
                    type="text"
                    label="E-Mail Title"
               />
         </results>
     </search>
</searches>
```

Notes

1 SP name concatenated from these search attributes.

2 Search name and description.

3 HTML help page for the current search.

4 Search parameter: input box for search text.

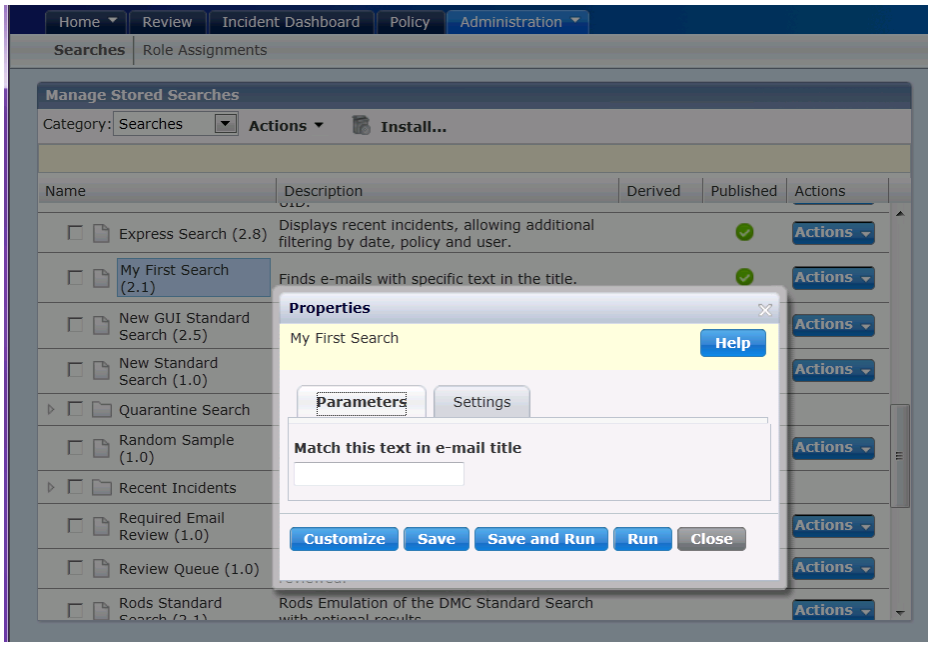
5 Event ID and E-Mail Title columns in search results screen. The name attribute is referenced in the SELECT statement in the MyEmailSearch SPs in Example SPs.

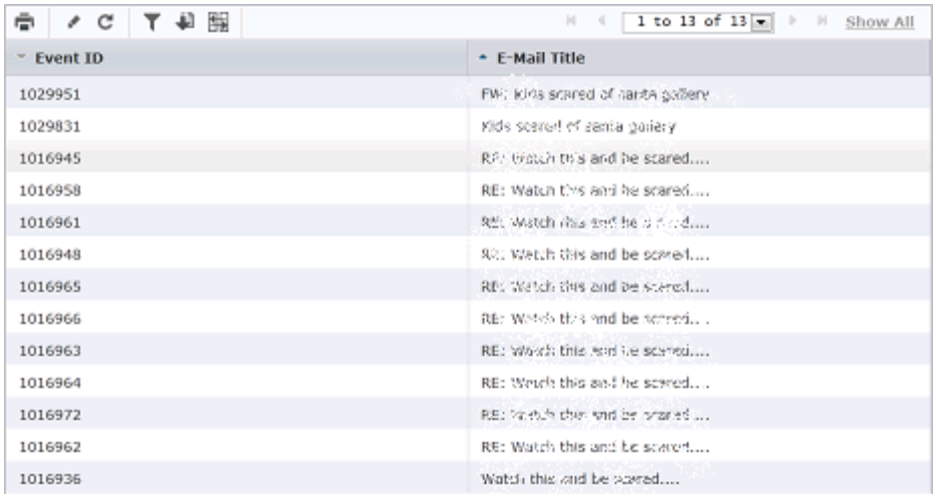
More information:

XML Elements (see page 33)
Reference Information (see page 127)
Example SPs (see page 23)

Example iConsole Screens

Based on the search definition file in Example Search Definition, the properties and search results pages for the MyEMailSearch SP and search definition are as follows:





Notes

Be aware of the following:

- You must save the search definition with UTF-8 encoding.
- If your search definition needs to include special characters such as &, <, >, " or ' these must use the XML escaped forms &, <, >, " or ' respectively. Other special characters such as © may be included using numeric character references like © or © (hex) where the number indicates the character code.

More information:

Default Sort and Column Selection Options (see page 33)

Default Sort and Column Selection Options

By default, all iConsole search screens include a standard set of settings such as Display Sort Order (unless the search results are defined with sortable="false") and selection of optional columns (if some are defined in the XML). These options are added automatically to the Settings tab of the search Properties page.

XML Elements

The main elements and attributes are summarized below.

<searches> root element

The root element must be <searches> and it must declare the XML Schema namespace and the location of the search definition schema document.

<search> element

The <search > element attributes define the name of the stored procedure (spname), the major and minor versions, a label that identifies the search in the iConsole, a description of the search, and an HTML help page and image for the help button.

<parameter> element

In the MyEmailSearch example, the <parameters> collection contains a single <parameter> element with these attributes:

name

This can be used in to identify the parameter in a script. It does not need to match the argument name in the SP.

type

This parameter type must be compatible with the SP.

label

This specifies the text that is shown next to the search parameter in the Customize Search screen in the iConsole.

argpos

This specifies the order in which the parameter is listed in the arguments passed to the SP (see the CREATE line in the examples on page). This allows you to change the parameter layout in a search screen independently of the order they are passed to the SP.

<results> element

This defines the columns in the search results screen. In the MyEmailSearch example, there are two <column> elements for the event UID and e-mail title columns. Each <column> element has these attributes:

name

The column name must match the name in the SP.

type

This column type must be compatible with the database type.

label

This specifies the column heading shown in the Search Results screen in the iConsole.

More information:

Search Result Column Type Mappings (see page 166)
Parameter Type Mappings (see page 163)
Reference Information (see page 127)

Search Class

Pre-defined behavior for a search can be applied by simply specifying the class attribute of the <search > tag. For the current release, the following search classes are supported:

EventReview

Creates tools for reviewing events and enables the 'Quick View' pane on the iConsole search results page.

The Quick View pane also provides a link to the Context Search which provides a popup window with events in the context of the current event. Searches of this class are subject to Security ID processing if this is configured.

EventReport

Searches of this class are also subject to Security ID processing if this is configured.

Administration

Searches of this class only require an administrative privilege (see the table below).

Each class requires that the user has the correct privileges as defined in the following table:

Class	Type	Required Privileges
EventReview	Metadata	Events: Allow event searches AND Events: View captured data
	Content	Events: Allow content searches AND Events: View captured data
	Metadata	Events: Allow event searches AND Events: View captured data
	Content	Events: Allow content searches AND Events: View captured data
Administration	Metadata	Admin: Allow administration searches

Class	Type	Required Privileges
<None>	Metadata	Admin: Allow administration searches OR Events: Allow event searches
	Content	Events: Allow content searches

More information:

Security ID Processing (see page 167)

Parameter Types

The search can define a number of different parameter types:

date

A date selectable via a calendar dialog (see Date Specification (see page 39)).

date_range

A date range selectable from specified or predefined periods, or using calendar dialogs (see Date Specification (see page 39)).

icon

Image for display only.

label

Text for display only.

numeric

An integer value.

real

A floating point number.

text

A text string.

checkbox

A Boolean value.

list

A list of possible values. Either a single or multiple values may be selected. A variety of display formats are possible for multi-select lists (see Multi-Select List Parameters (see page 38)).

choice

A set of exclusive choices displayed as a collection of radio buttons.

lookup

A list dynamically populated by a search, allowing selection of one or more values (such as trigger names).

lookupbyid

Similar to lookup but provides a list of unique IDs to the search, rather than the displayed ‘user-friendly’ values (for example, users).

systemid

Provides the id of one of a predefined set of values (for example, user, group).

Multi-Select List Parameters

Parameters with type="list" used in conjunction with multiple="true" can be rendered in several different ways. Prior to version 12.5 the list was displayed as a drop-down list that allowed more than one selection. However, this provides challenges for accessibility so when requires="12.5" is specified the list is rendered either as a set of checkboxes (when there are 4 or fewer items) or as a 'shuttle' control which shows two lists with buttons to move items between the 'available' and 'selected' lists. The default behavior can be overridden if desired; when a list contains more than 4 items it can be rendered as a set of checkboxes by specifying attrs="shuttle:false" and the converse is true for rendering small lists as a 'shuttle'.

Lists can be specified with or without values and this affects the value passed to the SP.

When no values are specified for the list items, such as the following example:

```
<parameter name="lstX" type="list" multiple="true" argpos="1" >
  <option>One</option>
  <option>Two</option>
  <option>Three</option>
  <option>Four</option>
</parameter>
```

If the user selects a single value, the index of the item is passed to the stored procedure as an integer. For example, if the user selects 'Two' the value passed is 2.

But what happens if the user selects multiple values? So that a single integer value can still be passed to the stored procedure, the selected values are used to set bits in the integer (bitmap) and the value is made negative to indicate that it's a multiple selection. For example, if the user selects 'Two' (bit 2 = 2) and 'Four' (bit 4 = 8) the value passed is -10.

Since the value passed to the stored procedure is a 64 bit integer, this means that a maximum of 63 items may be included in this kind of multi-select list.

When (from version 4.7 onwards) the list options are given explicit values to be passed to the stored procedure. In order to avoid being mistaken for the first form, the option values must all be non-numeric.

```
<parameter name="lstY" type="list" multiple="true" argpos="1" >
  <option value="one">One</option>
  <option value="two">Two</option>
  <option value="three">Three</option>
  <option value="four">Four</option>
</parameter>
```

If the user selects a single value, the option value is passed to the stored procedure as a string. For example, if the user selects 'Two' the value passed is 'two'.

If the user selects multiple values these are passed as a comma-separated list. For example, if the user selects 'Two' and 'Four' the value passed is 'two,four'.

Note: If a comma-separated list of values is required even when some or all of the values are numeric (or default index values are used) this can be achieved by adding the attribute attrs="text" to the list parameter definition.

Date Specification

The two date parameter types are designed to simplify the task of creating date parameters, with dates converted automatically to and from the user's locale. Dates can be specified as fixed (absolute) or relative dates.

Single Date

To specify an absolute date for a date parameter the value should be specified as milliseconds since 1 Jan 1970 00:00:00 UTC.

A relative date can be specified by supplying a negative value; this is taken as the number of days before today. For example, -1 represents yesterday.

An alternative time zone can be specified if required, so that dates in (say) Pacific-time can be entered. This is achieved through the attrs attribute of the parameter which should contain two values separated with semicolons (;) or colons (:) as follows:

1. The offset (in minutes) from UTC on 21 December (for example. -480 for Pacific time)
2. The difference in this offset on 21 June due to DST corrections (for example, 60 for Pacific time)

A date parameter to enter times in Pacific time would look like this:

```
<parameter name="pacific" type="date" label="Pacific Date:" attrs="-480;60"
argpos="1"/>
```

Note: This technique has been adopted because there is no agreed standard for naming time zones. These two values can be used by the iConsole to establish the likely time zone so that the relevant parameters can be used (for example, DST dates). This is accurate most of the time but there are one or two time zones that have identical values, but differing DST dates so absolute accuracy cannot be guaranteed.

The value attribute which is passed to the stored procedure will be UTC but the report_value attribute will contain the uncorrected time for the time zone. A report may then format the report_value (without local conversion) to display the time in the required time zone.

Date Range

This parameter type is an aggregate type. That is, it creates several input objects in order to create the dates to be passed to the stored procedure. It allows the user to specify the range of dates that they are interested in using a number of different mechanisms:

- A 'Specified Period' before the current date, such as a number of days
- A 'Predefined Period', such as last month
- A custom range where the user manipulates the 'From' and 'To' dates directly.

By default, the date_range will be set to a specified period of 1 week before today, but a different default can be specified in the search definition. Although a fixed value for the dates can be specified (two values separated with a comma) this is not usually useful as the dates would be fixed and would have to be specified in internal format (milliseconds since 1 January 1970 00:00:00 UTC). Instead, the default state can be set by manipulating the attrs attribute of the parameter which controls the other input fields, for example, the selection of Specified Period or Predefined Period, and optionally an alternative time zone (see the Single date section above for details). The attrs attribute should contain up to six values separated with semicolons (;) or colons (:) as follows:

1. The type of date range: 1=Specified Period, 2=Predefined Period (selects the radio button)
2. The number of periods when Specified Period is selected
3. The type of period when 'Specified Period' is selected: 1=Day, 2=Week, 3=Month
4. The type of period when 'Predefined Period' is specified:

   1=All Dates, 2=Today, 3=This Week, 4=Last Week, 5=This Month, 6=Last Month, 7=This Quarter, 8=Last Quarter, 9=This Year, 10=Last Year, 11=Custom
5. The number of minutes from UTC on 21 December (e.g. -480 for Pacific time)
6. The number of minutes for DST (e.g. 60 for Pacific time)

As an example, to set the default to be the Predefined Period 'Last Month', with the default Specified Period of 5 days the parameter would be:

```
<parameter name="dates" type="date_range" label="Dates:" attrs="2;5;1;6" argpos="1"
/>
```

As an alternative, a custom date can be specified using relative dates as indicated by a negative value for the 'from' date. The relative dates represent the number of days before today for example -1 represents yesterday. So to represent the day before yesterday, a Predefined Period of 'Custom with the value "-2,-1" could be specified like this:

```
<parameter name="dby" type="date_range" label="Dates:" attrs="2;1;2;11"
value="-2,-1" argpos="1"/>
```

Note: If the attributes are set for a Predefined Period of 'Custom' and no value is specified, the search will default back to a Specified Period of 1 week.

Note: If you receive an error message containing ' Input string was not in a correct format' when running your search this indicates that either the value or attrs attributes are incorrectly formatted. The value attribute should be two numbers separated with a comma and the attrs attribute should be at least four numbers separated by semicolons or colons.

Special Parameters

Special parameters begin with a $ and are reserved for use by the iConsole. The following special parameters are supported:

$CONTROL

Controls the operation of unlimited searches.

$ROWLIMIT

Tells the stored procedure the maximum number of results required. Lowering this value can significantly reduce the time taken to execute the search. If the value specified exceeds the system defined row limit it is reduced to that value to prevent unnecessary processing.

$DEBUG

Indicates to the stored procedure that logging should be enabled for debugging the search.

More information:

Supporting Searches of Unlimited Size (see page 94)

Content Search Parameters

When defining a search against the content database, a stored procedure is not used. Instead, a number of predefined parameters and results columns can be used in the content search definition. These parameters and the associated results columns are listed in Search Definition Syntax. Extracts from an example content search definition follow:

Example content search definition

```
<search
     spname="Content_Search"  major="1"  minor="0"
     label="Content Search"  class="EventReview"  show_parameters="true"
     database="content"  help="help/htm/Content_Search.htm">

     <parameters>
     …
          <parameter name="txtSearchTerm"  type="text"  argpos="3" size="100" rows="5" colspan="10"
           value="" label="Search Term:" tooltip="Enter text to search for in the content"/>
          <parameter name="chkEE"  type="checkbox"  argpos="17" label="E-Mail Events:"
           value="true" align="left" tooltip="Select to get captured Email Events" report="false"/>
          <parameter name="lstEE"  type="list"  argpos="4" value="1" tooltip="Select to limit source
           of Email Events" report="false">
               <option>Internal and External</option>
               <option>Internal Only</option>
               <option>External</option>
          </parameter>
          <parameter name="txtNameMatch" type="text" argpos="9" colspan="3" label="Name Match:"
           size="90" tooltip="Match string for user names" onkeypress="javascript:NameMatch();"
           onchange="javascript:NameMatch();" report="false"/>
          <parameter name="txtSpecificMatch" type="lookup" argpos="10" colspan="2" label="Specific
Match:"
           size="80" lookup_function="NameLookup" tooltip="Select a list of user names"
           onkeypress="javascript:SpecificMatch();" report="false"/>
          <parameter name="dateRange" type="date_range" argpos="11" report_label="Date Range:"/>
          </parameters>
<results row_function="_EventDetails" ref="event_uid" ref2="doc_id"
help="help/htm/Content_Search_Results.htm">
               <column name="event_uid"  type="id"  hidden="true"/>
               <column name="doc_id"  type="id"  hidden="true"/>
               <column name="event_type"  type="icon"  width="3%"  label="Event Type"
hide_label="true"/>
               <column name="attachment"  type="icon"  width="3%"  label="Attachment"
hide_label="true"/>
               <column name="relevance"  type="text"  width="5%"  label="Relevance"/>
               <column name="subject"  type="text"  width="42%" label="Subject"  show_tooltip="true"
/>
               <column name="timestamp"  type="timestamp"  width="17%"  label="Timestamp"
show_tooltip="true" />
```

```
               <column name="participants"  type="text"  width="30%" label="Participants"
show_tooltip="true" />
               <tool name="similar" tooltip="More Like This" icon="similar-results.gif"
function="SimilarContentResults"/>
               <tool name="more" tooltip="More Results" icon="more-results.gif"
function="MoreContentResults"/>
               <script>
               'More like this' function goes here.
               </script>
         </results>
</search>
```

Chapter 8: Example Script Use in a Search Definition

The example below shows a simplified search definition and indicates how to incorporate scripts and functions. For example, in the <parameters> element, the function MyLookupFunction is included in a search parameter definition; this function allows the user to select a value from a popup or picker window when customizing a search.

Similarly, the <script> element in the <results> section includes three functions necessary to display search results in an Excel view; these functions are ShowExcel(), ResultsComponent() and LoadSpreadsheet().

- The first function, ShowExcel(), is called by a toolbar button in the Search Results screen and defined in the <tool> element. This function then calls LoadSpreadsheet(), which in turn references an object defined in ResultsComponent(). ResultsComponent() is a reserved function name; if defined, this function is called automatically.
- A separate toolbar button calls the ToolbarDelete() function, which provides 'delete row' functionality.

Note: Notice how the script definition is contained with a CDATA section so that it does not get interpreted as XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<searches
  <search
    <parameters>
      <parameter_group … />
      <parameter_line … />
      <parameter … />
      <parameter Lookup_function=" MyLookupFunction" />
      <script>
        <[CDATA[
        <script type="text/javascript" src="scripts/search-helper.js">
        </script>
        <script type="text/javascript">
        <!--
          function MyLookupFunction(name) { … }
        -->
        </script>
        ]]>
      </script>
    </parameters>
    <results>
      <column … />
      <column … />
      <column … />
      <tool
         function="ShowExcel"
      />
      <tool
        function="ToolbarDelete"
      />
      <script>
        <[CDATA[
          <script type="text/javascript">
            <!--
            function ShowExcel() { … }
            function ResultsComponent() { … }
            function LoadSpreadsheet() { … }
            -->
          ]]>
        </script>
      </script>
    </results>
  </search>
</searches>
```

Libraries of script functions may be accessed by the search definition by including the file containing the functions. In the above example the search-helper.js library has been included. Script libraries need to be accessible to the web server and should be installed into the Web/scripts folder.

The search-helper.js library is installed as part of the product and contains a number of useful helper functions.

Note: The script element must be ended with a closing </script> tag, even though the element is empty. The empty element format, closing the tag using /> will not work correctly, remember this is HTML syntax and is not well-formed XML.

More information:

Standard Scripting Functions (see page 167)

Chapter 9: Install a New Search

This involves the following steps:

1. **Load the SP into the database**

   Use any database tool or the CA DataMinder RunScript function.

   If using RunScript, modify the SQL file to conform to RunScript requirements. Specifically, if supporting both Oracle and MSSQL you must prefix the code with the appropriate database identifier, <ORACLE> or <MSSQL>, and terminate it with a semicolon.

   For Oracle, line terminators in PL/SQL must be escaped with a backslash unless the SQL is included using the RunScript EXEC_NATIVESCRIPT command. For example:

   ```
   RETURN MyCursor;     would become     RETURN MyCursor\;
   ```

   The syntax for the RunScript function is:

   ```
   wgninfra –localexec wigan/schema/Schema RunScript MyEmailSearch.sql
   ```

2. **Load the search definition file**

   In the iConsole:

   a. Browse to the Manage Searches screen and click Install Search  .

   b. Locate and install the XML file that contains the search definition. This adds the default searches to the Unpublished folder of the Manage Searches screen, ready for testing.

      Note: During installation, the search definition XML is validated against an XML schema document 'SearchDefinition.xsd'. The XML schema document is in the \Web\schemas subfolder in the CA DataMinder installation folder. In addition to validating that the correct elements and attributes are specified, the schema also applies constraints to some attributes. For example, the length of the search label attribute is limited to 64 chararacters. Search definitions that fail validation are not installed.

      Note: The embedded JavaScript in a search definition can be error-prone. Because JavaScript is an interpreted language, errors only manifest themselves at runtime. To reduce the possibility of errors in script, you can configure the iConsole to validate the script when the search definition is installed. This feature is normally turned off because it imposes an unnecessary overhead when installing production search definitions, but it is useful when developing new searches. To enable JavaScript validation, set the ValidateSearchJavascript registry value in the WebService hive to 'true'. Advanced users may want to alter the JavaScript compiler options using the SearchJavascriptValidationOptions registry value (the default is '/warn:3 /fast-'). Details about these registry values are outside the scope of this guide.

3. **Copy any help files and script files for the search onto each front-end Web server**

   If required, copy the .html and .gif help files to the location specified by the *help* and *help_icon* attributes in the XML search definition file. Typically, you have separate help pages for the search screen and search results screen.

   Copy the script files to the \Web\scripts folder

   Do this on each front-end Web server.

4. **Test and publish the new search**

   In the iConsole:

   a. Browse to the Administer Searches, Searches screen and click Test  to test the parameters in the search definition.

   b. Click the Actions drop-down for the search and select Edit to see the properties page for the search. This page contains all the parameters defined for the search.

   c. Select Run to test the execution of the search stored procedure.

   d. When testing is complete, return to the Administer Searches, Searches page.

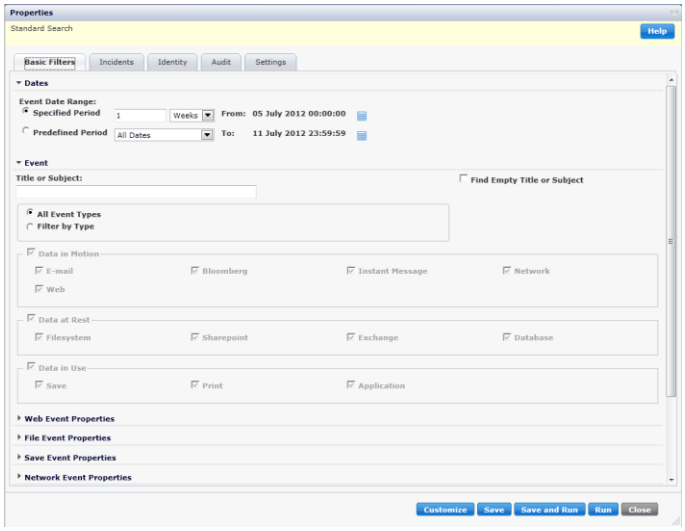
e. Click the Actions drop-down for the search and select Publish to make the searches available to all users logged on to the CMS.

Chapter 10: Configuring the Search Parameter Layout

For simple queries, the default layout for a search screen may be adequate. For more complex queries with many parameters, you may require more control over the layout. This section describes the facilities available to do that.

All search parameters are arranged on a grid and you have control over the location and size of a parameter within that grid.

Most of the supported layout methods are demonstrated by the iConsole Standard Search. This has many parameter groups, and each group has a variety of parameters and subgroups.



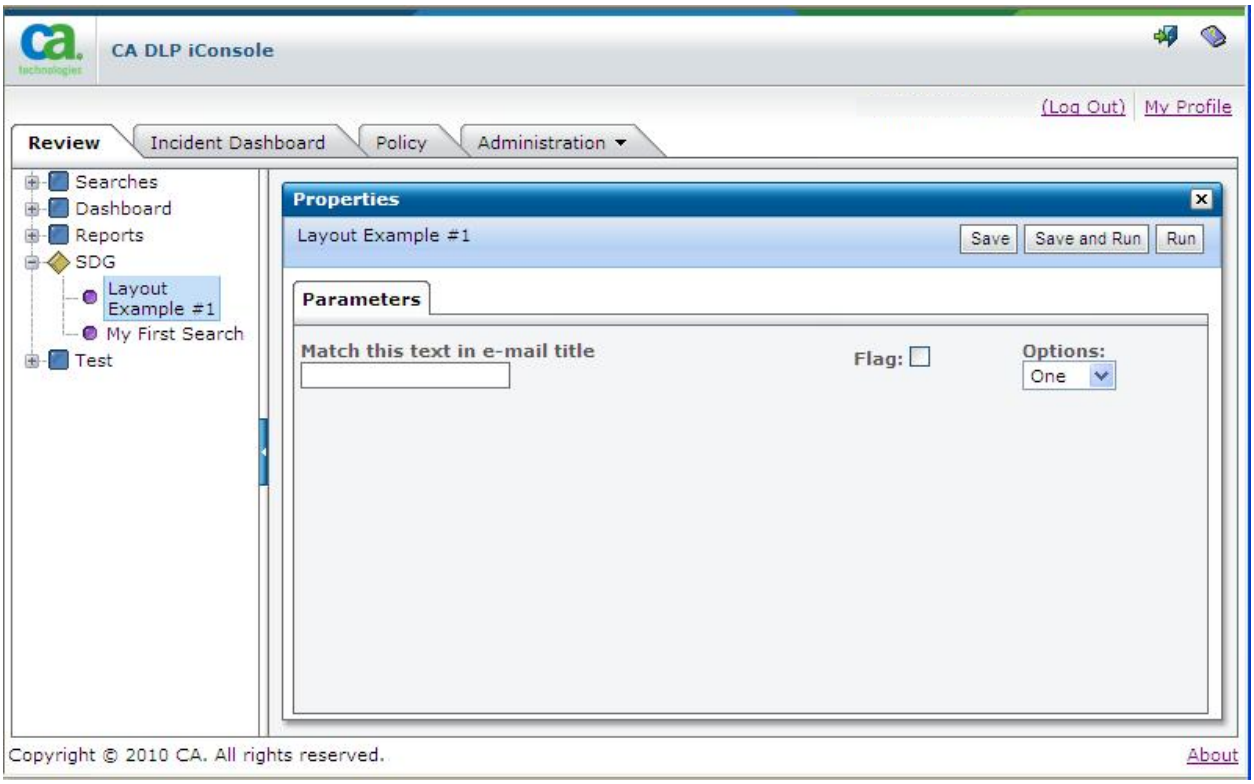
Note: Previous versions of the iConsole laid out parameters on separate rows unless contained within a <parameter_line> element. This scheme is still used unless the Default element contains a 'requires' attribute specifying version 12.5 or later. We recommend that you specify *requires="12.5"* in new searches and reports.

This section contains the following topics:

<parameter> Element

In the search definition file, the <parameter> element on its own displays each parameter in the next cell of the grid. By default each parameter will appear in the next column of the grid. For example, these three <parameter> elements generate the search screen below:

```
<parameters>
  <parameter
    name="txtMatch"
    type="text"
    label="Match this text in e-mail title"
    argpos="1"
  />
  <parameter
    name="chkFlag"
    type="checkbox"
    label="Flag:"
    argpos="2"
    align="right"
  />
  <parameter
    name="lstOptions"
    type="list"
    label="Options:"
    argpos="3"
  >
    <option value="1">One</option>
    <option value="2">Two</option>
    <option value="3">Three</option>
  </parameter>
</parameters>
```



Adjusting the Parameter Layout

Achieving an optimal layout can be a challenge, as the width of each column in the grid is determined by the widest <parameter> element in that column. This default behavior can be modified using <parameter> attributes, including:

col

Defines the grid column in which the parameter should start.

colspan

Specifies that a parameter is to span several columns.

row

Defines the grid row in which the parameter should start.

rowspan

Specifies that a parameter is to span several rows of the grid.

width

Changes the cell width, typically as a percentage of available space.

size

Specifies the number of characters that the parameter should display.

align

Specifies how the parameter is to be aligned within its cell.

valign

Specifies how the parameter is to be aligned vertically within its cell.

To lay out the parameters in the example above vertically we can use the ‘col’ and ‘row’ attributes to indicate how the parameter should be shown:

```
…
<parameters>
        <parameter
                name="txtMatch"
                type="text"
                label="Match this text in e-mail title"
                argpos="1"
                col="1"
                row="1"
        />
        <parameter
                name="chkFlag"
                type="checkbox"
                label="Flag:"
                argpos="2"
                align="right"
                col="1"
                row="2"
        />
```

```
        <parameter
                name="lstOptions"
                type="list"
                label="Options:"
                argpos="3"
                col="1"
                row="3"
        >
                <option value="1">One</option>
                <option value="2">Two</option>
                <option value="3">Three</option>
        </parameter>
</parameters>
…
```

In fact the 'row' attributes could be omitted because col="1" will automatically start a new row.

Example search screen: Based on the <parameter> elements with col/row specified

More information:

<parameter> Element (see page 131)

<parameter_line> Element

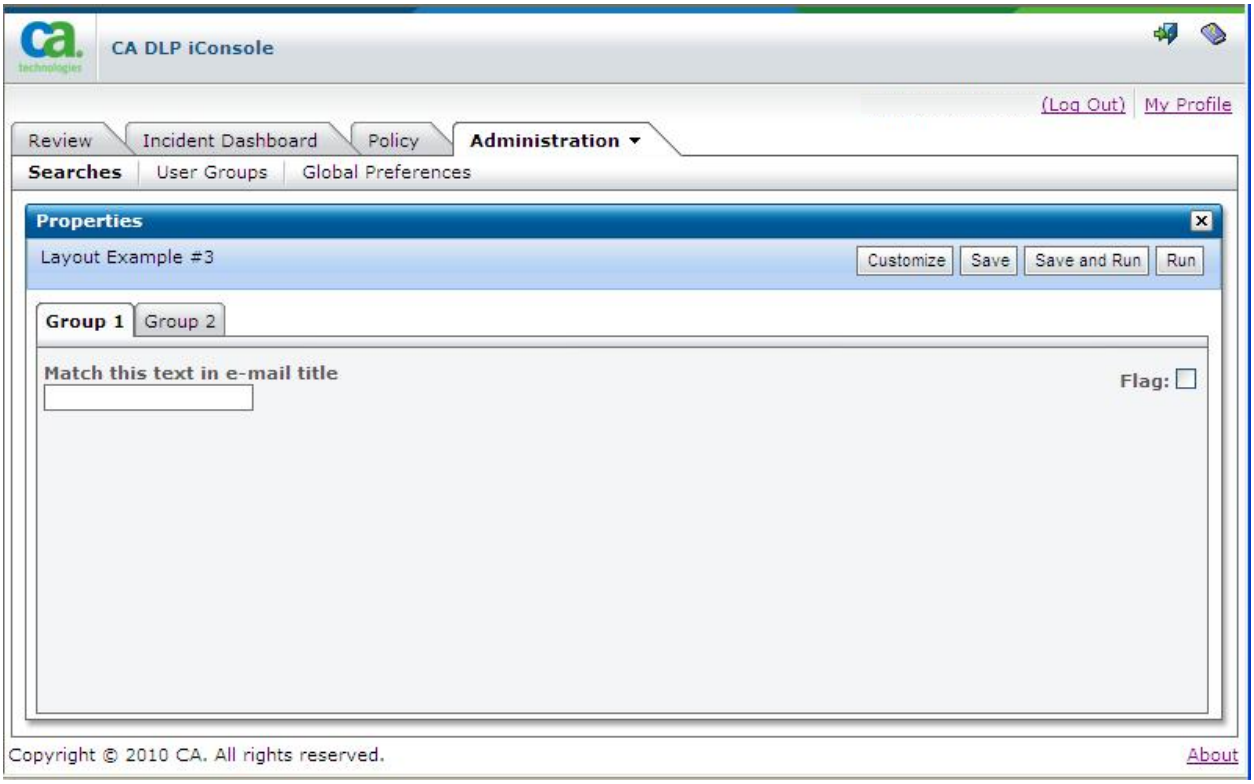
The <parameter_line> element was used for laying out parameters in earlier versions of the iConsole but is deprecated in this release. Specify requires="12.5" in the Default element to use the new scheme.

<parameter_group> Element

By default parameters are shown in a sub-tab labeled ‘Parameters’. The <parameter_group> element allows more control over the grouping of parameters where collections of <parameter> elements may be grouped under multiple sub-tabs. For example, this xml creates two sub-tabs labeled "Group 1" and "Group 2" each containing some of the parameters from a previous example:

```
<parameters>
  <parameter_group label="Group_1">
      <parameter
        name="txtMatch"
        type="text"
        label=" Match this text in e-mail title"
        argpos="1" />
      <parameter
        name="chkFlag"
        type="checkbox"
        label="Flag:"
        argpos="2"
        align="right" />
  </parameter_group>
  <parameter_group label="Group 2">
      <parameter
        name="lstOptions"
        type="list"
        label="Options:"
        argpos="3" >
        <option value="1">One</option>
        <option value="2">Two</option>
        <option value="3">Three</option>
      </parameter>
  </parameter_group>
</parameters>
…
```

Example search screen: Based on the <parameter> elements within <parameter_group> elements

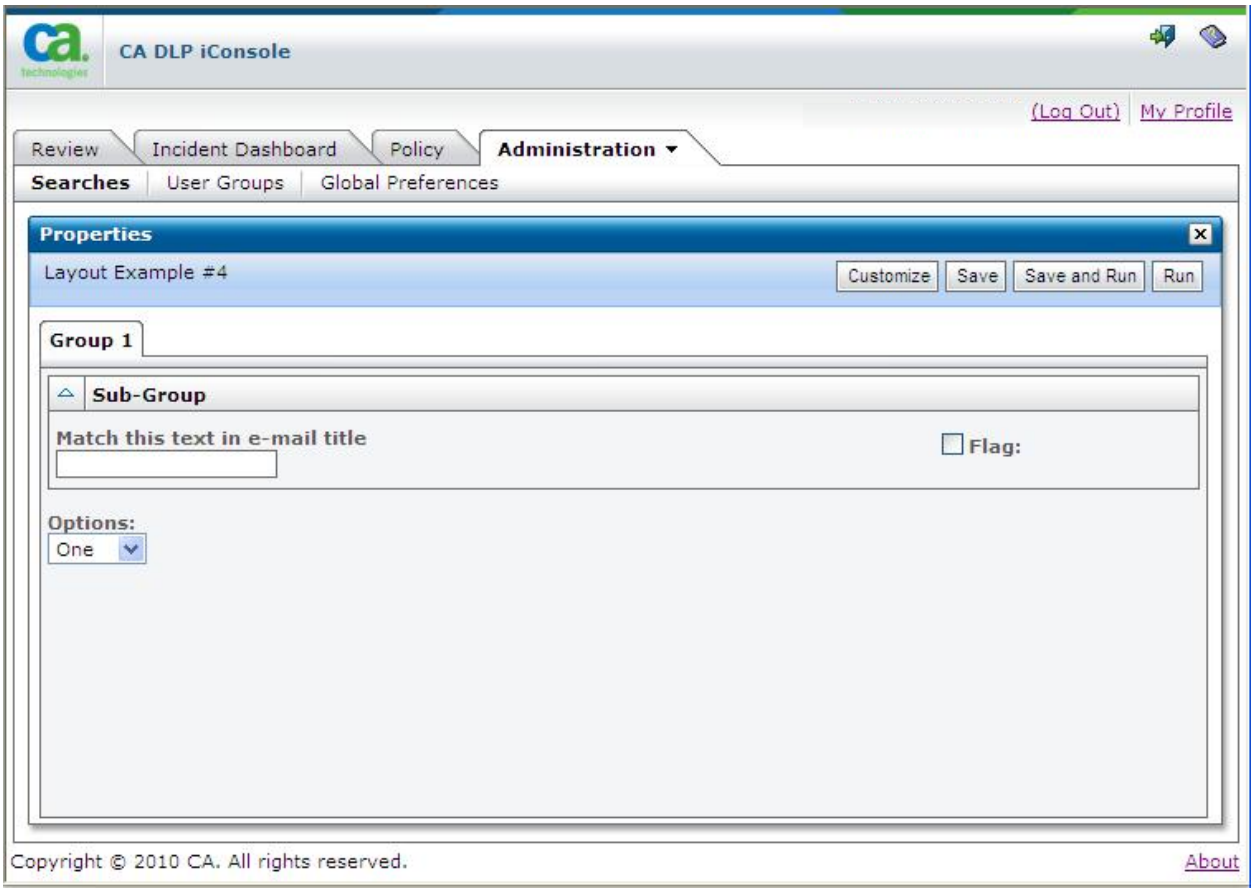

CA DLP iConsole
(Log Out)
My Profile
Review
Incident Dashboard
Policy
Administration
Searches
User Groups
Global Preferences
Properties
Layout Example #3
Customize
Save
Save and Run
Run
Group 1
Group 2
Match this text in e-mail title
Flag:
Copyright © 2010 CA. All rights reserved.
About

<subsection> Element

Parameters within a <parameter_group>may be further grouped using the <subsection> element as a container for the parameters to be included in the sub-group. This element creates a panel inside the tab which can also be collapsed to take up less space, the initial collapsed state can be set using the 'collapse' attribute. The following XML illustrates the use of <subsection>.

```
…
<parameters>
        <parameter_group label="Group 1">
                <subsection name="subgroup" label="Sub-Group" col="1" collapse="false">
                        <parameter
                                name="txtMatch"
                                type="text"
                                label=" Match this text in e-mail title"
                                argpos="1" />
                        <parameter
                                name="chkFlag"
                                type="checkbox"
                                label="Flag:"
                                argpos="2" />
                </subsection>
                <parameter
                        name="lstOptions"
                        type="list"
                        label="Options:"
                        argpos="3"
                        col="1" >
                        <option value="1">One</option>
                        <option value="2">Two</option>
                        <option value="3">Three</option>
                </parameter>
        </parameter_group>
</parameters>
…
```

Example search screen: Based on the <subsection> element



Dynamic Parameters

There are two types of parameter that use dynamic content: List parameters and Lookup parameters.

List Parameters

These can retrieve list values from the database using the spname attribute to call a stored procedure (SP). If the SP is in an Oracle package, you must also specify the sppackage attribute. The SP needs no parameters and returns two columns: the value of the list items and the text for the list items. An example is the WgnWCAuditField1List SP used by the Standard Search in the iConsole.

Note: The list of values retrieved from the SP can be cached for more efficient storage on the Web service.

Note: Where the parameter is a list of values for an audit field, the label for the parameter can be substituted with the audit field name specified in the Administration Console. This is achieved by adding attrs="subs" to the <parameter> element.

More information:

Search Definition Syntax (see page 166)

Lookup Parameters

Lookup (and lookupbyid) parameters allow the reviewer to look up values in the database to use as parameters to the search. The lookup procedure is invoked by a function. There are two standard functions for retrieving database values: Popup and Picker. These are described below.

The lookupbyid type passes the IDs of the selected items to the stored procedure and the textual values are passed as the report_value for inclusion on report headers. If the parameter name is prefixed with the reserved name '$GROUPS' the report value is populated with a list of the user's management groups when no groups are selected

Note: If the parameter name is prefixed with the reserved name '$GROUPS' the report value is populated with a list of the user's management groups when no groups are selected. The value passed to the stored procedure is still blank.

Note: If the user has the Admin: Disable Security Group Filtering privilege, the report_value is set to the top level group path.

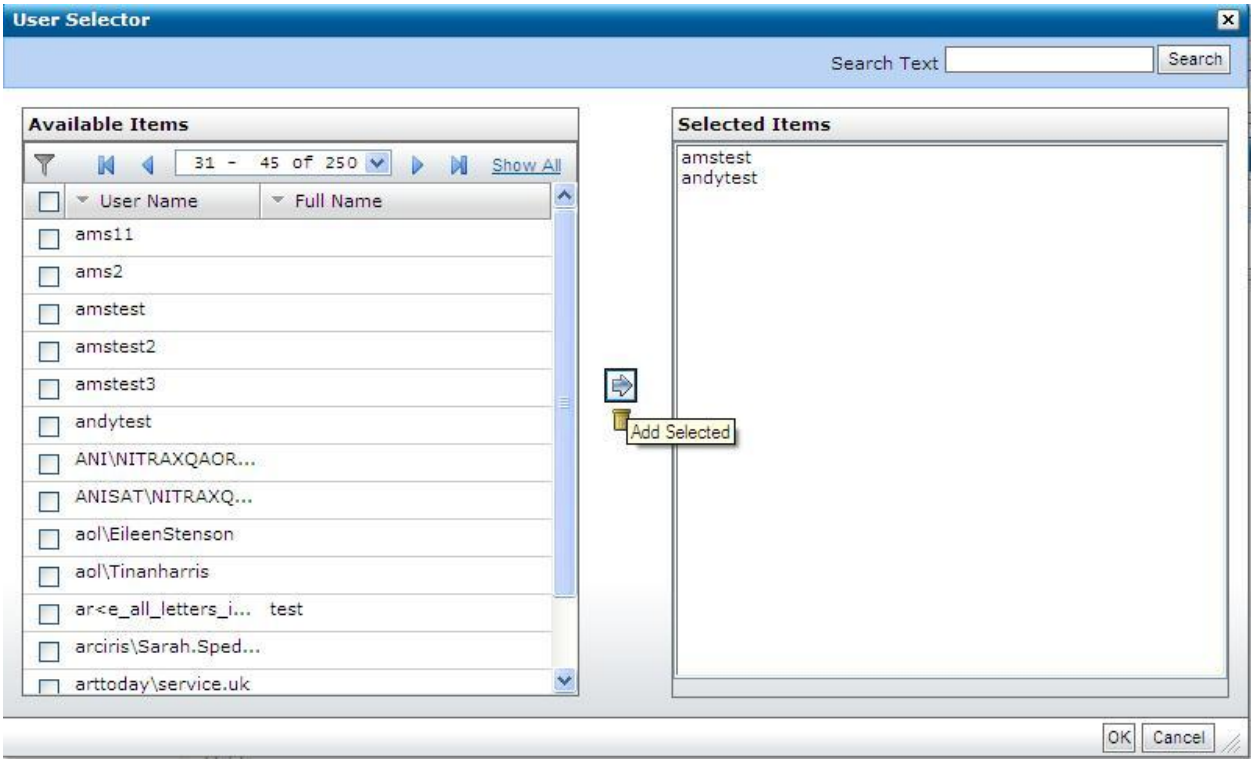
More information:

Lookup Parameter Syntax (see page 65)

Popup

This provides a popup window to gather selected items from a potentially large list. The list of available items is initially empty; the user can enter a search string before clicking the 'Search' button. This runs an SP to retrieve the results. Several columns of results may be returned, but only values from the first column are used for the 'lookup' parameter. One or more values may be selected and added to the existing list of values. Optionally, the values returned may be from a hidden column; this is useful when items need to be selected by name but searching must be performed using a unique ID for the item (e.g. groups).

Example popup: User Selector on Standard Search page

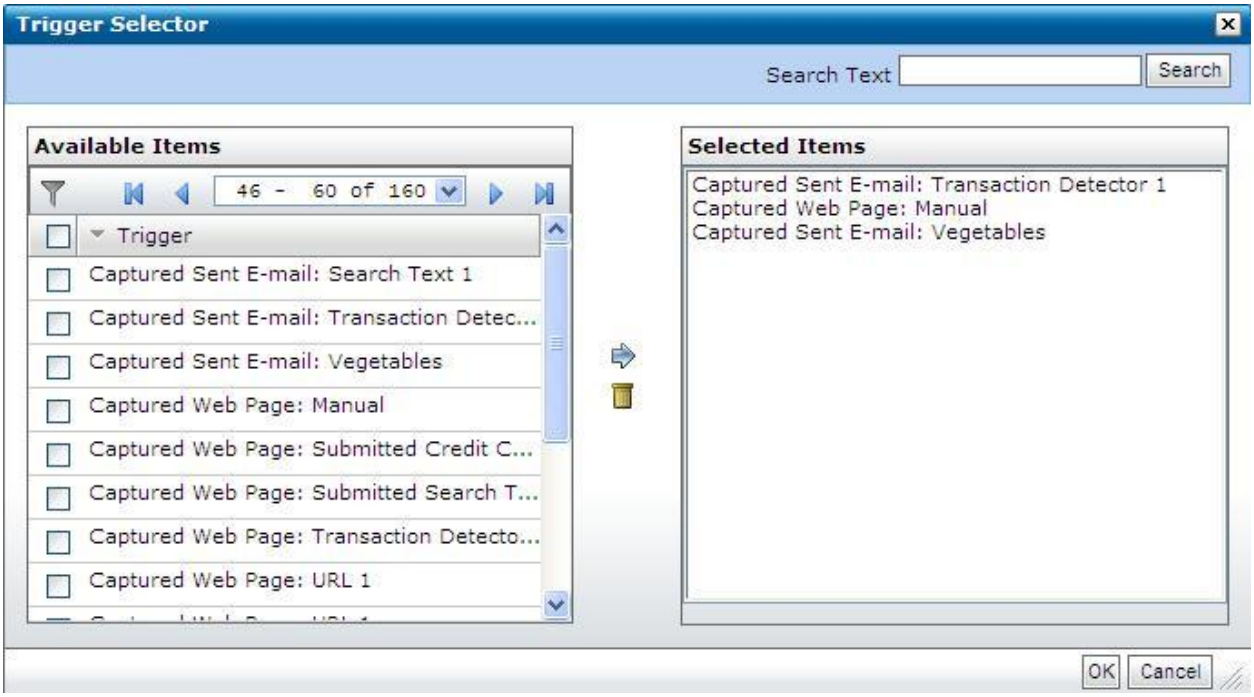


Picker

This is a simple alternative to the 'popup', where a small list of alternatives is expected. No parameters are initially passed to the SP, which is executed immediately so the list is populated as soon as the dialog appears. Usually, only one value can be selected, but multiple selections can be allowed if the 'multi' argument is specified in the lookup function. By default, there is no search capability, but this can be added by specifying the 'filter' argument.

Items from the Available Items list can be selected by clicking with the mouse and added to the Selected Items list by clicking the ‘arrow’ button. Selected Items can be removed using the ‘trash can’ button.

Example picker: Trigger Selector on Standard Search page



Lookup Parameter Syntax

Both Popup and Picker use an iConsole search to retrieve the list of items, this is generally stored as a 'System' search. The lookup SP is invoked by a user-defined script function (MyUserLookupFunction in the example below). This function permits a reviewer to select the required value(s), from a popup or picker window.

The function name is specified by the lookup_function attribute in the <parameter> element. The function takes one argument (name in the example below). This argument identifies the 'lookup' parameter field, that is, the parameter field that will hold the value selected by the user in the popup or picker windows. The PopupWithHiddenField parameter passes an additional argument to the lookup_function to identify the field that will receive the hidden IDs. In this case the selected names are for display only; it is the IDs that will be passed as a parameter to the search.

```
<parameter
  …
  lookup_function="MyUserLookupFunction"
  …
  …
  />
  <script type="text/javascript">>
    function MyUserLookupFunction(name)
    {
      Popup('User Selector','User_Picker',1,0,name, true);
    }
```

Lookup Parameter Calling Sequence

The popup and picker functions have the following calling sequences:

```
Popup('title','name','major','minor','lookup field', 'nameflag', 'popflag',
'concsep');
```

```
PopupWithHiddenField('title','name','major','minor','lookup field', 'hidden',
'nameflag', 'popflag', 'concsep');
```

```
Picker('title','name','major','minor','lookup
field','multi','filter','nameflag');
```

where:

title

Specifies the title to be displayed on the popup dialog.

name

Specifies the name of the lookup search to call.

major

Specifies the major version number of the lookup search.

minor

Specifies the minor version number of the lookup search.

lookup_field

Specifies the parameter field that will receive the selected items.

hidden

Specifies the parameter field that will receive the hidden IDs of the selected items

multi

Optional. When true, specifies that multiple selections are permitted (default is false).

filter

Optional. When true, specifies that results can be filtered by specifying a search string (default is false).

nameflag

Optional. When false (default), specifies that the name argument is the SP name (this is to support previous versions of the iConsole); when true, name is the search name.

popflag

Optional. When true (default false), the picker is pre-populated by running the associated search.

concsep

Optional. When not blank (default ''), multiple column values are concatenated using this separator.

Note: When the filter option is specified, the search should accept a string parameter to be used as the filter text. The search can return multiple columns and these are displayed in the picker; the filter is applied to all columns

Adjusting the Display

Text Styles

The overall style of the display (for example, text colors or fonts) is determined by the prevailing style sheet for the iConsole, but these may be overridden using the style attribute to emphasize a particular parameter. For example, you may want to display bold text in a particular column in the Search Results screen. The format of this attribute follows the Cascading Style Sheet Specification (CSS) and is outside the scope of this document.

Helper Functions

In addition to parameters that specify the values to be passed to the SP, the search definition may also specify labels to provide guidance and 'helper' parameters that can be used to construct the actual parameters. These helper functions are generally used in conjunction with scripting to derive, or calculate values for hidden parameters. Any script is specified using the script attribute of the <parameters> tag and uses the JavaScript format.

Hidden Parameters

To simplify the search screen layout, infrequently used parameters can be hidden using the hidden attribute. Administrators can make hidden parameters available by using the Administer Searches | Searches page to edit the properties of the search. The Customize button allows hidden parameters to be shown (or vice versa) and the search may then be saved as a derived search.

Chapter 11: Configuring the Search Results Layout

The <results> element of the search definition file should define, for each column, its name and type. It's also useful to specify a label to be used as the column heading as this is generally more user-friendly than the column name:

```
<results>
     <column
          name="event_uid"
          type="id"
          label="Event ID"
     />
     <column
          name="title"
          type="text"
          label="E-Mail Title"
     />
</results>
```

The column name must match the name in the SP, and each column type must be compatible with the database type. For details about the <column> element and its attributes, see XML Elements (see page 33).

Note: If no columns are defined in the search definition the columns will be derived from the results returned from the stored procedure.

This section contains the following topics:

Applying Styles to Rows, Columns and Cells

The styling to be applied to particular rows, columns or cells of the results can be controlled through the search definition and the stored procedure.

The style for every value in a column (including the heading) is defined using the style attribute of the column element, and can be any valid CSS style.

Controlling the style for individual rows is more complex as each row needs to be assigned a style based on some condition. Because of this, the style has to be defined within the stored procedure and returned in the results. A new column is therefore included in the results containing the name of a CSS class for each row. The name of this 'row_style' column is identified using the row_style_ref attribute of the results element.

Note: Default style settings for cells will take precedence over the row style. To force the row style to override the default cell styles, an additional CSS selector is required to apply the class to the td element. For example:

```
.myrowclass, .myrowclass td { …; }
```

If required, you can even control the styles of individual cells within the results. This works in a similar manner to the row styling, but an additional col_style column is required for each column of results that needs custom styling. The name of the 'col_style' column is identified using the col_style_ref attribute of the column element.

Row and cell styling will require corresponding CSS classes to be defined in the style-sheet.

Note: The precedence rules for CSS can be complex and it is sometimes necessary to force the required style to be applied by using the !important rule. For example:

```
.myclass {font-size: 20pt !important;}
```

Optional Columns in the Results

It is possible to define some of the results columns as optional. The columns required can then be selected from the Search Properties page. Optional columns may be defined as initially shown or initially hidden.

Columns are defined as optional by including the 'display' attribute in the column definition. When set to true, the optional column will be included in the results by default, when set to false, the column will not be included by default. If the display attribute is not given, the column is mandatory and will always be included in the results.

Note: All optional columns must be given a width. If only the columns initially displayed are given widths, and these add up to 100%, then the other optional columns could be displayed with zero width. The total column widths can exceed 100% and will be scaled accordingly.

More <column> Attributes

The following column attributes are also supported:

width

Specifies the column width, typically expressed as a percentage of available space (for example, width="30%").

units

Appends units to column values in the Search Results screen. This attribute is typically used to indicate that values are percentages (that is, units="%").

show_tooltip

Specifies whether the full content of a value is displayed as a tooltip (show_tooltip="true"). This is useful for long values that may exceed the available space.

icon_class

Specifies the subfolder containing the image files for icon columns (for example, icon_class="MyCustomIcons"). This subfolder is on the iConsole web server and must be directly below the \images\dlp\main\standard folder in the CA DataMinder installation folder. Place high visibility icons in a similarly named folder below \images\dlp\main\HiVis.

Examples:

```
C:\ProgramFiles\CA\CA DataMinder\Web\images\dlp\main\standard\MyCustomIcons
C:\ProgramFiles\CA\CA
DataMinder\Web\images\dlp\main\standard\HiVis\MyHiVizIcons
```

hidden

Specifies whether the column is hidden in the Search Results screen (hidden="true"). Hidden fields are useful as reference fields for 'link' functions (see below).

link_function and ref

These attributes together specify that column values are hyperlinked. When clicked, the column value invokes a function, say, to display a dialog containing details about the e-mail sender (for example, link_function="_UserDetails" ref="user_uid").

primary_key

Specifies that a column is identified as the primary key for row level operations. This attribute is typically used when removing irrelevant or uninteresting events from a set of search results. Only one column can be specified as the primary key. (primary_key="true").

Actions When a Row Is Clicked

You can configure the Search Results screen so that an action is invoked when a reviewer clicks an individual event (that is, when reviewer clicks any row in the Search Results screen). You specify the action by using the row_function attribute of the <results> tag. The associated ref attribute defines the reference column. For example, this is used by the Standard Search in the iConsole to show details about the selected event:

```
row_function="_EventDetails" ref="event_uid"
```

Multiple Row Operations

Here, 'multiple row operations' refers to operations on multiple events in the Search Results screen. Typically, you can configure tools on the iConsole toolbar for performing bulk operations e.g. auditing. You can enable a predefined set of tools by a specifying a class for the search or for specialized functions, explicit tools may be defined.

A check box is automatically added to each event, allowing it to be selected for a multiple row operation.

Note: Event check boxes are not displayed if no tools are enabled.

Predefined Tools

Currently, only the EventReview class creates a set of predefined tools. It creates up to five tools for bulk reviewing of events and enables the 'Quick View' pane on the iConsole search results page. The behavior of the tools can be configured using the Edit Audit Options function in the Administration console.

If no class is specified then tools for multiple row operations must be defined explicitly (see below).

More information:

Search Class (see page 35)

Defining Explicit Tools

Do this by using the <tool> element of the <results> element. Briefly, the attributes are:

name

Specifies an identifier for the tool.

tooltip

Specifies the tooltip text that displays when the mouse pointer hovers over the button.

icon

Specifies the icon image for the tool.

function

Specifies the function to call to process the rows.

More information:

Standard Scripting Functions (see page 167)
Search Definition Syntax (see page 166)

Chapter 12: Reports

The iConsole search mechanism incorporates a simple reporting capability. The report can include headings along with the parameters used to generate the report.

For simple reports that only return one record the results can be displayed as a summary rather than as a table with one row.

This section contains the following topics:

Report Attributes

To specify a report and display the underlying search parameters, add these attributes to the <search > element:

```
report="true"
```

```
show_parameters="true"
```

Suppress Individual Parameters

You may not always want to display all of the parameters. To suppress the display of a parameter, add the following attribute to the associated <parameter> element:

```
report="false"
```

Custom Parameter Labels

By default, parameters are identified on the report page by their label attribute (or by their name attribute, if no label is specified). If you want to display an alternative label for a parameter, add the following attribute to the <parameter> tag, for example:

```
report_label="My parameter label"
```

Chapter 13: Support for Web Components (Deprecated)

In order to provide reviewers with more flexibility when analyzing iConsole search results, the XML schema allows third-party Web components such as Microsoft Office Web Components to be embedded into the Search Results page. This gives reviewers greater power to manipulate search results (for example, in an Excel spreadsheet) and to export search results for further analysis outside of the iConsole.

Support for Web Components in the iConsole Search Results page is provided through scripting in the <results> element of the search definition. The search definition requires three components:

- A function to create the required Web component (for example, a spreadsheet view of the search results)
- A toolbar button and associated function to invoke the Web component
- A function to load the iConsole search results into the Web component

The necessary functions and elements to add support for a Web component to a Search Results page are described on the following pages.

Note: Third-party Web components may require licensing from the vendor to enable full functionality. These components are not provided by CA Technologies.

Note: Microsoft Office Web Components are a collection of COM controls that can be used to view spreadsheets, charts, and databases on the Web.

Note: Office 2007 does not support OWC and upgrading may remove OWC from your workstation. To continue using OWC after upgrading to Office 2007 you will need to download OWC10 or OWC11 from the Microsoft Download Center and re-install it.

Note: This capability is deprecated and may not be supported in future versions of the iConsole.

This section contains the following topics:

Defining Results Script

Any script functions must be defined in a <script> element within the <results> element, as shown below. .

```
<results>
  <script>
    <![CDATA[
      <script type="text/javascript">
      <!–
        // Your script here
      -->
      </script>
    ]]>
  </script>

</results>
```

Toolbar Button to Show Excel Spreadsheet

To create a toolbar button to invoke the Web component, add a <tool> element. The example below creates a 'Show Excel View' button to display search results in a spreadsheet:

```
<results>
...
  <tool
        name="excel"
        tooltip="Show Excel View"
        icon="excel.gif"
        function="ShowExcel" />
<script>
      <![CDATA[
              <script type="text/javascript"
                      src="scripts/show-excel.js">
              </script>
      ]]>
</script>
                    <results>
```

This calls the function SE_ShowExcel() provided in the show-excel.js library.

The SE_ShowExcel() function loads data into the component, displays the component and hides the results table. That is, it loads search results into a spreadsheet and displays the spreadsheet, and hides the standard table of search results.

Built-In Functions

The SE_ShowExcel() function can be examined to see how the capability is implemented. It depends on the following built-in functions:

GetResultsComponent();

This function retrieves the container for the OWC component.

ShowResultsComponent(display);

Shows the OWC component container and hides the results table when 'display' is true. Shows the results table and hides the OWC component container when false.

GetResultsTable();

This function retrieves the standard results table so that the data can be copied into the OWC component.

This process can only extract the results that are displayed on a single iConsole search results page. Therefore you typically override the default page size (25 results) by modifying the page_size attribute of the <results> element (see page 141). To ensure that all results are captured, set page_size to be the same as the MaximumResultsSetSize registry value on the application server. (This registry value specifies the maximum number of results returnable by an iConsole search and defaults to 1000; see the *Platform Deployment Guide* for details).

Note: Setting page_size to a large value has a negative impact on search performance. Long search result pages take longer to display and can cause a time-out.

Chapter 14: Custom Results Formatting

By default, the results of a search are formatted in tabular form (except for reports with a single result which may be formatted as a summary list).

Occasionally the tabular format is not suitable and a more complex layout is required. The search writer can define the following:

- A custom format to be used in place of the table on the search results page and refer to it using the format attribute of the results element.

Note: In earlier versions of the iConsole there was an option to replace the search results page using a custom XSL transform. The redesign of the iConsole no longer supports this approach and so this option is no longer available.

This section contains the following topics:

Custom Format

Defining a custom format allows just the table of results to be replaced with a layout of your choice. The custom format is implemented by an XSL transform that transforms the raw search results into the desired report layout. Knowledge of the raw search results format is therefore essential.

Example Report

In the following example, the search returns a list of users with their group and role. The results are formatted with a heading for each role and a table of users and groups below each heading.

The XSL transform is located by the format="$doc:user-role-summary.html" attribute. This breaks down as follows:

$doc

Indicates that this format produces a complete report document (previous iConsole versions allowed an html fragment to be produced; this facility is no longer supported)

user-role-summary

Identifies the name of the transform file, in this case user-role-summary.xsl located in WebService\transformations\report-formats.

.html

Defines the type of report document produced. In this case an HTML page.

The reportHeader class may be specified to use the standard report heading styles.

Example Search Definition

```
<search
     sppackage="TestSearches"
     spname="User_Role_Summary"
     major="1"
     minor="0"
     label="User Role Summary"
     description="Demonstrator for summary report."
     category="Test"
     report="true"
>
     <parameters/>
     <results format="$doc:user-role-summary.html">
          <column type="text" name="role" label="User Role"/>
          <column type="text" name="groupname" label="Group Name"/>
          <column type="text" name="username" label="User Name"/>
     </results>
</search>
```

Example Custom Format Transform

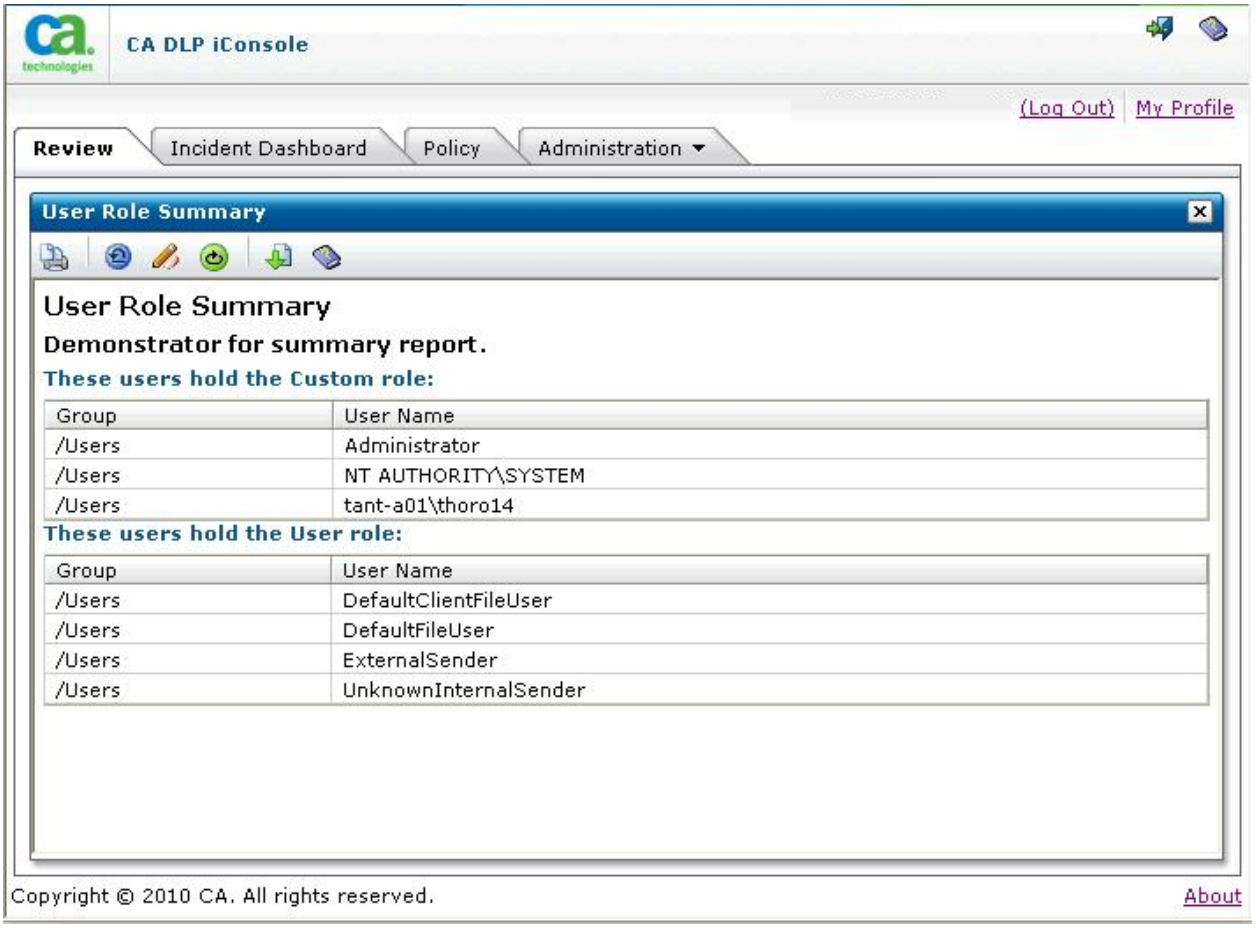
The custom format is produced by the XSL transform user-role-summary.xsl:

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="1.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
     <xsl:import href="../common.xsl"/>
     <xsl:output method="html" indent="yes"/>
     <xsl:template match="/">
          <html>
                <head>
                      <base target="_self"/>
                      <script type="text/javascript" src="scripts/common.js"/>
                </head>
                <body>
                      <div class="reportsHeader">
                                  <h1>
                                    <xsl:value-of select="//search/@label"/>
                                  </h1>
                                  <h2>
     <xsl:value-of
select="//search/@description"/>
                                  </h2>
     <xsl:for-each select="//row[not(role=preceding-sibling::row/role)]/role">
     <xsl:variable name="role" select="text()"/>
                                 <h3>These users hold the <xsl:value-of select="$role"/> role:</h3>
                                      <table id="tblUserRoleSummary" cellpadding="0"
cellspacing="0" border="0">
                                      <thead>
                                      <th>Group</th><th>User Name</th>
                                      </thead>
                                      <tbody>
      <xsl:for-each select="//search/results/row[role=$role]">
                                     <tr>
                                     [assign the value for TD in your book] <xsl:value-of
select="groupname"/></td>
                                     [assign the value for TD in your book]<xsl:value-of
select="username"/></td>
                                     </tr>
      </xsl:for-each>
                                     </tbody>
                                     </table>
                                     </xsl:for-each>

                                     </div>
                                     </body>
                                     </html>
                                     </xsl:template>
                                     </xsl:stylesheet>
```

Custom Format Results

Search Results page showing the custom format



Post-Processed Documents

To create reports in other formats it may be necessary to post process the output of the report transform because it isn't a simple matter to produce the format using XSLT.

A good example is the PDF format. To produce a report in PDF format an XSL Transform should be written to generate output in XSL-FO format as an intermediate step. This then needs to be post-processed using an external tool such as 'fop'. The iConsole needs to be configured to use this post-processing for the pdf format.

More information:

Customizing Results for Display or Download (see page 98)

Chapter 15: Displaying Aggregate Totals

Including aggregate totals in the search results returned by the stored procedure is problematic because only a single resultset is returned and the totals will not be distinguishable from regular data. The row of totals will be subject to sorting and filtering, confusing the results display. To overcome this limitation it's possible to request aggregate totals for a search by specifying show_totals="true" on the results element of the search definition. By default this will create a sum for every numeric, quantity or real column. The following functions are supported:

Avg

Calculate an average for the column.

Count

Count all the non-blank rows in the column.

Max

Show the maximum value for the column.

Min

Show the minimum value for the column.

StDev

Calculate the standard deviation for the column.

Sum

Add up all the values in the column.

Var

Calculate the variance for the column

The Count function can also be used for non-numeric columns.

To specify an alternative function, use the total_function attribute on the column element. For example, total_function="avg".

Note: If no total is required for a numeric column specify total_function="none".

The totals for the whole resultset will be displayed in the last line of results. If the results are paged; that is, the page does not contain all results, an additional line of totals will be included as the penultimate line showing the aggregate totals for the page. The style of the totals rows can be customized using the #trResultsPageTotal style for the page totals and the #trResultsTotal style for the overall totals.

Note: The aggregate functions for the grand totals will also be included in an Excel download of the results.

Chapter 16: Derived Searches

In order to provide customization of an existing, general purpose search, it is possible to create derived searches. A derived search references an existing search but allows certain features (such as parameter values and labels, and whether parameters are shown) to be overridden.

A derived search has similarities with a user's saved search except that a derived search can reference an unpublished baseline search, and the derived search may be published making it visible to all users.

This section contains the following topics:

Create Derived Search

The simplest way to create a derived search is as follows:

1. Select the baseline search from the Administer Searches | Searches page and select Actions | Edit.
2. Alter the values of parameters to the required values.
3. Show or hide the parameters as required. Use the Customize button to bring up a shuttle control showing hidden and unhidden parameters.
4. Click Save and specify a name and description for the derived search.

Edit Derived Search

More detailed customization, such as changing parameter labels, can only be done manually by editing the XML file:

1. Select the derived search that you wish to customize further and click the Actions | Export button.
2. Specify a name for the exported XML file and save it.
3. Edit the XML, changing attributes as required.
4. Re-Install the modified XML file.

Customizable Attributes

The attributes that can be overridden on the <search > element by specifying them in a derived search definition are listed below

```
class           max_rows
description     quick
help            refresh
help_icon       show_parameters
label           target
```

The <parameter> element attributes that can be overridden are:

```
attrs           size
cache           spname
constantsppackage
hidden          style
label           time
report          tooltip
report_label    units
rows            value
```

In addition it is possible to override these attributes on the <results> element:

```
page_size
sort_column
```

And these attributes can be overridden on the <column> element:

```
display         tooltip
label           hide_label
style           show_tooltip
hidden          units
real_format
```

More information:

Search Definition Syntax (see page 166)

Derived System Searches

A derived search created from a System search has an additional feature; when the derived search has the same name as the system search, the derived search is used in preference to the underlying system search. This feature enables system searches to be easily customized.

For example, the default system search for Context Searches defines the number of events before and after the selected event to be 62 (giving a total of 125 events or 5 pages). If only one page of events is required a derived search based on the system search can be created with a value of 12 and saved with the same name (such as Context Search). Then when a Context Search is run only one page of results will be returned.

Chapter 17: Advanced Features

This section contains the following topics:

Pre-search Processing

For a complex search definition it may be useful to perform some processing after the user has pressed the Run button, but before submitting the search request. For example, to ensure that the parameters specified form a valid combination or to generate new parameters derived from those specified.

This can be achieved by defining a function called OnSubmit in the <script> element of <parameters>. OnSubmit takes no parameters and should return either true to submit the search or false if the search is to be abandoned.

Similarly, it may be required to generate derived parameters when the search is saved. This can be achieved by defining a function called OnSaveSearch in the <script> element of <parameters>. OnSaveSearch takes no parameters and should return either true to continue to save the search or false if the save is to be abandoned.

Note: If 'Save and Run' is used, both the OnSaveSearch and OnSubmit functions will be called.

Supporting Searches of Unlimited Size

By default the iConsole limits the number of results that can be returned in order to protect itself from being overwhelmed by large result sets and preclude the adverse effect on performance this would have for other iConsole users.

A controlled mechanism is provided to override this behavior but this requires particular support in the search definition and stored procedure, and appropriate privileges and configuration.

In use, the search will function as normal, but if the results have been constrained by the system limit (MaximumResultSetSize), it will be possible to click the Show More link to retrieve the total record count. In addition, an estimate of the execution time to retrieve all results will be displayed, giving the user the choice of continuing with the unlimited search, or canceling it.

Search Definition

A special parameter is required to control the operation of the iConsole and the stored procedure. This is done using the reserved parameter $CONTROL which should be defined as a constant choice with a value of 0, for example,

```
<parameter name="$CONTROL" type="choice" constant="true" argpos="6" value="0"/>
```

The iConsole will use this parameter to re-run the search to request the total record count or the unlimited results set.

Stored Procedure

The stored procedure must respond to the control parameter values as follows:

0

Perform a normal search, honoring a row limit if this is defined. The iConsole will limit the number of records based on the registry setting MaximumResultSetSize,

1

Return the total count of all results for the search. This should return a single, integer column containing the total result count. The name of the column can be anything you choose.

2

Perform an unlimited search, overriding any row limit specified. The iConsole will not apply a limit on the number of records but will retrieve all of the results. This may take some time and could overload iConsole resources.

Privileges

To run an unlimited search, the user must have the 'Allow searches of unlimited size' privilege.

Configuration

To enable the unlimited search capability, the registry setting AllowUnlimitedSearch must be given the value true.

Updating Results in the Cache

Sometimes the results from a search need to be refreshed because the underlying data has changed, for example, the audit status has been updated. This could be achieved by re-running the search, but this would be unnecessarily time consuming. Instead, there is a mechanism to run a mini search to retrieve just the updated information, and merge this into the results already in the results cache. This is enabled using the update="true" attribute on the <results> element.

The mini search stored procedure has the same name as the main search, but prefixed with 'Update_' for example, for the standard search, the main SP name is Standard_Search_V2_1; the update mini SP is called Update_Standard_Search_V2_1. The stored procedure takes a single parameter, which is a string containing a comma-separated list of values. These are key values obtained from the column identified by the ref attribute of the <results> element, for every selected row of the results. The ref attribute must refer to a column defined as a primary key i.e. the corresponding column must have the attribute primary_key="true".

The names of the columns in the result-set from the mini search must correspond to those in the original results (although not all columns need to be defined). The primary key column is used to identify records in the original results, and columns with matching names are replaced by values from the mini search results.

Customizing Time Display in Date Parameters

By default date and date_range parameter types display dates and times to the second (using local format). If this level of precision is not required (e.g. only hours and minutes are required) this can be specified using the time attribute to specify the required precision.

As an example, this parameter will display a date range with times specified as hours and minutes (no second component):

```
<parameter name="dates" type="date_range" argpos="1" label=" Dates:" time="minute"/>
```

Customizing the date_range Parameter

The collection of specified and pre-defined periods provided by default with the date_range parameter type are adequate for most purposes, but if required, the list of options can be customized.

To customize the date_range periods, simply specify option elements within the date_range parameter for each period required. The type attribute is used to select either the 'specified' or 'predefined' range. If no type is specified, the default is pre-defined.

- Specified periods are selected by number from the following list:

  1=Days, 2=Weeks, 3=Months

- Pre-defined periods are selected by number from this list:

  1=All Dates, 2=Today, 3=This Week, 4=Last Week, 5=This Month, 6=Last Month, 7=This Quarter, 8=Last Quarter, 9=This Year, 10=Last Year, 11=Custom

The default specified or pre-defined period label can be overridden by specifying the required text in the <option> element as shown in pre-defined options 1 and 6 below. Specified options 1 and 2, and pre-defined options 3, 4, 5 and 11, will use the default text.

In addition, user-defined periods can be specified using a negative option value, the value being the length of the period. For example, -42 gives a range of six weeks. All user-defined periods begin the specified number of days ago and end today; there is no facility to specify the end of the period.

Example user-defined periods

```
<parameter name="dates" type="date_range" argpos="1" label=" Dates:"
attrs="2;5;1;-21">
    <option type="specified" value="1"/>
    <option type="specified" value="2"/>
    <option type="specified" value="-14">Fortnights</option>
    <option type="predefined" value="1">Every date possible</option>
    <option value="-2">Two Days</option>
    <option value="3"/>
    <option value="4"/>
    <option value="-10">Ten Days</option>
    <option value="-21">Three Weeks</option>
    <option value="5"/>
    <option value="6">A month ago</option>
    <option value="-42">Six Weeks</option>
    <option value="11"/>
</parameter>
```

The example on the previous page will produce this list of specified periods:

- Days
- Weeks
- Fortnights

And this list of pre-defined periods:

- Every date possible--Three Weeks
- Two Days--This Month
- This Week--A month ago
- Last Week--Six Weeks
- Ten Days--Custom

Customizing Results for Display or Download

By default, you can download the complete set of results from an iConsole search in XML Spreadsheet format, suitable for loading into Microsoft Excel. This behavior can be customized in three ways:

- The download format for all searches can be changed by using the registry value SearchResultsDownloadFormat to specify the file type (such as csv).

  Note: The file extension used by the iConsole when downloading search results can be set using the SearchResultsDownloadExt registry value. If using the xls format, the default file extension is xml for compatibility with Microsoft Excel 2007. For Excel 2003, we recommend setting this value to xls.

- Additional download tools can be specified in the search definition using the <tool> element to invoke the function DownloadResults, specifying the required format. For example:

```
…
<tool name="xyz_download"
  tooltip="Download file in XYZ format"
  icon="download-xyz.gif"
  function="DownloadXYZ"/>
<script>
  <![CDATA[
    <script language="javascript">
    <!--
      function DownloadXYZ()
      {
        DownloadResults('xyz');
      }
    -->
    </script>
  ]]>
</script>
…
```

- Specify a download format to be displayed inline. In this case the format to be used is specified in the format attribute of the results element, and has the '$doc:' prefix. For example:

```
<results … format="$doc:myformat.pdf" >
```

Built-in support is provided for two formats;

- **XLS:** XML Spreadsheet format
- **CSV:** Comma-Separated Values format.

Note: Built-in download formats are case-sensitive.

Custom formats can be implemented using Extensible Stylesheet Language Transformations (XSLT). A stylesheet named 'format'.xsl must be written to perform the required transformation from the search results, and saved in the following folder on the Web Service (back-end) server:

```
WebService\transformations\report-formats
```

Writing such transformations is beyond the scope of this document.

Note: The built-in download formats are highly optimized in order to support large numbers of results without reaching system time-out limits. Formats making use of XSL transformations will perform more slowly and should only be used when the number of results is modest (c. 2000), depending on the complexity of the transformation.

Note: Inline document formats are processed as part of the search and are therefore not subject to time-out issues, except when downloading the final document.

More information:

Toolbar Button to Show Excel Spreadsheet (see page 78)

Post Processing

Using the mechanism described above, custom download files can be produced in a text format, but it is not possible to produce binary format files, such as PDF, using an XSL stylesheet alone.

In order to provide the ability to produce such files, it is possible to define a custom download that uses a post processing step to convert an intermediate, text file into the final binary file.

To request a post processing step the download format should be specified in two parts, separated by a full stop e.g. xslfo.pdf. The first part (before the full stop) is used to define the name of the stylesheet i.e. search-results-xslfo.xsl. The second part (after the full stop) is used to define the post processor. The post processor is specified in the registry, in the WebService hive with a key of the form SearchResultsDownloadConvertTo'post-processor'. For the example format above, the key would be SearchResultsDownloadConvertToPDF.

The value assigned to the post processor registry key should be the path to a command line application, accessible on the iConsole application server, which simply takes the names of an input file and an output file as parameters and performs the conversion from one to the other.

Note: If the particular post-processor requires additional parameters to function correctly, a batch file should be produced that incorporates these.

Note: If the conversion may take some time it is possible to increase the time that the iConsole will wait for the converter to respond. By default the timeout is set to 90s but this can be overridden using the registry value SearchResultsConverterTimeoutSecs.

PDF Reports

This section describes a configuration for producing PDF reports as output from an iConsole search (either for download as a file or to display inline).

The first requirement to produce a PDF as output is a stylesheet that will transform the search results into a document description using a formatting language such as XSL-FO (Extensible Stylesheet Language Formatting Objects), the second step is to post process the resulting XSL-FO to produce a PDF using a tool such as Apache FOP or one of the other commercially available products.

Apache FOP Installation

FOP is simple to download (http://xmlgraphics.apache.org/fop/download.html) and install but there is a problem with Java installations prior to version 1.6 (see below). With JRE (Java Runtime Environment) 1.6 or later, FOP should work without any modification to the installation.

FOP is a java application which relies on a recent version of certain libraries (see below). Unfortunately the libraries included with the Sun JRE before version 1.6 are too old and need to be overridden. With JRE 1.4 it's not sufficient to just specify the new libraries in the classpath, they have to be copied into the …/jre/lib/endorsed folder, (the endorsed folder may need to be created). This is an official way to override libraries without overwriting them and is known as the Endorsed Standards Override Mechanism.

According to the FOP 0.94 FAQ concerning this issue, only the Xalan library is affected. Nonetheless, copy all of the following libraries into the endorsed folder from the FOP installation:

- xalan-2.7.0.jar
- serializer-2.7.0.jar
- xml-apis-1.3.02.jar

Note: If there are multiple Java installations then it is the jre that will be referenced by the JAVA_HOME environment variable that needs to be modified in this way, as this is the one that will be used (by default) to run FOP.

Note: If it is necessary to have multiple Java installations and it is not possible to set JAVA_HOME to reference the version to be used by FOP, it may be necessary to modify the fop.bat file to set the path to the correct java installation.

Note: Care should be taken when installing FOP from a remote source onto a Virtual PC installation running Windows XP SP2 or later. The installed files may be marked as 'Blocked' by the operating system as a security measure and will cause the PDF generation to fail. If FOP is run from the command line this error is displayed: "This file came from another computer and might be blocked to help protect this computer.". To avoid this problem copy the FOP zip file to the Virtual PC, show the file properties and click 'Unblock' before unzipping.

Drilldown Links

A PDF report produced by the iConsole can offer drilldown capability by including links in the report. These links will then run a specific search in the iConsole to retrieve the live drilldown data. The links will function both when viewing the report within the iConsole using the Acrobat Reader browser plug-in and also when viewing the PDF report using the standalone Acrobat reader, as long as the iConsole is accessible from the computer on which the PDF is being viewed. This means that even when a PDF report is generated automatically using WgnReport and emailed to a user, that user may still access the drilldown links.

A helper template has been produced in order to simplify the creation of the url for use in the XSL-FO basic-link element. The template is called ExternalSearchURL and takes the parameters name, major['1'], minor['0'], section['Published'], category[''] and parameters[''] (default values in square brackets). Only the name parameter is mandatory to define the name of the search to be run. The parameters parameter specifies the parameters to be used for the search as an ampersand separated list in the format name1=value1&name2=value2. If the parameter name is enclosed in square brackets the value is interpreted as the attrs for the parameter. This is particularly useful for date range parameters, allowing pre-defined date ranges to be specified.

The following example creates a url to run version 2 of the standard search to get emails for the last calendar quarter:

```
<xsl:variable name="search-url">
        <xsl:call-template name="ExternalSearchURL">
                <xsl:with-param name="name" select="'Standard Search'"/>
                <xsl:with-param name="major" select="'2'"/>
                <xsl:with-param name="parameters"
                        select="'chkEE=true&amp;[dateRange]=2;1;2;8'"/>
        </xsl:call-template>
</xsl:variable>
```

This may then be used in the basic-link element like this:

```
<fo:basic-link show-destination="new" external-destination="{$search-url}">
        <xsl:value-of select="'My Link'"/>
</fo:basic-link>
```

By default, the drilldown will open in a new window. To display the drilldown in the iConsole window specify the pseudo parameter NewWindow=false. This will prompt the user with the option to replace the PDF report; otherwise the drilldown will open in a new window.

More information:

Date Range (see page 40)

Graphics and Charts

While HTML and XSL-FO have no built in graphics capability, it is possible to include SVG (Scalable Vector Graphics) XML in the formatted results. To include SVG in HTML, a browser extension is required. The most common one for Internet Explorer is the Adobe SVG Plugin. This can be declared in the HTML (usually in the <head> section) like this:

```
<object id="AdobeSVG" classid="clsid:78156a80-c6a1-4bbf-8e6a-3cd390eeb4e2" />
<xsl:processing-instruction name="import">namespace="svg"
implementation="#AdobeSVG"
</xsl:processing-instruction>
```

To include SVG within XSL-FO it should be enclosed within an fo:instream-foreign-object element. Clearly, to produce graphics in the final PDF, the post processor must support the conversion of SVG embedded in the XSL-FO. Apache FOP includes Apache Batik to provide this capability.

Charts

SVG only defines graphics primitives; to simplify the production of business charts, a template library (svg-chart.xsl) has been produced, providing basic pie, bar, column and line charts. To use the charting templates, the library can be incorporated into the custom format stylesheet by defining the namespace and using the xsl:import element to include the templates and the xsl function library as follows:

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:svg-chart="http://www.ca.com/svg-chart">

    <xsl:import href="../xsl-functions.xsl"/>
    <xsl:import href="../svg-chart.xsl"/>
…
```

Functions from the xsl function library can be accessed in the report by declaring the required namespaces. The most useful library is the string library (http://www.orchestria.com/string) which contains functions encodeName(*name*) and decodeName(*name*) which can be used to convert XML names that have been encoded with Unicode hex strings (e.g. an XML name cannot contain a space so 'My Name' is encoded as 'My_x0020_Name').

Producing a chart is a two-step process:

1. Define the data to be charted.
2. Draw the chart using the prepared data.

Preparing Chart Data

Three templates are available to help with the preparation of the chart data:

svg-chart:data-totals

Prepares chart data from the totals of the specified column names

svg-chart:data-columns

Prepares chart data from the specified column names

svg-chart:data-add-series

Prepares chart data from a list of values

Note: Chart data must be stored in an xsl:variable so it can be passed to the svg-chart:draw template.

Note: A list of names or values must be separated with spaces, not commas.

Example 1

In this example, chart data is prepared for the search results columns ColA, ColB, ColC, column labels for these columns from the search definition are used to identify the values which are the totals for each column. One data series is produced with the name 'totals'.

```
<xsl:variable name="totals-data">
    <xsl:call-template name="svg-chart:data-totals">
        <xsl:with-param name="columns" select="'ColA ColB ColC'"/>
    </xsl:call-template>
</xsl:variable>
```

Example 2

In this example, chart data is prepared for the search results columns ColB, ColC, ColD. The values in ColA are used to identify the values. Three data series are produced, each with the name of the column label from the search definition:

```
<xsl:variable name="column-data">
    <xsl:call-template name="svg-chart:data-columns">
        <xsl:with-param name="labels" select="'ColA'"/>
        <xsl:with-param name="columns" select="'ColB ColC ColD'"/>
    </xsl:call-template>
</xsl:variable>
```

Example 3

In this example, chart data is specified directly as a list of values. The specified labels and values are used. Two data series are produced with the names 'Series 1' and 'Series 2':

```
<xsl:variable name="list-data">
    <xsl:call-template name="svg-chart:data-add-series">
        <xsl:with-param name="name" select="'Series 1'"/>
        <xsl:with-param name="labels" select="'A B C'"/>
        <xsl:with-param name="values" select="'1 2 3'"/>
    </xsl:call-template>
    <xsl:call-template name="svg-chart:data-add-series">
        <xsl:with-param name="name" select="'Series 2'"/>
        <xsl:with-param name="labels" select="'X Y Z'"/>
        <xsl:with-param name="values" select="'7 8 9'"/>
    </xsl:call-template>
</xsl:variable>
```

The resulting chart data will be structured like this (although this is transparent):

```
<series name="Series 1">
    <point label="A" value="1">
    <point label="B" value="2">
    <point label="C" value="3">
</series>
<series name="Series 2">
    <point label="X" value="7">
    <point label="Y" value="8">
    <point label="Z" value="9">
</series>
```

Drawing the Chart

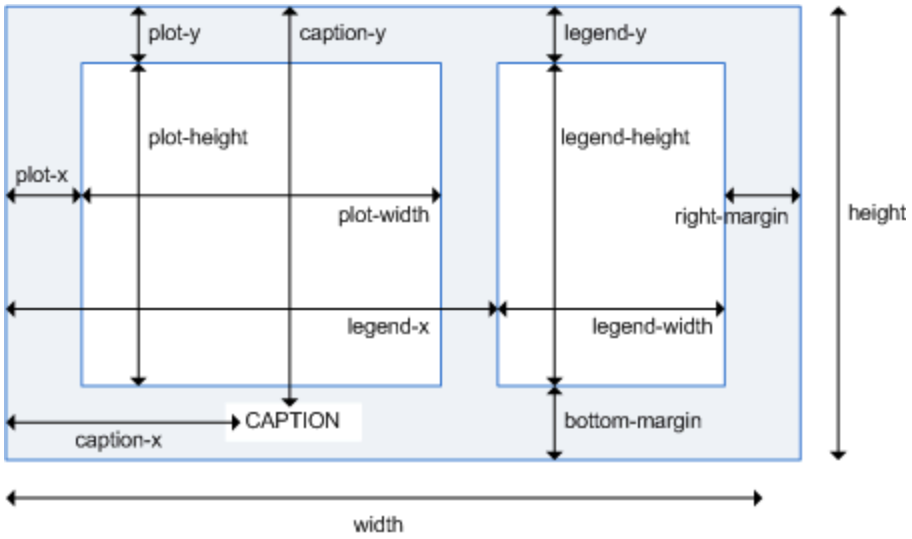
Once data has been prepared, a chart can be drawn by simply calling the svg-chart:draw template and passing the prepared data:

```
<xsl:call-template name="svg-chart:draw">
    <xsl:with-param name="data" select="$totals-data"/>
</xsl:call-template>
```

The svg-chart:draw template has many parameters allowing the layout of the chart to be customized but these are all defaulted, producing a pie chart with a reasonable layout.

Customizing Chart Layout

The default chart layout is like this, showing the parameters that control the layout:



The width and height values are interpreted as millimeters and define the size of the chart in the final document; all of the other values are in arbitrary plot coordinates. If no layout parameters are specified a chart of width 100mm is produced with no legend so the plot area fills the available space. The height of the chart is usually determined automatically from the other setting.

The default value for each of the layout parameters is shown in the table of parameters in the reference section on page .

More information:

Chart Template Parameters (see page 152)

Running a Sub-Query

When more detailed information is required, based on the results of a search (e.g. drilldown), it is possible to run a Sub-Query to return the detailed results. The sub-query can be run synchronously (for very quick searches) or asynchronously and the results can either replace the current page, or be presented in a popup window.

The sub-query is run using the RunQuery helper function which has the following calling sequence:

RunQuery Calling Sequence

The following is the syntax for RunQuery:

```
RunQuery('name','major','parameters','sync','popup','minor','type','category','target');
```

where:

name

Specifies the name of the sub-query search.

major

Optional. Specifies the major version number of the sub-query search (default is 1).

parameters

Optional. Specifies the parameters for the sub-query either as an XML fragment (see below) or as a javascript array of parameters.

sync

Optional. When true, specifies that the sub-query should be run synchronously (default is false).

popup

Optional. When true, specifies that the sub-query should display its results in a popup window (default is false).

minor

Optional. Specifies the minor version number of the sub-query (default is latest).

type

Optional. Specifies the sub-query type (default is 'published').

category

Optional. Specifies the category (sub-folder) of the sub-query (default is '').

target

Optional. Specifies the target page for the results of the sub-query (default is 'search-results').

The order of parameters is designed to define those that are most often used first, allowing later parameters to be omitted. As a minimum, only the name of the sub-query needs to be defined:

```
RunQuery('My Sub-Query');
```

Parameters

Parameters for the sub-query can be specified as an XML fragment containing parameter specifications which will override those in the search definition for the sub-query. For example, to specify parameters to define a title parameter as 'My Title' and num_results parameter as 50 would need the following string:

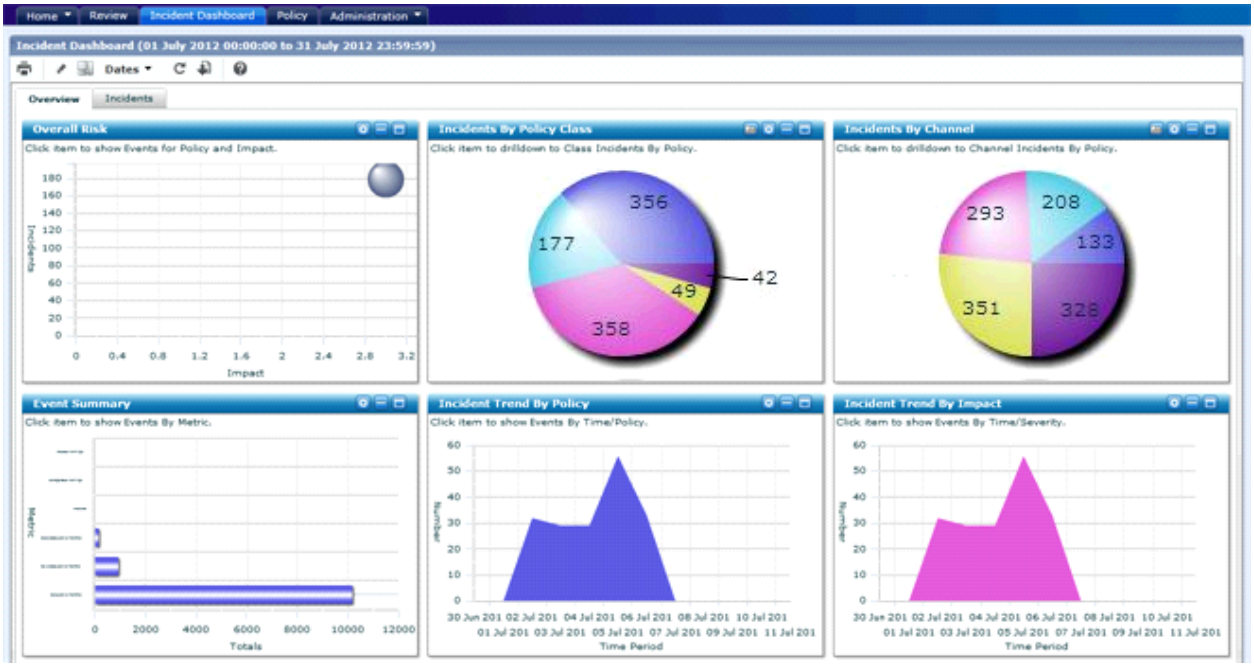
```
'<parameters><parameter name="title" value="My Title"/><parameter
name="num_results" value="50"/></parameters>'
```

Alternatively, parameters can be specified as a javascript array:

```
[{name:"title",value:"My Title"},{name:"num_results",value:"50"}]
```

Dashboards

A dashboard comprises a collection of searches, each of which populates a 'pane' in the dashboard display. This collection of 'panes' can be grouped into separate pages or onto a sidebar that is visible for all pages. The dashboard, pages and panes are all created as derived searches with baselines of Dashboard Master, Page and Pane respectively.



Individual panes on the dashboard can be minimized or maximized depending on requirements and the layout for a particular dashboard will be remembered the next time that dashboard is displayed.

Where a pane displays a chart, the type of chart (and other options) can be defined in the derived pane search and the user can change them by clicking the 'Options' button on the pane title bar.

Pane Attributes

Additional pane attributes can be specified using the Pane Attributes parameter. Attributes consist of a name:value pair, with multiple attributes separated with a semicolon. The only currently supported attribute is ValueAxisType which defines how the chart value axis (the vertical axis on a column chart) will be formatted; this can take the values Normal, Log (logarithmic) and Date. To set a logarithmic scale simply set the value of Pane Attributes to value="ValueAxisType:Log". This will then use a log scale for the value axis on Column, Bar, Line, Bubble and Area charts. Setting the Date type will format dates in the current locale and will expect standard CA DataMinder date values (such as milliseconds since 1 Jan 1970).

Drilldown

The chart can also specify a drilldown search (using the drilldown parameter) to present more detailed information for the selected item (such as a pie wedge) when it is clicked on. When present, the drilldown is detailed in the text above the chart and the cursor changes to a hand pointer over the items where drilldown is available. The drilldown can take one of three forms specified by the lstDrilldown parameter:

Drilldown in place

The detailed results replace the existing chart and a 'Back To' button is added to the title bar. To return to a previous chart, select it from the 'Back To' list.

External drilldown

Clicking the item results in a search being run with the results displayed on the iConsole search results page, replacing the entire dashboard.

External popup

Similar to 2) but the results are displayed in a popup window, leaving the dashboard displayed in the main window.

A drilldown search is defined in a similar way to a normal pane search but external drilldown searches will probably need to specify column information in order to display the results appropriately. Drilldown can be to an arbitrary depth.

Dashboard charts are generally populated from pre-aggregated data, showing a snapshot at the time of aggregation. Each chart can also define a 'live data' search using the liveSearch parameter; this is selected using the Open button on the chart title bar (if available). This opens a popup window to display the up-to-date live results corresponding to the chart.

By default, dashboard pane searches are run unrestricted so that RLS users can access aggregated data that they would normally not have access to. If drilldowns need to be run restricted e.g. if they are based on the Standard Search, the drilldown search definition should include the restricted="true" attribute in the search element.

Dashboard Search Architecture

A dashboard search consists of a hierarchy of search definitions derived from a set of master search definitions. The hierarchy comprises:

- The overall dashboard search definition which defines the name, date range, report format (optional), sidebar (optional) and one or more pages.
- The sidebar and page definitions which list the panes that make up the page.
- The individual pane definitions that define the search to be executed, display type and drilldown capabilities. Drilldown searches and live data searches are implemented as additional panes.

A dashboard can be created in two ways:

- Manually writing the derived searches that make up the dashboard.

  This option requires an understanding of the underlying search definitions but provides most flexibility in creating and distributing dashboard definitions.

- Deriving a search from an existing dashboard in the Dashboard folder.

  This option is easier to use and permits the incorporation of pages or panes that are already defined into a new dashboard arrangement. When deriving a new dashboard, new page and pane definitions can be created through the iConsole Administer Searches | Searches page.

Dashboard Search SQL

Dashboard searches (and drilldown and live data searches) are all run indirectly via the 'Pane' search which is installed as part of the core support for dashboards. The search to populate a pane must therefore conform to a standard calling sequence with parameters as follows:

datFrom

The start date for the search

datTo

The end date for the search

nTop

The number of results to be obtained (for example, top 10)

sCategory

Category information for the search. This is automatically populated for a drilldown search with the row identifier (column 1 value) from the selected item, allowing the drilldown search to obtain relevant results.

sSeries

Series information for the search. This is automatically populated for a drilldown search with the column identifier (column name) from the selected item, allowing the drilldown search to obtain relevant results.

idUser

The id of the user running the search

bDebug

A debug flag allowing debug information to be written when set to 1.

Column names are not predefined by the 'Pane' search as there can be a variable number of columns. The column names are taken directly from the search results when displaying in the dashboard. (External drilldown searches may use all of the search definition facilities for defining columns and results attributes.) The first column will be used to define the Category (x-axis) and subsequent columns are used to define data series.

Bubble Charts

The Bubble chart is an exception to this rule in that columns have specific meanings as follows:

Column 1

Bubble label

Column 2

Bubble X Axis value

Column 3

Bubble Y Axis value

Column 4

Bubble size. Bubbles are automatically resized to fit the display but maintain a relative size difference.

Column 5

Optional: Bubble color. The value is used to index into the standard color series.

Data Grid Styles

Additional style control is available for panes displayed as a data grid. Style names can be defined for individual cells or complete columns so that they may be highlighted using e.g. background color. Styles are defined by appending the style name to the column name or cell value using '::' as a separator. For example:

```
SELECT 'Credit Information' AS "Policy Name", '4::SeverityLow' AS "Risk", '5' AS
"High::SeverityHigh"
```

Will use the 'SeverityLow ' style for the cell containing the Credit Information 'Risk' and the 'SeverityHigh' style for the entire 'High' column. At present only four styles are supported: SeverityNotSet, SeverityLow, SeverityMed and SeverityHigh. Since the styles are currently compiled into the Flash object adding new styles is not a simple customization task.

Default Dashboard (Deprecated)

It is possible to specify a particular dashboard to be the default and it will then appear as an additional tab. This is done by setting the DefaultDashboard value in the following registry key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\ComputerAssociates\CA DataMinder
    \CurrentVersion\WebService
```

DefaultDashboard is a string value with the format:

```
'DashboardName,MajorVersion,MinorVersion,Section,Category'
```

Its default values are:

```
MajorVersion: 1
MinorVersion: 0
Section: Published
Category: Dashboard
```

This allows the default dashboard to be simply specified as the dashboard name (such as Demo Dashboard) when the default values are appropriate.

More information:

Common Searches (see page 114)

Common Searches

Sometimes particular searches or reports are used much more frequently than others. To provide quicker access to these searches it is possible to add them as additional tabs on the main iConsole page. This is achieved by specifying task_page="true" as an attribute of the search element in the search definition.

Note: For compatibility with older releases of the iConsole that do not support the task_page attribute, it is also possible to achieve the same result by adding a prefix of '*' to the legal attribute.

Chapter 18: Localization

Search definitions can be enhanced to support local languages. This is achieved through translation dictionary files which contain translations for all of the terms used within the search definition. Search localization files are found under the Webservice/lang folder.

This section contains the following topics:

Search Definitions and Search Results

Since search definitions and results are already in XML format with name attributes on the elements that require translation, the translation can be performed automatically by supplying appropriate dictionaries (see below).

Dictionary Format

The format of the dictionary xml files is illustrated in the following example. An XLIFF file consists of one or more 'file' elements. Each 'file' element contains a 'body' element, and can contain a 'header' element with project data. The 'body' element contains 'trans-unit' elements. A 'trans-unit' element contains 'source' and 'target' elements, and can contain 'alt-trans' elements.

Within these elements are the terms to be translated and correspond to attributes of the search definition parameters. This generally only involves label, description and tooltip attributes, and option elements for list parameters. Since element names cannot contain spaces or other special characters these need to be expanded into a hexadecimal format, as illustrated by the element for the 'Standard Search'.

```
<?xml version="1.0" encoding="utf-8"?>
<xliff version="1.2" xsi:schemaLocation="urn:oasis:names:tc:xliff:document:1.2
xliff-core-1.2- strict.xsd" xmlns="urn:oasis:names:tc:xliff:document:1.2"
xmlns:xsi="http://www.w3c.org/2001/XMLSchema-instance">
  <file original="default.xml" source-language="en" target-language="fr"
datatype="xml" xmlns="">
    <body>
      <group resname="Standard_x0020_Search">
        <trans-unit id="ID0AYRAI04RAI" resname="label">
          <source>Standard Search</source>
          <target>Recherche Standard</target>
        </trans-unit>
        <trans-unit id="ID0AYRAI05RAI" resname="description">
          <source>iConsole Standard Search for general purpose searching for all
event types.</source>
          <target>La Recherche de Norme d'iConsole pour chercher tous les types
d'événement.</target>
        </trans-unit>
        .....
      </group>
    </body>
  </file>
</xliff>
```

To simplify the production of dictionary files, there is an XSL transform available which will take a search definition xml file as input and produce a dictionary file as output. This transform is called SD-to-i18n-dict.xsl and can be found in the Webservice/transformations folder.

Dictionary Precedence

The iConsole looks for several dictionary files to perform the translation and applies these in order of precedence:

- Search specific strings based on the dictionary attribute (e.g. dictionary="mysearch") of the searches element, for the local dialect (such as, fr-BE for Belgian French) would be found in file lang/fr-BE/mysearch.xlf
- Search specific strings for the local primary language (e.g. fr for French) would be found in lang/fr/mysearch.xlf
- General strings for the local dialect in file lang/fr-BE/~global.xlf
- General strings for the local primary language in file lang/fr/~global.xlf

In this way the number of translation strings is minimized, with common terms in the general string files, generic French terms for the specific search in one file, and only Belgian variations in the Belgian French file.

Note: Not all of these files need to exist, the iConsole will use whichever dictionaries it finds, in the order above.

Note: The tilde (~) prefix in the global dictionary files ensures that these files are accessed after all other dictionary files.

Custom Reports

Custom reports use an XSL stylesheet to transform the results into the desired format. In order for the text embedded in these reports to be translated, the strings need to be defined in a structured way. This is achieved using an XML data island such as the example here, inserted into the root element of the stylesheet:

```
<ca:report name="MyReport" dictionary="MyDict">
  <ca:string name="heading">English Heading</ca:string>
  <ca:string name="string1">English 1</ca:string>
  <ca:string name="string2">English 2</ca:string>
  <ca:string name="string3">English 3</ca:string>
</ca:report>
```

The ca namespace needs to be declared using:

```
xmlns:ca="http://www.ca.com"
```

The strings are accessed using:

```
 <xsl:value-of select="//ca:string[@name='heading']" />
```

The namespace is required as the xslt schema does not allow elements with null namespace at the root level.

Dictionary entries would then be of the form:

```
<MyReport>
  <heading>French Heading</heading>
  <string1>French 1</string1>
  <string2>French 2</string2>
  <string3>French 3</string3>
</MyReport>
```

Chapter 19: Performance Tips

To make your searches as efficient as possible, you must understand how results are retrieved by the iConsole. When a search is requested, the sequence of events is:

1. The database builds a set of results to satisfy the query.
2. The iConsole infrastructure retrieves results one row at a time into a cache.
3. The iConsole service limits the maximum number of rows that can be stored in the cache to prevent resources being exhausted (defaults to 1000).
4. The browser requests a page of results (25) from the iConsole service's cache.

This section contains the following topics:

Tips for SPs and Search Definitions (see page 119)

Tips for SPs and Search Definitions

Consider the following when using SPs and search definitions:

- Avoid using DISTINCT and ORDER BY, particularly in queries that may generate a very large result set. These require the full results set to be built before any results can be returned and may never return any results before time-out limits are reached.
- For Oracle databases, use the /*+FIRST_ROWS*/ optimizer hint to return results as soon as they are available, but not in conjunction with DISTINCT or ORDER BY! This can significantly reduce response times.
- Limit the size of the result set using SELECT TOP (for SQL Server) or WHERE rownum < 'n' (for Oracle) in the SQL to avoid runaway queries.

Chapter 20: Changes to Javascript for Custom Reports and Searches

The iConsole has a new look and feel. The result is changes to the objects and their properties that appear on the HTML customization and results pages for searches and reports.

This chapter describes the main changes and what to do to make your Javascript compatible with the new iConsole version. If you require further assistance, contact CA Technical Support. A complete list of locations, primary service hours and telephone numbers, can be found at http://support.ca.com.

There are also changes to iConsole 12.5 XML capabilities, which are defined elsewhere in this iConsole Search Definition Guide.

From the point of view of Javascript, which may be defined in the reports XML file to be used on these customize and results pages, the changes to the objects and their properties to be referenced mean that corresponding changes to the Javascript logic are required to your customized reports. We have already applied the changes to the standard reports and searches .

This section contains the following topics:

The requires Attribute

The search (and report) XML definition file may contain an optional attribute requires, which is described in the Reference Information chapter. For new reports, we recommend the value requires="12.5". Use of this value enables the new iConsole features.

Using requires="12.5" causes a change in the way the iConsole renders the customization page. The controls on each tab are created dynamically when the tab is clicked. Thus the Javascript that references controls must handle the situation whereby the controls on as yet unrendered tabs will not have been created.

Javascript Validation

The first step is to ensure that the iConsole in-built Javascript validation is applied. This validation cannot guarantee the complete compatibility of your Javascript with the Release 12.5 iConsole, but it is essential to pass this stage before proceeding further.

The validation is switched on by setting the following registry Key Value to "true" on the client computer used to install the reports XML files. For example:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\ComputerAssociates\CA DataMinder
    \CurrentVersion\WebService]
"ValidateSearchJavascript"="true"
```

A basic validation process is applied by the iConsole when installing any reports XML file. To ensure this validation is applied to your customized reports, you must reinstall the reports XML.

If any errors or warnings are reported while installing an XML file, review and correct these before proceeding. For example, you must explicitly declare undeclared variables. Although these may seem trivial, they can cause errors later if a variable has the same name as an HTML element.

The following shows an example of an undeclared variable.

```
function setRollupOrder()
{
sRollupOrder = document.getElementById('sRollupOrder')
sRollupOrder.value = str;
}
```

This is the iConsole validation warning message:

```
The search definition javascript has errors.
    JS1135:Warning-Variable 'sRollupOrder' has not been declared Line 36, position
8.
   sRollupOrder = document.getElementById('sRollupOrder')
```

In the function above, and depending on the browser used, the variable sRollupOrder may be implicitly defined as a global variable, which then clashes with the same name used for an HTML element. The function must define the variable explicitly to avoid this:

```
function setRollupOrder()
{
       var sRollupOrder = document.getElementById('sRollupOrder')
       sRollupOrder.value = str;
}
```

Javascript Global Statements

The use of Javascript global statements is not supported, except for the definition of global variables. Because of the asynchronous loading process used in the iConsole for some elements, global statements might be executed before the page loading is complete in the browser and will not be executed on returning to the page. This could lead to errors or unwanted behavior.

To illustrate this, the references to "RemoveAnyFromAuditList" are calls to a user-defined function (defined later in the report Javascript but not shown here):

```
<script language="javascript">
<!--
RemoveAnyFromAuditList(1);
RemoveAnyFromAuditList(2);

function LocalReset()
{
        // LocalReset is called automatically when the page is loaded (or reloaded)
        SetInterventionchk();
       }
-->
</script>
```

If these functions are required to be executed on page load and reload, they must be moved to the standard function LocalReset as follows:

```
<script language="javascript">
<!--
function LocalReset()
{
        // LocalReset is called automatically when the page is loaded (or reloaded)
             RemoveAnyFromAuditList(1);
             RemoveAnyFromAuditList(2);
             SetInterventionchk();
       }
-->
</script>
```

Custom iConsole Functions

In the past, some reports have defined their own versions of standard iConsole functions. While supported, we strongly recommended that you adopt the use of standard functions as documented in this guide. For example, running a drilldown query is now only supported by the standard function RunQuery.

If the report used a custom version of RunQuery called rut_RunQuery and set the event UID for the drilldown in an HTML Element called 'event_uid' then this is supported automatically. However, we recommend that you update the report to use RunQuery directly.

The logic used in a report's customized version of this function might have been appropriate with earlier versions of the iConsole, but the changes in iConsole mean that may no longer be the case. Even though the Javascript may function as expected, we recommend that you change the Javascript to call the standard functions.

The iConsole changes cause many objects on the customization and results pages to be rendered differently and to have different properties. This means that custom code written to manipulate these might not work. The standard iConsole functions referred to above, to be used instead of the old custom code, can be made available by including the supplied search-helper.js file.

This is achieved by adding the following to your Javascript:

```
<script type="text/javascript" src="scripts/search-helper.js"></script>
```

Some examples (before and after):

Before:

```
document.getElementById('numIntervention').value = numInter;
```

After:

```
SH_SetTextBoxByName('numIntervention', numInter);
```

Before:

```
document.getElementById('chkBL').checked = true;
```

After:

```
SH_SetCheckboxByName('chkBL', true);
```

Before:

```
var     obj = document.getElementById('lstBRE');
        if (obj) obj.disabled = bDisabled;
```

After:

```
SH_DisableParameterByName('lstBRE', bDisabled);
```

Microsoft Office Web Components

Some reports have used Microsoft Office Web Components (OWC) on the results pages to enable the download of the results into a customized Excel file. Although the use of OWC is supported in the iConsole, we no longer recommend it. We provide an Excel download tool on the toolbar. You can customize the download format using a custom xsl transform.

However, if a current report displays an Excel view by using a tool named 'excel' and a function name of 'ShowExcel', this is supported automatically. Note that this uses the standard in-built ShowExcel function.

If you have customized the report's ShowExcel function, these customizations will no longer be present. To reinstate these customizations, we recommend that you use the download mechanism described Customizing Results for Display or Download.

More information:

Customizing Results for Display or Download (see page 98)

Chapter 21: Reference Information

This section contains the following topics:

<column> Element

The <column> element in an XML search definition defines a column in the Search Results screen, where the column values represent a field in the event record.

The supported entities are as follows:

name

(Optional) Specifies a name that uniquely identifies the column in the Search Results screen.

width

(Optional) Specifies the preferred column width in the Search Results screen.

Example:

```
width="20%" or width="120px"
```

label

(Optional) Specifies the column heading in the Search Results screen.

Example:

```
label="Audit Status"
```

hide_label

(Optional) Specifies that the column heading label should not be shown, typically used when the column width is very narrow.

Example:

```
hide_label="true"
```

link_function

(Optional) Specifies the predefined function that is called when a reviewer clicks an individual value in the results column. A range of standard scripting functions are supported—see Standard Scripting Functions (see page 167).

Example: This attribute is used by the Standard Search in the iConsole to show details about an email sender:

```
link_function="_UserDetails"
```

Note: Can be used in association with the ref attribute.

ref

(Optional) Specifies the name of the column to be used as the unique reference for a 'column action'. The results element must include a column with this name (column names are specified using the name attribute).

If the ref attribute is omitted, a normal hyperlink is created.

Example: This attribute is used by the Standard Search in the iConsole to show details about an (Optional) email sender:

```
ref="user_uid"
```

Note: Always used with the link_function attribute.

type

Specifies the type of column. This must be compatible with the associated database types; see Search Result Column Type Mappings (see page 166).

The supported column types are:

id

An identifier field containing values for the ref attribute.

text

Generic text field.

numeric

Generic numeric field.

date

Generic date field.

timestamp

Generic timestamp field.

icon

A two part field comprising the filename of the icon and a tooltip separated by a colon.

list

A list of values separated by semicolons.

link

A field which has a hyperlink to retrieve additional information such as user or event details. Link columns use the link_function attribute or display a popup.

Alternatively, if the ref attribute is not defined, the field is used as a URL to create a hyperlink to a page which is opened in a separate window. If required, the link can be displayed with alternative text (i.e. not the URL) by returning Label;URL as the column value.

quantity

A numeric field with unit information appended. For example, 123 MB, 30 days.

boolean

Generic Boolean field (that is, column values can be True or False).

real

A real (decimal) number, used in conjunction with the real-format attribute.

On Oracle 10 or later, the value returned must be a fixed-point value such as NUMBER(38,6).

hidden

(Optional) Specifies whether the column is hidden by default.

Example:

```
hidden="true"
```

Note: Typically used for id columns.

units

(Optional) Specifies the units to be appended to column values, typically when used with a quantity column.

Example:

```
units="%"
```

show_tooltip

(Optional) Specifies that the full content of a cell is shown in a tooltip when the mouse pointer hovers over the cell. This is useful for narrow columns that contain very long values.

Example:

```
show_tooltip="true"
```

primary_key

(Optional) Specifies that the column is a unique key. For the current release, this attribute is used to delete events from the Search Results screen (using the ToolbarDelete function (see page 167)) and for updating results (using the ToolbarUpdate function).

icon_class

(Optional) Specifies a subfolder containing the icon images for an 'icon' column type. This must be subfolder of the \images folder in the CA DataMinder installation folder on the iConsole front-end Web server.

tooltip

(Optional) Allows a tooltip to be defined for the column heading.

Example:

```
tooltip="My column tooltip"
```

real_format

(Optional) Specifies the format to be used for a real number using XML stylesheet formatting syntax.

Example: '#.00' will format a number to 2 decimal places, 1234.5678 will be formatted as1234.57.

```
real_format="#.00"
```

display

(Optional) When this attribute is included, the column is optional. It specifies whether the column should be included in the displayed results. When set to "false" the column is not displayed in the search results unless selected from the Column Selector of the search results or search properties page.

Example:

```
display="false"
```

style

(Optional) Specifies a style for this column. This will be applied to both the headings and cell contents and is used typically to specify text formatting.

Example:

```
style="text-align: center"
```

total_function

(Optional) Specifies the aggregate function to be used to calculate the total for this column. See Displaying Aggregate Totals (see page 87).

col_style_ref

Specifies a column in the results that will define the name of a CSS class in the stylesheet that will be used to define the style for each cell in this column.

The stored procedure is responsible for defining the class for each row; a NULL value will apply no class.

The precedence of certain styles (e.g. font-size) may need to be forced by using the '!important' rule.

```
.bigfont {font-size: x-large !important;}
```

<parameter> Element

The <parameter> element in an XML search definition defines defines an individual parameter for a search. Specifically, it defines an input field that can be used directly or indirectly as a search parameter.

The supported entities are as follows:

<option>

Specifies a required item for parameters of type="list" or type="choice". List and choice parameters generally need at least two options.

Example:

```
<option value="4">Four</option>
```

name

Specifies a unique name for the parameter. You can use this attribute to identify the parameter in a script. It does not need to match the argument name in the SP.

Example:

```
name="idUser"
```

Note: This attribute is also used to configure context searches. See Context Searches <parameter> (see page 156).

value

(Optional) Specifies the default value for the parameter in the Search screen.

Example:

```
value="my value"
```

See also Context Searches <parameter> (see page 160).

type

Specifies the type of search parameter. The type must be compatible with the SP. See Parameter Type Mappings (see page 163).

Example:

```
type="text"
```

The following types are supported:

label

A text label, for enhancing the screen display. It provides no input to the search.

icon

A graphic, for enhancing the screen display. It provides no input to the search.

text

A text input box.

numeric

A numeric value.

date

A date value. Dates may be specified as absolute or relative dates. See Single Date (see page 39).

date_range

A pair of date values indicating a time period. This is an aggregate parameter that includes several fields to simplify the selection of common time periods. Defaults for the date range may be specified using special attributes as absolute or relative dates. See Date Range (see page 40).

checkbox

A check box.

list

A list of items from which the user may select one or more. You can enable multiple selections using the multiple attribute and specify dynamic retrieval using the spname attribute. The appearance of the list can be modified using the attrs attribute.

choice

Radio buttons.

lookup

A text box with dynamic lookup, for example, to retrieve a list of users matching a name fragment and to allow selection from a popup.

lookupbyid

A text box similar to lookup, but uses an ID for the search rather than a string. As an example, this is used for the user lookup where instead of using the user name in the stored procedure, the user's id is used which is more precise and more efficient.

real

A real (decimal) number.

systemid

A special parameter type that provides the reviewer's user ID or machine ID. These parameters are used internally and do not appear in iConsole search screens.

attrs

(Optional) Specifies additional, type-specific attributes which control the behavior of the parameter. The primary use for this attribute is in conjunction with the list parameter type. See also Multi-Select List Parameters (see page 38), the date parameter type (see page 39), and the date range parameter type (see page 40).

argpos

(Optional) Specifies the order in which the parameter is listed in the arguments passed to the SP (see the CREATE line in the examples). This allows you to change the parameter layout in a search screen independently of the order in which they are passed to the SP.

Note: If no argpos attribute is specified, the parameter is passed to the SP but may be used in the generation of an actual parameter.

label

(Optional) Specifies the text that is shown next to the search parameter in the Customize Search screen.

Example:

```
label="Audit Status"
```

units

(Optional) Specifies a text label appended to a numeric parameter in the search screen.

Example:

```
units="%"
```

onchange

(Optional) Defines the JavaScript code to be executed if the parameter value is changed. Use this attribute to give dynamic behavior to the search properties page, for example, to alter another parameter based on the setting of this one.

Example:

```
onchange="UpdateParameters();"
```

onclick

(Optional) Defines the JavaScript code to be executed if the parameter is clicked. Use this attribute to give dynamic behavior to the Customize Search page, for example, to disable a particular feature of the search if a parameter is selected.

Example:

```
onclick="DisableFeature();"
```

onkeypress

(Optional) Defines the JavaScript code to be executed if a key is pressed. Use this attribute to give dynamic behavior to the Search Properties page, for example, to apply input validation rules.

Example:

```
onkeypress="ValidateInteger();"
```

constant

(Optional) A Boolean attribute for specifying that the search parameter cannot be seen or modified by the user. This can be useful for making parameters inaccessible in a simple search variant.

Example:

```
constant="true"
```

sppackage

(Optional) Specifies the package name for an SP.

Example:

```
sppackage="MyPackage"
```

Note: On an Oracle database this defines the package in which the SP resides. On an SQL Server database it defines a prefix which is added to the spname parameter.

spname

(Optional) Specifies the name of an SP to use to retrieve a dynamic list of items for a list parameter. The SP returns two columns: values and text labels for each item in the list box.

Example:

```
spname="WgnWCAuditField1List"
```

cache

(Optional) Applies only to parameters which are based on stored procedures (that is, those that include the spname attribute—see above). Setting this attribute to true, allows the values retrieved from the stored procedure specified in spname to be cached. This speeds up loading of the properties page as the iConsole does not need to retrieve these values from the database each time the page is opened.

Default: false.

By default, the cache expires 5 minutes after it has first been populated. To change this default, you need to add or edit the registry value SearchParamsCachePeriodMinutes (for details, see the *Platform Deployment Guide*; search for 'timeouts, for iConsole').

restricted

(Optional) Applies only to those parameters based on stored procedures. Specifies whether 'row level security' (RLS) is switched on or off for the stored procedure used to retrieve the parameter values. If set to true, RLS is switched on, which can cause the retrieval of the parameter values to run more slowly. If set to false, searches will run unrestricted, allowing access to all data values.

Default: true

lookup_function

Defines the function used to show a dialog for selecting values for lookup and lookupbyid parameter types. The list of values is retrieved from the database and may be sorted and filtered to ease selection from a large list of items. This attribute is required for lookup and lookupbyid types.

Example:

```
lookup_function="MyUserLookupFunction"
```

For details and example usage, see Lookup Parameters (see page 62).

multiple

(Optional) Specifies a flag to indicate that more than one item may be selected from a list type parameter.

Example:

```
multiple="true"
```

Default: Specifying this attribute sets the visible size of the list to 4 items (see **size**). There are two variants of list parameter when used with the multiple flag. For details see List Parameters (see page 62).

size

(Optional) Specifies the maximum number of characters that can be displayed in the input for text, numeric, real, lookup or lookupbyid parameters. For list parameters, it specifies the number of items to be displayed.

Default: 1 for a normal list; or 4 when multiple is selected.

Example:

```
size="3"
```

rows

(Optional) Specifies the number of rows to display in a ‘text’, 'lookup' or lookupbyid parameter.

tooltip

(Optional) Specifies the text displayed as a tooltip when the mouse pointer hovers over the parameter.

Example:

```
tooltip="Specify the audit status of the emails you want to review."
```

align

(Optional) Specifies the horizontal alignment of the field: left, center, right. For check boxes, the text label is placed to the right if align="left" and vice versa.

valign

(Optional) Specifies the vertical alignment of the field: top, middle, bottom.

col

(Optional) Specifies the column where this parameter should be placed in a grid. By default a parameter will be specified in the next column. If the value of col is less than the current column a new row is started.

Example:

```
col="1"
```

Note: This attribute is only supported when requires="12.5" (or later).

colspan

(Optional) Specifies how many columns this parameter spans in the current parameter line.

Example:

```
colspan="2"
```

row

(Optional) Specifies the row where this parameter should be placed in a grid.

Example:

```
row="2"
```

Note: This attribute is only supported when requires="12.5" (or later).

rowspan

(Optional) Specifies how many rows this parameter spans in the current parameter group.

width

(Optional) Specifies the cell width for the current parameter.

Example:

```
width="50%" or width="120px"
```

hidden

(Optional) Specifies whether the parameter should be hidden by default.

Example:

```
hidden="true"
```

report

(Optional) If search attribute show_parameters="true" (see Default Element), this specifies whether the current parameter is shown in a search report. This attribute defaults to true so report="false" must be specified to disable the display of the parameter in the report.

Example:

```
report="false"
```

Default: true

report_label

(Optional) If parameter attribute report="true" (see above), this specifies a text label for the parameter in the report and overrides the default label attribute described earlier in this topic.

Example:

```
report_label="reviewed emails"
```

report_value

(Optional) Specifies a value to be displayed on a report rather than the value passed to the stored procedure. This attribute gets populated automatically for list and choice parameter types, with the item value (generally an index number) being passed to the stored procedure and the report_value being set to the item text.

style

(Optional) Specifies a CSS style for this parameter. This is used typically to specify text formatting.

Example:

```
style="font-weight: bold"
```

time

(Optional) Specifies the precision required for the time component in date and date_range parameters (see Customizing Time Display in Date Parameters (see page 95)). The following values are possible:

none

no time shown

hour

only hours shown

minute

hours and minutes shown

second

hours, minutes and seconds shown (default)

Example: Use the following to hide the seconds component:

```
time="minute"
```

subsection

(Optional) Specifies the name of a subsection to be controlled by this parameter when type="checkbox". This is used to display optional parameters.

Example:

```
subsection="optionalParameters"
```

<parameter_group> Element

The <parameter_group> element in an XML search definition defines a logical group of search parameters arranged as a subtab on the Edit Properties page.

The supported entities are as follows:

<parameter>

(Optional) See <parameter> Element (see page 131).

<parameter_line> (deprecated)

(Optional) See <parameter_line> Element (see page 140).

<subsection>

(Optional) See <subsection> Element (see page 149).

label

(Optional) Defines the group label. This text label is used for the group subtab.

Example:

```
label="Audit State"
```

tooltip

(Optional) Specifies the text displayed as a tooltip when the mouse pointer hovers over the group subtab.

Example:

```
tooltip="Search for emails by attribute (for example, blockings or warnings) or trigger name."
```

collapse (no longer supported)

(Optional) Specifies whether the group is collapsed by default.

Example:

```
collapse="true"
```

width

(Optional) Specifies the width of the parameter group in the iConsole screen.

Example:

```
width="100%"
```

init

(Optional) Specifies the name of a javascript function that is called to initialize the parameter group. The class is called the first time the group tab is clicked.

Example:

```
init="MyInit"
```

<parameter_line> Element (Deprecated)

The <parameter_line> element in an XML search definition defines a logical group of search parameters arranged in a single row in the search screen.

Note: This element is deprecated and is not supported when requires="12.5" is specified.

The supported entities are as follows:

<parameter>

See <parameter> Element (see page 131).

width

(Optional) Specifies the width of the parameter line in the iConsole screen.

Example:

```
width="100%"
```

<parameters> Element

The <parameters> element in an XML search definition defines the full set of parameters for a search.

The supported entities are as follows:

<parameter>

(Optional) See <parameter> Element (see page 131).

<parameter_group>

(Optional) See <parameter_group> Element (see page 139).

<parameter_line> (deprecated)

(Optional) See <parameter_line> Element (see page 140).

script

(Optional) Specifies a script to control the behavior of the Edit Properties page locally on the browser.

<results> Element

The <results> element in an XML search definition defines the search results attributes including the column and toolbar button definitions.

The supported entities are as follows:

<column>

(Optional) See <column> Element (see page 127).

<tool>

(Optional) See <tool> Element (see page 151).

<script>

(Optional) Specifies a script to control the behavior of the Search Results page locally on the browser.

row(Optional) -function

Specifies the action invoked when a reviewer clicks an individual row in the table of results (see Actions When a Row Is Clicked (see page 73)). This attribute specifies the function that invokes the action.

Example: This attribute is used by the Standard Search to show details about the selected event:

```
row_function="_EventDetails"
```

Note: Always used in association with the ref attribute.

ref

(Optional) Specifies up to three columns (comma-separated) to be used as the unique reference for a 'row action'. The results element must include columns with these names (column names are specified using the name attribute).

Example: This attribute is used by the Standard Search to show details about the selected event and to define key values for a context search:

```
ref="event_uid,capturets,part_index"
```

Only the first value is passed as a parameter to the row_function. If required, the second and third values can be accessed in the function using oEvent.currentRefB and oEvent.currentRefC respectively.

Also used to identify the key values to use for an update search (see Updating Results in the Cache (see page 95)).

ref2

(Optional) Specifies a column to be used as the unique reference for use by custom tools.

Example: This attribute is used by the Content Search results screen to show details about the selected event:

```
ref2="doc_uid"
```

Note: Always used in conjunction with a tool element.

ref3

(Optional) Defines a column in the result that should be dynamically updated from a javascript function.

This functionality is used for the event review process; when the audit status of an event is changed, the displayed audit status for the selected event is update in place. Used in conjunction with the 'update' attribute.

Example:

```
ref3="audit"
```

page_size

(Optional) Specifies the page size for the results, overriding the default. If a particular search needs to always display 10 results per page, this parameter can be used to do it.

Example:

```
page_size="10"
```

help

(Optional) Specifies the path and file name of the HTML page containing the help for the search results page of the current custom search. The path is relative to the front-end Web server virtual directory.

Example:

```
help="help/htm/MyEmailSearchResultsHelp.htm"
```

help_icon

(Optional) Specifies the path and file name of the image used for the help button for the search results page of the current custom search. If this parameter is not specified, the default image is used. The path is relative to the front-end Web server virtual directory.

Example:

```
help_icon="help/htm/MyHelpButton.gif"
```

Note: In the search results screen, the help button always appears next to the navigation buttons at the top of the page. See the example search page screenshot.

row_style_ref

(Optional) Specifies a column in the results that will define the name of a CSS class in a stylesheet that will be used to define the style for each row.

The stored procedure is responsible for defining the class for each row; a NULL value will apply no class.

A selector may also be required for the cell in order to override the default styles.

Example: For a style class of 'highlight', the stylesheet needs to include the selector:

```
.highlight, .highlight td {…}
```

The precedence of certain styles (e.g. font-size) may need to be forced by using the '!important' rule.

```
.bigfont {font-size: x-large !important;}
```

update

(Optional) Used to indicate that an update stored procedure is available for this search, this can then be invoked by the ToolbarUpdate function or the UpdateQuery method. The update procedure can be used to run a simple search to retrieve updates to refresh the cached results without having to re-run the entire original search. See Updating Results in the Cache (see page 95).

This functionality is used in the event review process. When the audit status of an event is changed, the update SP is run to update the cached results with the new audit status. Used in conjunction with the 'ref3' attribute; this defines the displayed results column to be updated on the page.

Example:

```
update="true"
```

format

(Optional) Define the name of a custom format for laying out the results. The format must be defined in results-formats-custom.xsl. See details of custom formatting. (see page 81)

sortable

(Optional) When set to 'false' specifies that the results should not be sortable by clicking on the column headings. Default is 'true'.

show_totals

(Optional) Specifies that totals row(s) should be displayed on the search results page. See Displaying Aggregate Totals (see page 87).

<search > Element

The Default element in an XML search definition defines an individual search, including the SP reference, screen labels, category and class, and whether the search displays in tabular format or in a report.

The supported entities are as follows:

<parameters>

See <parameters> Element (see page 140).

<results>

See <results> Element (see page 141).

no_settings

(Optional) Specifies whether to hide the Settings tab in the Search Properties dialog. By default, this tab is displayed and contains options such as the Column Selector and Display Sort Order. Set this attribute to 'true' to hide the Settings tab:

```
no_settings="true"
```

spname

(Optional) Defines the name of the stored procedure that performs the search. This name is combined with the major and minor version numbers (see below) to generate the name used to call the stored procedure.

Example:

```
spname="MyEmailSearch"
```

You can create multiple search variants that all refer to the same stored procedure: Create search definitions with a different label but the same stored procedure name.

Example:

```
spname="simple.MyEmailSearch"
```

This specifies a search variant named 'simple' for the MyEmailSearch SP.

Limits: 64 characters

sppackage

(Optional) Specifies the package name for the SP (Oracle only).

Limits: 30 characters

major

Defines the required major version number of the search. Search definitions with different major versions are not guaranteed to be compatible.

Example:

```
major="2"
```

minor

(Optional) Defines the minor version number of the search. Search definitions with different minor versions but the same major version are guaranteed to be compatible with existing customized searches.

Example:

```
minor="1"
```

label

Defines the required text label used to identify the search in the iConsole search screens.

Example:

```
label="My First Search"
```

Limits: 64 characters

description

(Optional) Defines a brief description of the search. This is shown below the search label in the iConsole Search screen.

Example:

```
description="Finds emails with specific titles"
```

Limits: 255 characters

class

(Optional) Specifies the search class which determines the set of predefined tools (that is, toolbar buttons) that are available in the Search Results screen, and the privileges required to access the search.

For the current release, the following search classes are supported:

```
class="EventReview"
class="EventReport"
class="Administration"
```

The first two classes support Security ID checking when this is enabled (see Security ID Processing (see page 167)) and require Events: View captured data and either Events: Allow event searches for metadata searches or Events: Allow content searches for content searches.

help

(Optional) Defines the path and file name of the HTML page containing the help for the current custom search. The path is relative to the front-end Web server virtual directory.

Example:

```
help="help/htm/MyEmailSearchHelp.htm"
```

help_icon

(Optional) Defines the path and file name of the image used for the help button for the current custom search. If this parameter is not specified, the default image is used. The path is relative to the front-end Web server virtual directory. For example:

```
help_icon="help/htm/MyHelpButton.gif"
```

Note: In the search screen, the help button always appears below the custom help parameters and before the default sort and save options. See the example search results screenshot in Example iConsole Screens.

quick

(Optional) Specifies whether you want to bypass the Search Progress page. If the search typically runs very quickly, set this to 'true'.

Default: false

Example:

```
quick="true"
```

Note: Quick searches cannot be canceled.

refresh

(Optional) Specifies an interval (in seconds) between automatic refreshes of the search results. For example, you can use this attribute to automatically update simple searches that return a statistical counter.

Example: Use the following value to update the results every minute:

```
refresh="60"
```

category

(Optional) Specifies under which category the search is listed. You can organize published searches into categories in the 'Published' folder in the iConsole.

Example:

```
category="Compliance breaches"
```

Limits: 20 characters

type

(Optional) Specifies the type of search. Typically, this attribute is only used to specify 'System' searches, listed under the System folder in the iConsole. The full range of supported search types is:

```
type="system"
```

```
type="published"
```

```
type="unpublished"
```

database

(Optional) Specifies which type of database to access. This can be:

```
database="cache"
database="cms "
database="combined "
database="content "
```

max_rows

(Optional) Specifies the maximum number of rows that can be cached for in-memory sorting. This attribute overrides the system-wide maximum defined in the registry.

Limits: 20 characters

report

(Optional) Specifies whether search results are displayed in a report.

Example:

```
report="true"
```

show_parameters

(Optional) Specifies whether the search parameters used in the search are displayed in the resulting report where report="true".

Example:

```
show_parameters="true"
```

no_cache

(Optional) Specifies that the search results are not cached on the iConsole application server. This attribute is intended for use with searches that only retrieve a single record. Its purpose is to avoid loading the cache unnecessarily, which could undermine the performance of subsequent searches.

Example:

```
no_cache="true"
```

requires

(Optional) Specifies the version of the iConsole product (both web and application servers) that is required to support this search definition. You need to specify the version if the search definition takes advantage of a new feature (for example, parameter type) which is only available in this version. For details see Search Visibility.

Example:

```
requires="12.5"
```

tabular

(Optional) Specifies whether to allows searches that return only a single result to format this as a list of column values (false) rather than in a table (true). Used in conjunction with the report attribute.

Example:

```
tabular="false"
```

target (no longer supported)

(Optional) Specifies a target XML stylesheet to format the results in place of the default results page. This requires the development and installation of an XML stylesheet (for example, MyResults.xsl).

Advanced use only.

Example:

```
target="MyResults"
```

Limit: 20 characters

restricted

(Optional) Specifies whether the search should be run restricted (that means, apply RLS - the default), or unrestricted. In a derived search restricted may be set to true only if the baseline search is set to false, such as a derived search can only be made more restrictive.

Advanced use only.

Example:

```
restricted="false"
```

baseline

(Optional) For a manually created derived search, specifies the name of the search upon which the derived search is based.

Advanced use only.

base_section

(Optional) For a manually created derived search, specifies the location of the search upon which the derived search is based.

Advanced use only.

legal

(Optional) Specifies a legal notice to be included in the search definition. If prefixed with '*' has the same effect as task_page="true".

Example:

```
legal="© Copyright CA 2010. All rights reserved."
```

task_page

(Optional) Specifies whether the search should have a tab of its own on the iConsole main page.

Example:

```
task_page="true"
```

must_edit

(Optional) Specifies that the search should open in the properties window (true) rather than in the results window (false). Setting this attribute to true allows the user to specify parameters for each run. A typical case where the user generally wishes to specify a search term is the Content Search.

Example:

```
must_edit="true"
```

<searches> Element

The root <searches> element in an XML search definition contains one or more search definition records.

The supported entities are as follows:

dictionary

(Optional) Defines the string dictionary to be used for translating the searches within this search definition file. For details, see Localization (see page 115).

<search>

For details, see <search> Element. One or more <search>elements are required.

<subsection> Element

The <subsection> element in an XML search definition defines a collection of parameters that should be grouped together. They will be displayed in a separate panel that can be collapsed when those parameters are not of interest.

Note: This element is only supported when requires="12.5" (or later).

The supported entities are as follows:

<parameter>

(Optional) See <parameter> Element (see page 131).

name

Defines the required name by which the subsection can be identified.

Example:

```
name="optionalParameters"
```

label

(Optional) Defines a text label to be used as the title for the subsection panel.

Example:

```
label="Optional Parameters"
```

tooltip

(Optional) Defines the text displayed as a tooltip when the mouse pointer hovers over the subsection.

Example:

```
tooltip="Show optional parameters."
```

col

(Optional) Specifies the column where this subsection should be placed in a grid. By default it will be specified in the next column. If the value of col is less than the current column a new row is started.

Example:

```
col="1"
```

row

(Optional) Specifies the row where this subsection should be placed in a grid.

Example:

```
row="2"
```

colspan

(Optional) Specifies how many columns this subsection spans in the current.

Example:

```
colspan="2"
```

rowspan

(Optional) Specifies how many rows this subsection.

Example:

```
rowspan="2"
```

collapse

(Optional) Specifies a flag indicating whether the subsection should be collapsed by default.

Example:

```
collapse="true"
```

<tool> Element

The <tool> element in an XML search definition defines a custom tool in the toolbar of the Search Results page.

The supported entities are as follows:

name

Specifies the required name of the tool.

tooltip

(Optional) Specifies the tooltip text shown when the mouse pointer hovers over the toolbar button.

Example:

```
tooltip="My button"
```

icon

Specifies the required icon file for the toolbar button.

Example:

```
icon="MyButton.gif"
```

The image file must be in the \images folder in the CA DataMinder installation folder on the iConsole front-end Web server.

function

Specifies the required function that is called when a reviewer clicks the toolbar button. See Standard Scripting Functions (see page 167).

Example:

```
function="MyTool"
```

Chart Template Parameters

This is a list of all of the svg-chart:draw template parameters that can be used to customize the layout of a chart.

Parameter Name	Default	Description
data	-	An XML fragment containing the data series to be charted
type	Pie	The type of chart to be drawn: ■ pie ■ bar ■ column ■ line
width	100	The width of the chart object in the containing document (mm)
height	-	The height of the chart object in the containing document (mm), usually calculated automatically.
chart-width		The width of the chart in plot units. Usually calculated automatically from the other parameters.
chart-height		The height of the chart in plot units. Usually calculated automatically from the other parameters.
options	category-labels	Optional chart elements. Space separated list of: ■ category-labels ■ legend ■ data-labels ■ percentage ■ grid-x ■ grid-y ■ axis-x ■ axis-y
caption	-	Text for the caption for the chart
caption-x	120	Horizontal position of the caption
caption-y	240	Vertical position of the caption

Parameter Name	Default	Description
caption-style	text-anchor: middle; font-size:10pt; font-weight: bold;	Style for the caption text
colors	Built-in color series of 16 colors.	List of colors in RGB format and separated with spaces e.g. #aabbcc #ffeedd #112233 etc. each color must be 8 chars long.
plot-x	20	Horizontal position of the plot area
plot-y	20	Vertical position of the plot area
plot-width	200	Width of the plot area
plot-height	200	Height of the plot area
legend-x	250	Horizontal position of the legend area
legend-y	20	Vertical position of the legend area
legend-width	140	Width of the legend area
legend-height	200	Height of the legend area
legend-style	font-size: 8pt; stroke-width: 0.5; stroke: black;	Style for the legend.
right-margin	20	Width of right margin
bottom-margin	20	Width of bottom margin
category-x-angle	-45	Angle of category labels on x axis
category-y-angle	0	Angle of category labels on y axis
category-style		Style of category labels

Parameter Name	Default	Description
axis-style	font-family: sans-serif; font-size: 8pt; text-anchor: end;	Style of axis labels
data-style	-	Style of data bars or pie slices
data-text-style	font-family: sans-serif; font-size: 8pt; text-anchor: end;	Style of data value text
debug	no	When 'yes' embeds a comment containing the chart parameters in the output.

More information:

Graphics and Charts (see page 103)

Content Search Result Columns

This is a list of predefined column names that can be used to define columns in the results of a Content Search; that is, where database="content". .

doc_id

Returns the document ID from the content database. For example:

```
<column name="doc_id" type="id" hidden="true" />
```

Required: No

relevance

Returns a confidence level (as a percentage). This indicates how closely the document fits the search criteria, with a higher score indicating a closer fit. For example:

```
name="relevance"
```

Required: No

event_uid

Returns the event UID. For example:

```
name="event_uid"
```

Required: No

attachment

Returns an icon indicating that the result is an attachment. For example:

```
name="attachment"
```

Required: No

event_type

Returns an icon indicating the event type. For example:

```
name="event_type"
```

Required: No

subject

Returns the event subject. For example:

```
name="subject"
```

Required: No

timestamp

Returns the event timestamp. For example:

```
name="timestamp"
```

Required: No

retrieval_ts

Returns the timestamp to be used for event retrieval. Currently, this will be the same as timestamp for an Oracle CMS but zero for a SQL Server CMS as the timestamp resolution is different between the Content Database and SQL Server CMS. For example:

```
name="retrieval_ts" type="timestamp" hidden="true"
```

This column should be specified in the results ref attribute, for example:

```
ref="event_uid,retrieval_ts"
```

Required: No

participants

Returns a list of all event participants. For example:

```
name="participants"
```

Required: No

sender

The person that sent the email/posted the IM/accessed the web site.

Required: No

recipients

The people who received the email/saw the IM/the URL of the web site.

Required: No

Content Searches <parameter>

These parameter names are available for defining content searches. They define input fields that can be used as content search parameters. **You must not rename these parameters!**

The supported attributes are as follows:

lstMachine

Specifies the content proxy server. This can be a drop-down list or a hidden text input box set to a predefined content proxy.

Whatever the parameter type, its name **must** be lstMachine.

Required: No

txtSearchTerm

A text input box for specifying the search text, that is, the key words or phrases to look for in an e-mail or attachment.

Required: No

chkEE

A check box for choosing whether to search for e-mail events.

Required: No

lstEE

Specifies a drop-down list for choosing the scope of an e-mail search. The supported options (that is, list items) are:

1 Return internal and external events

2 Return only internal events

3 Return only external events

Required: No

txtEEAddress1

A text input box for specifying the From: address.

Required: No

lstEEDir

Specifies a drop-down list for choosing the e-mail direction. The supported options (that is, list items) are:

1 Return incoming and outgoing events

2 Return only incoming events

3 Return only outgoing events

Required: No

txtEEAddress2

A text input box for specifying the To: address.

Required: No

chkIME

A check box for choosing whether to search for IM events.

Required: No

lstIME

Specifies a drop-down list for choosing the scope of a search for IM events. The supported options (that is, list items) are:

1 Return internal and external events

2 Return only internal events

3 Return only external events

Required: No

chkFE

A check box for choosing whether to search for File events.

Required: No

chkME

A check box for choosing whether to search for 'My events' only. When selected, this retrieves events associated with the current user, that is, the account they currently logged on to the CMS with.

Required: No

txtNameMatch

A text input box for specifying the name, or name fragment, of a CA DataMinder user. The search returns events associated with this user.

Required: No

txtSpecificMatch

A field for specifying the name, or list of names separated by semicolons, of CA DataMinder users. This parameter typically uses a lookup_function to allow the reviewer to browse the user hierarchy and choose specific users. The search returns events associated with these users.

Required: No

txtUserIDs

A field for specifying a list of user IDs separated by semicolons or CRLF. This will usually be populated using the lookupbyid parameter type and PopupWithHiddenField lookup_function.

Required: No

txtGroupIDs

A field for specifying a list of group IDs separated by semicolons or CRLF. This will usually be populated using the lookupbyid parameter type and PopupWithHiddenField lookup_function.

Required: No

chkIncSubGrps

A check box to indicate that the search is to also include events from groups that are descendants of the specified groups.

Required: No

dateRange

A date field for specifying the search period.

Required: No

numConfidence

A numeric input field for specifying the confidence level (as a percentage) for the search results.

This is an estimate of how relevant a document is to the main search 'theme'. Documents that meet the search criteria but which do not have a high enough confidence level are excluded from the results.

Required: No

numRowLimit

A numeric input field for specifying the maximum number of results that the search can return.

Required: No

txtSimilar

This input field is normally hidden. It is populated automatically with document IDs when the user runs the 'More Like This' search from the Search Results screen.

Required: No

numSampling

Specifies the result sampling rate. The value is the sampling percentage but this will only give accurate results in a content search when the reciprocal is a whole number e.g. 1/20% = 5 so 1 in 5 results will be returned, however 1/15% = 6.667 and 1 in 6 will be returned.

Recommended options are:

100 Return all results

20 Return 1 in 5 results (20%)

10 Return 1 in 10 results (10%)

5 Return 1 in 20 results (5%)

1 Return 1 in 100 results (1%)

Required: No

choSort

Specifies a list of sorting options. The supported options (that is, list items) are:

1 Sort by Relevance

2 Sort by Descending Date

Required: No

choResultsClass

Specifies a compulsory post-processing option that enables the iConsole to interpret the search results as, for example, e-mail or IM events. The supported options are:

Email

For e-mail and IM searches.

IM

For IM searches.

Required: Yes

Context Searches <parameter>

These parameters are used to configure context search results. They define the number of 'before' and 'after' events shown when a reviewer chooses to display context search results. These are events associated with the current event's participants that were captured immediately before and after the current event.

The supported attributes are as follows:

no_events

Specifies the maximum number of events that can be listed before and after the current event when it is reviewed in context.

Defaults to 62.

For example, if you set value=<10> you can get up to 21 results: 10 before and 10 after the selected event, which is displayed in the middle of the context search results list.

External Searches and Bookmarks

The iConsole provides mechanisms for accessing a search using a URI (Uniform Resource Identifier).

- The iConsole bookmark facility automatically generates a URI to execute or customize the currently selected search
- An external search facility provides a means of running a search with custom parameters which could for example, be used to provide drilldown capability in a report.

A typical iConsole URI has the form:

```
http://<server>/CA/Default.aspx?external="<type>"&parameters="<request>"
```

where <server> is the name of the iConsole Server <type> is the type of external request and <request> contains details for the request.

For a bookmark:

```
<type> = bookmark
```

```
<request> = <CMS>, <bookmark_type>, <search name>, <major>, <minor>,
<section>,<category>, <version>, <target>, <timezone>
```

where:

<CMS>

The name of the CMS that the iConsole is to connect to.

<bookmark_type>

'search' to run the search or 'customize' to customize the search

<search name>

The name of the iConsole search

<major>

The major version number of the search

<minor>

The minor version number of the search (-1 for latest, -2 for derived)

<section>

The section containing the search (e.g. 'published' or 'custom' for a saved search)

<category>

The category folder containing the search (e.g. Reports)

<version>

The URI format version number (current values are 3 for bookmarks and 2 for external searches)

<target>

The target page for the search (usually blank)

<timezone>

The time zone information to be used for the search (from bookmark version 3).

This takes the form tzoffset:dst where:

tzoffset

Is the offset (in minutes) from UTC for the time zone on 21 December (for example -300 for EST)

dst

Is the difference in the offset on 21 June due to DST corrections (for example 60 for EST)

Note: The time zone definition was different for bookmark version 2. The values had the opposite sign and It took the form todayoffset: tzoffset: dst where:

todayoffset

Is the number of minutes from UTC today

tzoffset

Is the number of minutes from UTC on 21 December (for example 300 for EST)

dst

Is the number of minutes for DST (for example -60 for EST)

For example, this bookmark would run the latest version 2 standard search on 'myserver' connecting to CMS 'myCMS' in the EST time zone.

```
http://myserver/CA/Default.aspx?external=%22bookmark%22&parameters=%22myCMS,
Search,Standard%20Search,2,-1,published,,3,,-300:-300:60%22
```

Note: Here, the special characters quotes(") and space have been URL Encoded as %22 and %20 respectively.

For an external search:

```
<type> = search

<request> = <search name>, <major>, <minor>, <section>, <category>, <timezone>,
<version>
```

The external search is more powerful because search parameters can be specified by appending them to the end of the URI in the form &<name>=<value> e.g. &chkEE=true for the standard search would search for email events. If the name is enclosed in square brackets the value is interpreted as the parameter attributes. This is useful for date ranges e.g. &[dateRange]=2;1;2;1 would set the date range to 'All Dates'.

Note: To specify time zone information, the external search version must be set to 2.

No CMS is specified with an external search; if more than one CMS is available the user would be expected select an appropriate CMS when logging on, or to be authenticated by SSO.

Parameter Type Mappings

For details about the type attribute for <parameter> elements (see page 131). The supported types are mapped as follows:

<parameter> type	Oracle type	SQL server type
text	VARCHAR2	NVARCHAR()
numeric (long)	NUMBER	BIGINT
numeric (short)	PLS_INTEGER	INT
date	DATE	DATETIME
date_range (2 parameters passed)	DATE	DATETIME
checkbox	PLS_INTEGER	BIT
list	PLS_INTEGER	INT
choice	PLS_INTEGER	INT
systemid	NUMBER	BIGINT
real	FLOAT	REAL

Registry Settings

The following table list registry settings affecting searches in the iConsole:

Name	Key	Type	Description
AllowUnlimitedSearch	WebService	REG_SZ	A Boolean value indicating that unlimited searches are supported, if the user has sufficient privilege.
DefaultDashboard (deprecated)	WebService	REG_SZ	The definition of a dashboard search to be displayed as the default dashboard on the Tasks page.
MaximumResultSetSize	WebService	REG_DWORD	The maximum number of records that a search can return. Default is 1000. Note however that this can be overridden if the user has sufficient privilege.
SearchJavascriptValidationOptions	WebService	REG_SZ	JavaScript compiler options controlling the compilation process e.g. level of error reporting. Default is '/warn:3 /fast-'.
SearchResultsConverterTimeoutSecs	WebService	REG_DWORD	The time that the iConsole will wait for a response from a download converter to respond. Default is 90s.
SearchResultsDownloadConvertTo<X>	WebService	REG_SZ	The command to convert the search results into a specific format, where <X> specifies the format. The value is the path to the fop.bat command file. For example, to convertfrom XSL-FO to PDF using Apache FOP, use: SearchResultsDownloadConvertToPDF
SearchResultsDownloadExt	Web	REG_SZ	The default file extension to be used for search results downloads. The default is xml as the Excel download format is in XML Spreadsheet format.

Name	Key	Type	Description
SearchResultsDownloadFormat	Web	REG_SZ	The default format for search results downloads. Built-in formats are xls (Excel) and csv. The default is xls.
ValidateSearchJavascript	WebService	REG_SZ	A Boolean value indicating whether search definition files are to have embedded javascript validated. Default is 'false'.

Reserved Parameter and Column Names

All parameter and column names beginning with $ are reserved for use by the iConsole. These reserved parameters are interpreted as having special meaning by the iConsole.

Name	Parameter/Column	Type	Description
$CONTROL	parameter	choice	This parameter signals to the iConsole when the results from the search will be a total record count or an unlimited result set, rather than a normal, limited result set.
$DEBUG	parameter	numeric	This parameter is used to request logging in the search stored procedure. Although no special processing is performed in the iConsole, it is defined as a reserved word for future enhancement.
$GROUPS	Parameter	lookupbyid	When a group lookup parameter begins with '$GROUPS', the report value for the parameter is set to the list of the user's management groups when no groups are selected. (or the top level of the hierarchy if the user has the Admin: Disable Security Group Filtering privilege).
$ROWLIMIT	parameter	numeric	This parameter is used by the stored procedure to limit the number of results returned; this can make a significant difference to the performance of the procedure. If the value specified exceeds the currently defined system results limit the iConsole constrains it to that value. This is a performance optimization for SQL Server.
$SID	column	text	This column is used to return the Security ID associated with the record.

Search Definition Syntax

Conventions: In the topics that follow:

- XML elements are in brackets, for example: <parameters>
- Attributes for XML elements are without brackets, for example: spname

Search Result Column Type Mappings

For details about the type attribute for <column> elements (see page 127). The supported types are mapped as follows:

Column type	Oracle type	SQL server type
id	NUMBER	BIGINT
text	VARCHAR2	NVARCHAR()
numeric	NUMBER	BIGINT
date	DATE	DATETIME
timestamp	DATE	DATETIME
icon	VARCHAR2	NVARCHAR()
list	VARCHAR2	NVARCHAR()
link	VARCHAR2	NVARCHAR()
quantity	NUMBER	BIGINT
boolean	PLS_INTEGER	BIT
real	NUMBER(38,n)	REAL

Security ID Processing

Some customers require additional security measures to ensure that users are only permitted to view events that they are allowed to see. This is implemented by adding a Security ID to each event on capture and checking that this ID is included in a list of IDs permitted to the user.

In order to support this capability in iConsole searches the only requirement is that the search should be of class EventReview or EventReport and return the Security ID in a column named $SID.

The security mechanism is enabled by defining a registry setting on the iConsole server back end with a key of UserSIDAttribute. This setting identifies the name of a user attribute that will contain the permitted Security IDs for the user.

When this feature is enabled, an error will be produced if:

- The UserSIDAttribute setting is blank.
- The search is of class EventReview or EventReport and does not return a column named $SID.
- The results contain a Security ID that is not in the user's list of permitted SIDs.

Standard Scripting Functions

These are predefined functions that can be associated with hyperlinked rows or columns or with toolbar buttons in the Search Results screen.

Function name	Calling element and attribute	Ref column	Description
_UserDetails	<column> link_function	yes	Displays user information, including user name and group history.
_EventDetails	<column> link_function <Results> row_function	yes yes	Displays the event details panel at the bottom of the Search Results screen, giving full details about the event and providing review capabilities.
DownloadResults	<tool> function		Downloads the complete search results. The format is specified by a string parameter passed to the function containing the format type (e.g. 'xls')
RunQuery	Called by a function in the results <script> element		Runs a sub-query e.g. to show drilldown information.

ShowResultsComponent	Called by a function in the results <script> element	Shows or hides the results component depending on the single Boolean parameter. Shows the component if 'true'.
ShowResultsTable	Called by a function in the results <script> element	Shows or hides the results table depending on the single Boolean parameter. Shows the table if 'true'.

search-helper.js

These are the helper functions available by including search-helper.js. Most functions have two forms, one taking the object to be operated on enabling 'this' to be passed from event handlers, and the other taking the name of the parameter.

Note: These functions must be used for updating parameters from javascript, changing values using document.getElementById('myparam').value will not function correctly. This is because the way that data is managed in r12.5 is substantially different from previous releases.

Function name	Description	Argument Name	Argument Description
SH_DisableCheckbox	Changes the disabled state of a checkbox parameter.	obj	The checkbox object
		bDisabled	Disable the checkbox if true
SH_DisableCheckboxByName	As SH_DisableCheckbox but using the checkbox name	sName	The name of the checkbox parameter
		bDisabled	Disable the checkbox if true
SH_DisableList	Changes the disabled state of a list parameter or selected list items.	obj	The list object
		bDisabled	Disable the list or list items if true
		aItems	(Optional) An array of item values to be disabled.
SH_DisableListByName	As SH_DisableList but using the list name.	sName	The name of the list parameter.

Function name	Description	Argument Name	Argument Description
		bDisabled	Disable the list or list items if true.
		aItems	(Optional) An array of item values to be disabled.
SH_DisableLookup	Changes the disabled state of a lookup parameter, including associated buttons	obj	The lookup object
		bDisabled	Disable the lookup parameter if true
SH_DisableLookupByName	As SH_DisableLookup but using the checkbox name	sName	The name of the lookup parameter
		bDisabled	Disable the lookup parameter if true
SH_DisableParameter	Changes the disabled state of a parameter. This function handles all parameter types.	obj	The parameter object
		bDisabled	Disable the parameter if true.
SH_DisableParameterByName	As SH_DisableParameter but using the parameter name	sName	The name of the parameter.
		bDisabled	Disable the parameter if true
SH_DisableParameterList	Changes the disabled state of a list of parameters.	sList	The comma-separated list of parameter names.
		bDisabled	Disable the parameters if true
SH_EscapeXmlText	Escapes special characters in the text to ensure valid xml for drilldown parameters.	sText	The text to be escaped.
SH_ExclusiveCheckbox	Disables a list of parameters based on the state of the specified checkbox.	obj	The checkbox object.

Function name	Description	Argument Name	Argument Description
		sList	The comma-separated list of names of parameters to be disabled when the checkbox is ticked.
SH_ExclusiveCheckboxByName	Disables a list of parameters based on the state of the specified checkbox.	sName	The checkbox parameter name
		sList	The comma-separated list of names of parameters to be disabled when the checkbox is ticked.
SH_GetName	Gets the name of the parameter or its id if the name is not set.	obj	The parameter object.
SH_SetCheckbox	Sets the checked state of a checkbox parameter. N.B. This function should always be used to set a checkbox programmatically.	obj	The checkbox object.
		bChecked	Tick the checkbox if true
SH_SetCheckboxByName	As SH_SetCheckbox but using the parameter name.	sName	The checkbox parameter name
		bChecked	Tick the checkbox if true
SH_SetListItems	Sets the select state of a list.	obj	The list object.
		bSelected	Item(s) selected when true.
		aItems	An array of item values to be selected.
		bOnChange	(optional) Prevents the OnChange method from firing when false.
SH_SetListItemsByName	Sets the select state of a list but using the list name.	sName	The list parameter name.
		bSelected	Item(s) selected when true.

Function name	Description	Argument Name	Argument Description
		aItems	An array of item values to be selected.
		bOnChange	(optional) Prevents the OnChange method from firing when false.
SH_SetTextBox	Sets the text in a textual parameter.	obj	The text object.
		sText	The text to set.
SH_SetTextBoxByName	Sets the text in a textual parameter	sName	The text parameter name
		sText	The text to set.
SH_ClearLookup	Clears a lookup or lookupbyid parameter.	obj	The lookup object.
SH_ClearLookupByName	Clears a lookup or lookupbyid parameter.	sName	The name of the lookup parameter.
SH_UpdateMultiSelect	Update a multi-select list parameter which has an [All] option. De-selects any other selections in the list when the [All] option is selected	obj	The list object
SH_UpdateMultiSelectByName	As SH_UpdateMultiSelect but using the parameter name	sName	The list parameter name
SH_ValidateNumberInRange	Validates the value entered into a numeric parameter as being within the specified range. If not, the value is reset to the default value	obj	The numeric parameter object
		nMin	The range minimum
		nMax	The range maximum
		nDefault	The default value (may be null)

Function name	Description	Argument Name	Argument Description
SH_ValidateNumberInRangeByName	As SH_ ValidateNumberInRange but using the parameter name	sName	The numeric parameter name

Standard SQL Helper Functions

Hierarchy: WgnWC_Hier Collection (Deprecated)

Functions

Function name	Type Oracle or MSSQL	Parameter Type Oracle or MSSQL	Returns Oracle/MSSQL	Description
ExpandGroups	Function		WGNWC_HIER_EXPAND_GROUP_TAB/ TABLE(GroupIDM INTEGER, GroupID INTEGER, RollupIDM INTEGER, RollupID INTEGER, ManagerIDM INTEGER, ManagerID INTEGER, Hierarchy INTEGER, GrpLevel INTEGER, GroupName NVARCHAR(255), FullPath NVARCHAR(1000))	Expands the group hierarchy from the defined point (default is root) and returns a table of groups.
		NUMBER/ INTEGER		The IDM of the start group
		NUMBER/ INTEGER		The ID of the start group

Logging: WgnWC_Log Collection

Functions

Function name	Type Oracle or MSSQL	Parameter Type Oracle or MSSQL	Description
Close	Procedure		Closes an open log file, writing a final message
		UTL_FILE.FILE_TYPE/ NVARCHAR(255)	The log file handle, returned closed
Open	Procedure		Opens a log file, writing an initial message
		UTL_FILE.FILE_TYPE/ NVARCHAR(255)	The returned open log file handle
		VARCHAR2/ NVARCHAR(255)	Optional file prefix. Defaults to 'iConsole_Search_'
		VARCHAR2/ NVARCHAR(255)	Optional directory to store the log file. (Oracle: Name of a DIRECTORY object. Defaults to 'WGNWC_LOG_DIR') (MSSQL: Directory path. Defaults to 'C:\temp')
Write	Procedure		Writes a message to the log file.
		UTL_FILE.FILE_TYPE/NVARCHAR(255)	The log file handle
		VARCHAR2/ NVARCHAR(4000)	The message string

Utilities: WgnWC_Util Collection

Functions

Function name	Type Oracle or MSSQL	Parameter Type Oracle or MSSQL	Returns Oracle or MSSQL	Description
Action_2	Function		VARCHAR2/ NVARCHAR(255)	Returns the event action, based on the supplied attributes, in a string.
		PLS_INTEGER		The event major type (0 if unknown).
		VARCHAR2/ NVARCHAR(255)		The event attributes.

Function name	Type Oracle or MSSQL	Parameter Type Oracle or MSSQL	Returns Oracle or MSSQL	Description
AttributeValue	Function		VARCHAR2/ NVARCHAR(255)	Returns the value of the named attribute from the list of attributes in a string.
		VARCHAR2/ NVARCHAR(255)		A list of attributes in the form /A1=value1 /A2=value2 /A3=value3 etc.
		VARCHAR2/ NVARCHAR(255)		The required attribute name e.g. A2
AuditField1List	Function/ Procedure		SYS_REFCURSOR/ Resultset	Returns the current list of Audit Status Field 1 values
AuditField1ListNoAny	Function/ Procedure		SYS_REFCURSOR/ Resultset	Returns the current list of Audit Status Field 1 values without the [Any] item
AuditField2List	Function/ Procedure		SYS_REFCURSOR/ Resultset	Returns the current list of Audit Status Field 2 values
AuditField2ListNoAny	Function/ Procedure		SYS_REFCURSOR/ Resultset	Returns the current list of Audit Status Field 2 values without the [Any] item
AuditField3List	Function/ Procedure		SYS_REFCURSOR/ Resultset	Returns the current list of Audit Status Field 3 values
AuditField3ListNoAny	Function/ Procedure		SYS_REFCURSOR/ Resultset	Returns the current list of Audit Status Field 3 values without the [Any] item
AuditStatus	Function		VARCHAR2/ NVARCHAR(255)	Gets the current audit status text for the specified event or 'Multiple Issues' if there is no unique status.
		NUMBER/ INTEGER		The event UID

Function name	Type Oracle or MSSQL	Parameter Type Oracle or MSSQL	Returns Oracle or MSSQL	Description
		DATE/ DATETIME		The event timestamp
ColumnExists [MSSQL only]	Function			Returns 1 if the specified column exists
		NVARCHAR(255)		The name of the table
		NVARCHAR(255)		The name of the column
CommaEncode	Function		VARCHAR2/ NVARCHAR(765)	Replaces commas in a string with an encoded string
		VARCHAR2/ NVARCHAR(255)		The string to be encoded
DelimStringFromNodes [Oracle only]	Function		VARCHAR2	Returns a table of class nodes as a string using the specified delimiter
		WGNCLASSNODES		A table of class nodes
		VARCHAR		The delimiter, defaults to ','
DisplayName	Function		VARCHAR2/ NVARCHAR(255)	Gets the display name for the specified user
		NUMBER/ INTEGER		The IDM of the user
		NUMBER/ INTEGER		The ID of the user
EscapeString	Function		VARCHAR2/ NVARCHAR(255)	Adds an escape clause to the specified SQL LIKE string if one is required.
		VARCHAR2/ NVARCHAR(255)		The SQL LIKE string

Function name	Type Oracle or MSSQL	Parameter Type Oracle or MSSQL	Returns Oracle or MSSQL	Description
EventImage2	Function		VARCHAR2	Gets the event icon filename and tooltip text corresponding to the specified event details
		NUMBER/ BIGINT		The event UID (not used)
		DATE/ DATETIME		The event timestamp (not used)
		PLS_INTEGER/ INTEGER		The event major type
		PLS_INTEGER/ INTEGER		The event minor type
		PLS_INTEGER/ INTEGER		The event AES major type
		PLS_INTEGER/ INTEGER		The event AES minor type
		VARCHAR2		The event attributes
		PLS_INTEGER/ INTEGER		No of attachments
		NUMBER/ BIGINT		The quarantine UID (defaults to 0)
		PLS_INTEGER/ INTEGER		The transaction state (defaults to NULL)
FileParticipant	Function		VARCHAR2/ NVARCHAR(2000)	Gets the file owner or file host for a file event
		NUMBER/ BIGINT		The event UID
		DATE/ DATETIME		The event timestamp
FixedWidthString	Function		VARCHAR2/ NVARCHAR(255)	Gets a string of the specified width, right justified if it's a number
		VARCHAR2/ NVARCHAR(255)		The initial string

Function name	Type Oracle or MSSQL	Parameter Type Oracle or MSSQL	Returns Oracle or MSSQL	Description
		PLS_INTEGER/ INTEGER		The required length (in characters)
GetUsersNameHistory	Function		VARCHAR2/ NVARCHAR(2000)	Gets a comma-separated string of all full names of a user (most recent first)
		NUMBER/ INTEGER		The IDM of the user
		NUMBER/ INTEGER		The ID of the user
GroupPath	Function		VARCHAR2/ NVARCHAR(4000)	The full path to the specified group
		NUMBER/ INTEGER		The IDM of the group
		NUMBER/ INTEGER		The ID of the group
HexDecode	Function		VARCHAR2/ NVARCHAR(1000)	Returns a string with hex encoded strings replaced with the decoded characters
		VARCHAR2/ NVARCHAR(1000)		The string to be decoded
Hex2Dec [MSSQL only]	Function		INTEGER	Converts a 2 character hex string to an integer
		NVARCHAR(2)		
IDM_From_UID	Function		NUMBER/ INTEGER	Gets the IDM from a UID
		NUMBER/ BIGINT		A UID
ID_From_UID	Function		NUMBER/ INTEGER	Gets the ID from a UID
		NUMBER/ BIGINT		A UID

Function name	Type Oracle or MSSQL	Parameter Type Oracle or MSSQL	Returns Oracle or MSSQL	Description
LikeList	Function		VARCHAR2/ NVARCHAR(4000)	Returns a where clause fragment to compare the column with the list of values using the given separator
		VARCHAR2/ NVARCHAR(255)		The column name
		VARCHAR2/ NVARCHAR(1000)		The list of values to compare
		VARCHAR2/ NVARCHAR(10)		The separator used in the list
		VARCHAR2/ NVARCHAR(10)		The Boolean operation between list items i.e. 'AND' or 'OR'
		VARCHAR2/ NVARCHAR(255)		The prefix to be added e.g. % to match the start of the column value
		VARCHAR2/ NVARCHAR(255)		The suffix to be added e.g. % to match the end of the column value
LikeList2	Function		VARCHAR2/ NVARCHAR(4000)	Returns a where clause fragment to compare the column with the list of values using the given separators
		VARCHAR2/ NVARCHAR(255)		The column name
		VARCHAR2/ NVARCHAR(1000)		The expression to compare

Function name	Type Oracle or MSSQL	Parameter Type Oracle or MSSQL	Returns Oracle or MSSQL	Description
				The prefix to be added e.g. % to match the start of the column value ‘’
		VARCHAR2/ NVARCHAR(255)		The suffix to be added e.g. % to match the end of the column value ‘’
		VARCHAR2/ NVARCHAR(10)		The separator for the primary list ‘|’
		VARCHAR2/ NVARCHAR(10)		The primary comparison operator ‘OR’
		VARCHAR2/ NVARCHAR(10)		The separator for the secondary list ‘&’
		VARCHAR2/ NVARCHAR(10)		The secondary comparison operator ‘AND’
		VARCHAR2/ NVARCHAR(10)		The separator for unary operations ‘!’
		VARCHAR2/ NVARCHAR(10)		The unary operator ‘NOT’

Function name	Type Oracle or MSSQL	Parameter Type Oracle or MSSQL	Returns Oracle or MSSQL	Description
LikeString_3	Function		VARCHAR2/ NVARCHAR(512)	Gets a string for use with LIKE with special characters replaced and escaped as required
		VARCHAR2/ NVARCHAR(255)		The prefix to be added e.g. % to match the end of the column value
		VARCHAR2/ NVARCHAR(255)		The string to match
		VARCHAR2/ NVARCHAR(255)		The suffix to be added e.g. % to match the start of the column value
		PLS_INTEGER/ BIT		Trim blanks when = 1
LikeString_2	Function		VARCHAR2/ NVARCHAR(512)	Gets a string for use with LIKE with special characters replaced and escaped as required
		PLS_INTEGER/ INTEGER		Comparator value See Constants (see page 184).
		VARCHAR2/ NVARCHAR(255)		The string to match
ParticipantClassNode	Function		NUMBER	Gets the participant class node for different types of event and participant type
		NUMBER		The event major type
		NUMBER		The sender=1, or recipient=2
ParticipantIndex	Function		NUMBER	Gets a Participant Index for the given participant class for the event

Function name	Type Oracle or MSSQL	Parameter Type Oracle or MSSQL	Returns Oracle or MSSQL	Description
		NUMBER		The event UID
		DATE		The event timestamp
		NUMBER		The participant class
Participants	Function		VARCHAR2	Gets a class of participants for the specified event
		NUMBER		The event UID
		DATE		The event timestamp
		NUMBER		The participant class
Participants_2	Function		VARCHAR2	Gets a class of participants for the specified event, using eventtext1 if available
		NUMBER		The event UID
		DATE		The event timestamp
		NUMBER		The major type of the event
		NUMBER		The sender=1, or recipient=2
		VARCHAR2		The EventText1 field for the event
Percentage	Function		NUMBER	Gets a percentage value
		NUMBER		The numerator
		NUMBER		The denominator
PolicyListAll	Function/ Procedure		SYS_REFCURSOR/ Resultset	Returns the list of policy values
PolicySeverityFilter_2	Procedure		VARCHAR2/ NVARCHAR(MAX)	Builds clauses to select events by policy/severity
		VARCHAR2/ NVARCHAR(32)		The event view alias

Function name	Type Oracle or MSSQL	Parameter Type Oracle or MSSQL	Returns Oracle or MSSQL	Description
		VARCHAR2/ NVARCHAR(MAX)		The required policy class
		VARCHAR2/ NVARCHAR(MAX)		The required severity ranges (e.g.1-100)
		VARCHAR2/ NVARCHAR(MAX)		The WITH clause (output)
		VARCHAR2/ NVARCHAR(MAX)		The FROM clause (output)
		VARCHAR2/ NVARCHAR(MAX)		The WHERE clause (output)
RootGroupPath	Function		VARCHAR2	Returns the full path of the group that is the root (i.e. management group) of the given group
		NUMBER		The IDM of the group
		NUMBER		The ID of the group
SetAddressTable	Function		VARCHAR2	Inserts a list of addresses matching a given string into the specified table.
		VARCHAR2		The name of the table to contain the addresses
		VARCHAR2		The string to match the addresses
SeverityClass	Function	PLS_INTEGER/ INTEGER	VARCHAR2/ NVARCHAR(263)	Returns the style class name for the severity
SeverityList	Procedure		SYS_REFCURSOR/ Resultset	Returns the current list of severity values
SeverityText	Function		VARCHAR2/ NVARCHAR(255)	Returns the text for the severity based on defined thresholds
		PLS_INTEGER/ INTEGER		The severity value

Function name	Type Oracle or MSSQL	Parameter Type Oracle or MSSQL	Returns Oracle or MSSQL	Description
SubField	Function		VARCHAR2	Returns a particular item from a delimited list
		VARCHAR2		The delimited list
		NUMBER		The item number to be retrieved (starts at 1)
		VARCHAR2		The separator used in the list
TableFromString	Function		WGNWC_UTIL_STRINGS	Returns a table of strings from a delimited string
		VARCHAR2		The delimited list
		VARCHAR2		The separator used in the list (Defaults to ',')
Tidy	Function		VARCHAR2	Tidies a string by removing empty elements ',,', leading and trailing commas and repeated ellipsis
		VARCHAR2		The string to be tidied
UID_From_2ID	Function		NUMBER	Gets the UID from IDM and ID
		NUMBER		The IDM
		NUMBER		The ID
UserUID	Function		NUMBER	Gets a user UID for the given class of participant for the event (0 if multiple users)
		NUMBER		The event UID
		DATE		The event timestamp
		NUMBER		The required class

Constants

On Oracle, constants can be specified using *package.constant* (e.g. WgnWC_Util.COMP_EQUALS).

MSSQL does not support constants; the values need to be used instead.

Group	Constant Name	Constant Value	Description
Comparator	COMP_CONTAINS	1	String contains the specified value
	COMP_NOT_CONTAINS	2	String does not contain the specified value
	COMP_EQUALS	3	String equals the specified value
	COMP_NOT_EQUALS	4	String does not equal the specified value
	COMP_STARTS_WITH	5	String starts with the specified value
	COMP_NOT_STARTS_WITH	6	String does not start with the specified value
	COMP_ENDS_WITH	7	String ends with the specified value
	COMP_NOT_ENDS_WITH	8	String does not end with the specified value
	COMP_IS_EMPTY	9	String is empty
	COMP_IS_NOT_EMPTY	10	String is not empty

XSL Extension Functions

The functions available within XSL are very limited. These extension functions provide additional capability to ease the process of writing XSL Stylesheets. The functions are divided into separate libraries of related functions. Each library can be referenced by defining its namespace at the top of the XSL Stylesheet, for example to use the string library the stylesheet declaration should look like this:

```
<xsl:stylesheet version="1.0"    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:str="http://www.orchestria.com/string">
```

N.B. The list of functions is not exhaustive but just comprises those functions that have been found useful to date.

Library	Function Name	Parameters	Parameter Type	Description
date	nowUTC			The time now in Universal Coordinated Time (UTC) in human readable form.

Library	Function Name	Parameters	Parameter Type	Description
date	rangeUTC	dateRange	string	The input string is two WgnDate values separated with a comma. The returned string is the date range converted to UTC in human readable form.
date	rangeUTCFrom	dateRange	string	The input string is two WgnDate values separated with a comma. The returned string is the 'from' date converted to UTC in human readable form.
date	rangeUTCTo	dateRange	string	The input string is two WgnDate values separated with a comma. The returned string is the 'to' date converted to UTC in human readable form.
io	fileExists	filename	string	Returns true if the specified file exists. The filename path should be relative to the virtual directory.
io	fileList	path pattern	string string	Returns a comma-separated list of filenames matching the specified pattern in the given directory path.
math	cos	angle	degrees	Returns the cosine of the angle
math	min	val1 val2	double double	Returns the minimum of the two values
math	max	val1 val2	double double	Returns the maximum of the two values
math	sin	angle	degrees	Returns the sine of the angle
math	tick	range	double	Returns a suitable axis tick interval for the given axis range
string	compare	str1 str2	string string	Compares the two strings returning: <0 if str1 < str2 0 if str1 = str2 >0 if str1 > str2
string	decodeName	str	string	Decodes an XML name that has been encoded because it contains special characters (e.g. space)
string	elementStyle	str	string	Returns the style component of a string in the form value::style.
string	elementValue	str	string	Returns the value component of a string in the form value::style.

Library	Function Name	Parameters	Parameter Type	Description
string	encodeName	str	string	Encodes a name containing special characters in a form suitable for use as an XML name.
string	escapeString	str	string	Returns the given string escaped using URL encoding
string	indexOf	str1 Str2	string string	Returns the position of str2 within str1 or -1 if not found.
string	JSescapeString	str	string	Returns the given string escaped using backslashes.
string	replace	str strold strnew flags	string string string string	Replaces the first occurrence of strold with strnew in str. If flags is set to 'g' all occurrences are replaced
string	toLowerCase	str	string	Returns the given string converted to lower case characters
string	toUpperCase	str	string	Returns the given string converted to upper case characters

Index

I

J

L

M

O

P

R

S

T

U

V

W

X