



# I\*Reports

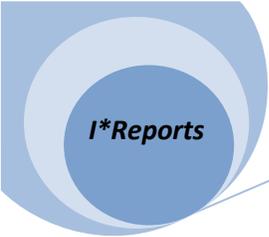
## Technical Documentation

This report provides technical documentation regarding I\*Reports, initially deployed at Southern Oregon University in Fall 2011. The audience for this report is Institutional Research staff at Southern Oregon University that are responsible for maintaining I\*Reports, the front-end Cognos Reporting System developed for access to the data warehouse for the Student Information System.

**12/12/2011**

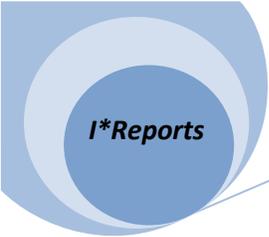
**Katie Pittman**

**Interim Project Director, Institutional Research**



## Table of Contents

OVERVIEW.....	1
OPERATING ENVIRONMENT .....	1
Development vs. Live Environment.....	1
File Organization.....	2
Interface Design / Pages .....	5
Reports (tabs) in a Multi-Page Dashboard.....	6
Channels.....	6
Editing the channel and other properties of how a report appears within a page/tab: .....	6
Editing existing reports .....	7
REPORTS .....	8
Prompts (User-Selections) .....	8
Prompt Properties .....	9
Queries and Filters in Reports.....	10
Render Variables .....	11
Reusable Components.....	12
There are also several queries in the Reusable Component report that can be copied into new reports for prompts. To copy a query from one report to another, it is easiest to copy and paste from the Query Explorer which can be accessed from the menu using View, Queries.....	12
Drill-through links.....	13
Report Views .....	14
Custom Reports.....	15
PERIODIC MAINTENANCE .....	16
At the beginning of the Academic Year .....	16
Change defaults in the following reports:.....	16
At the beginning of each term .....	17
Save prompt values for the following report views: .....	17
Change default values for prompts in the following reports .....	17
UPDATING I*Reports .....	18



## OVERVIEW

I\*Reports is a front-end interface to the Student Information System (SIS) data warehouse at Southern Oregon University. This system is intended to provide user-friendly accessibility to SIS data to support operations, decision-making, and internal reporting requirements – primarily in the Academic and Academic support units.



This documentation is intended to provide an explanation of how I\*Reports was put together. The bulk of I\*Reports was built in the July-November 2011 period by two developers who had no prior experience or formal training within the Cognos environment. A trial-by-error development approach was used, and there are undoubtedly areas that could be improved upon -- some of these areas are identified within this report using the red star as shown to the left.

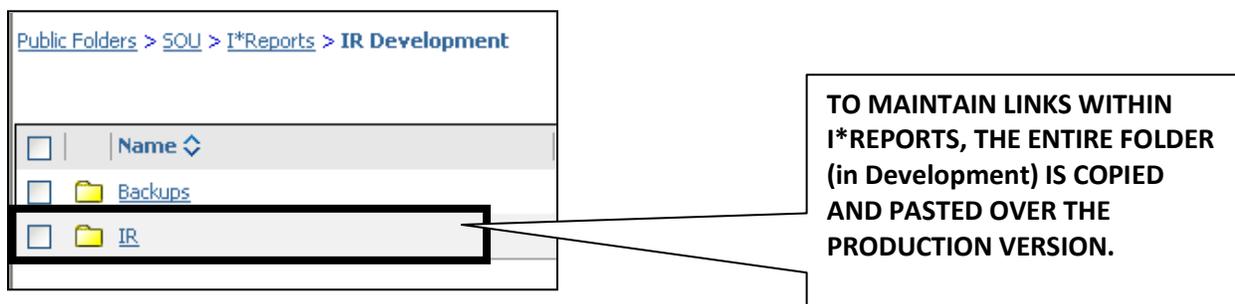
## OPERATING ENVIRONMENT

I\*Reports was built using Cognos 10.1 Report Studio. End-users access the system using Cognos Connection; therefore, they must be licensed as a “Consumer” (or higher) with access to the SIS packages.

There is a three-tiered system in place. The Institutional Research department at SOU is responsible for the development and maintenance of I\*Reports—this document is limited to that effort. The SOU Information Technology department manages the SOU SIS data warehouse and the corresponding Cognos data model (package) using Framework Manager; and the OUS fifth site manages and maintains the server and Cognos software suite that is hosted in Corvallis.

## Development vs. Live Environment

All updates to I\*Reports are made in the IR Development environment. To update the production version, the IR folder in Development is copied to Public Folders > SOU > I\*Reports for end-user access.

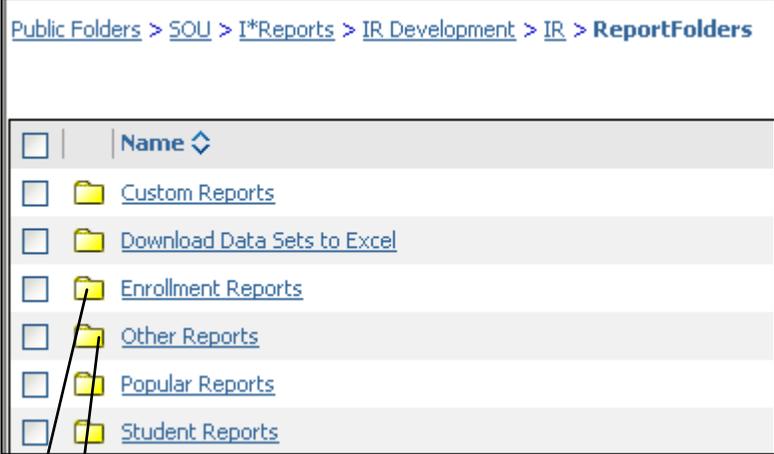
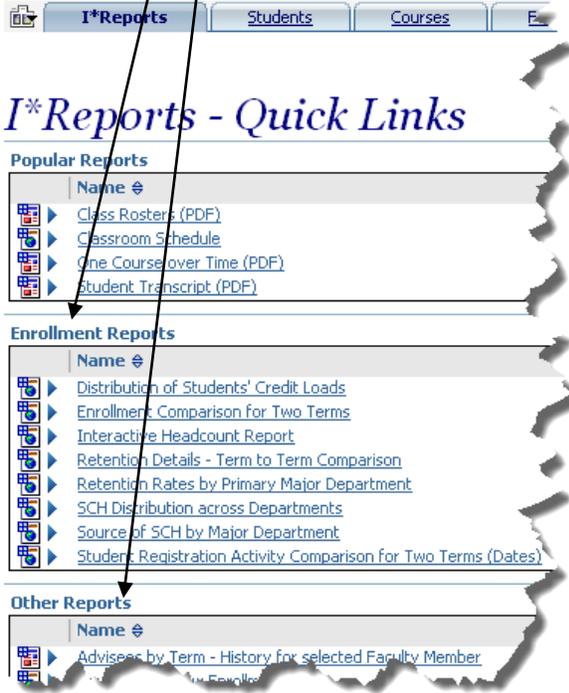


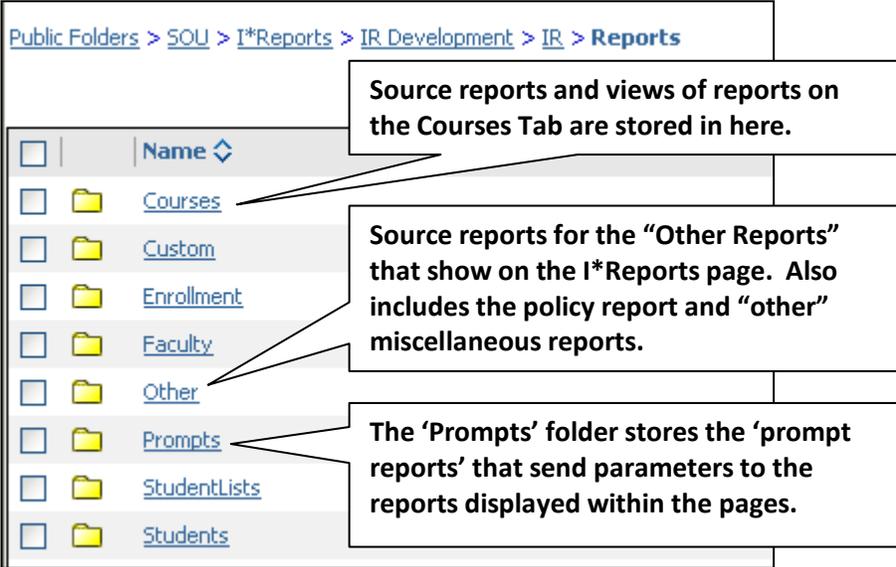
## File Organization



**I\*REPORT FILES ARE ORGANIZED INTO THESE FOUR PRIMARY FOLDERS – SEE TABLE BELOW FOR DETAILS.**

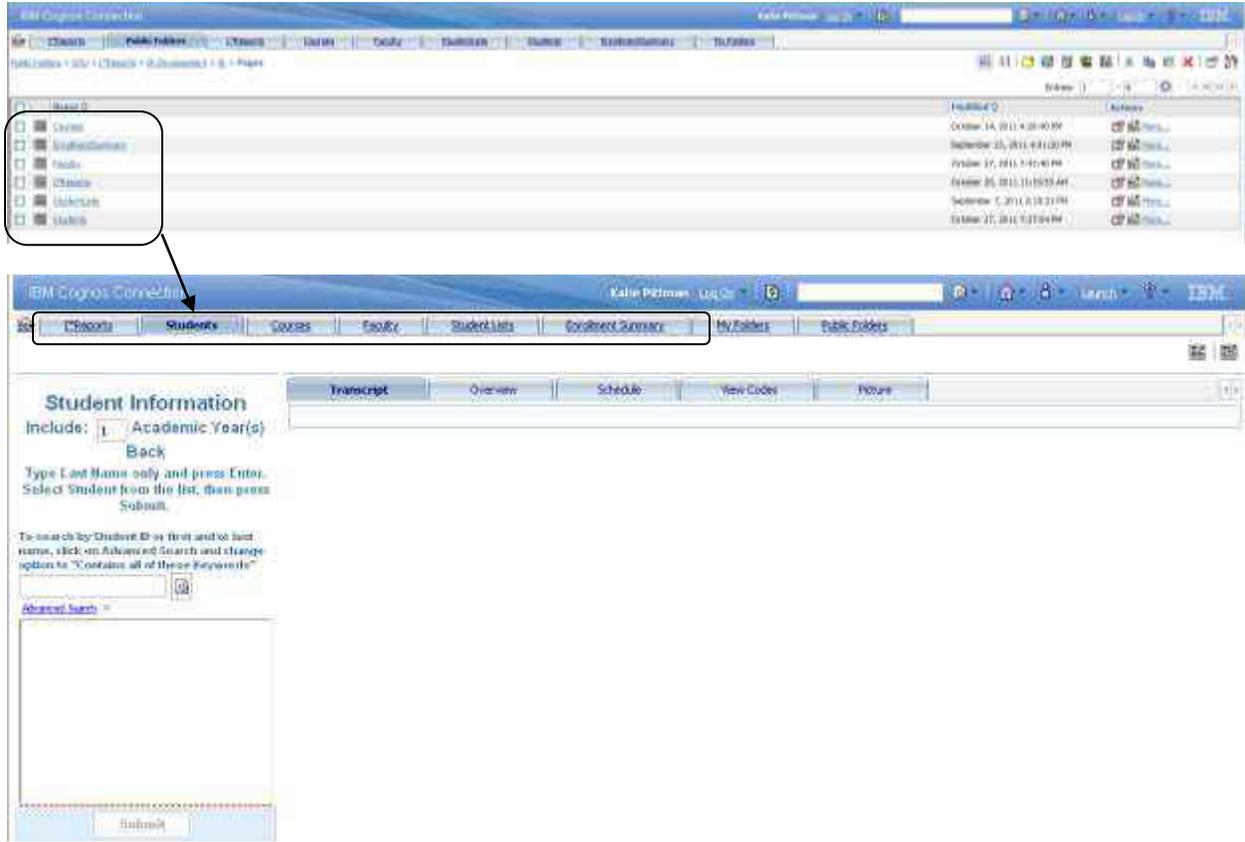
Folder	Description and Contents
Pages	<p>This folder contains the six main “pages” that make up the I*Reports interface. The first time a user accesses I*Reports, they set up their personalized view so that each page below shows up as a tab when they log in.</p> <div style="text-align: center;"> </div> <p>Direct links to pages are also available from the I*Reports (Quick Links) page. This enables users to open just that page in its own tab, or right-click to open the page in a new window.</p>

Folder	Description and Contents
<p> ReportFolders</p>	<p>These folders, shown on the I*Reports page, contain “views” of reports. Report Views are links to reports. Views can be set to run the report in a particular format and/or with prompt values pre-selected. To add a new report to a list, the report view can simply be added to the corresponding folder. Changes in a report are automatically reflected in that report’s corresponding view(s).</p>  

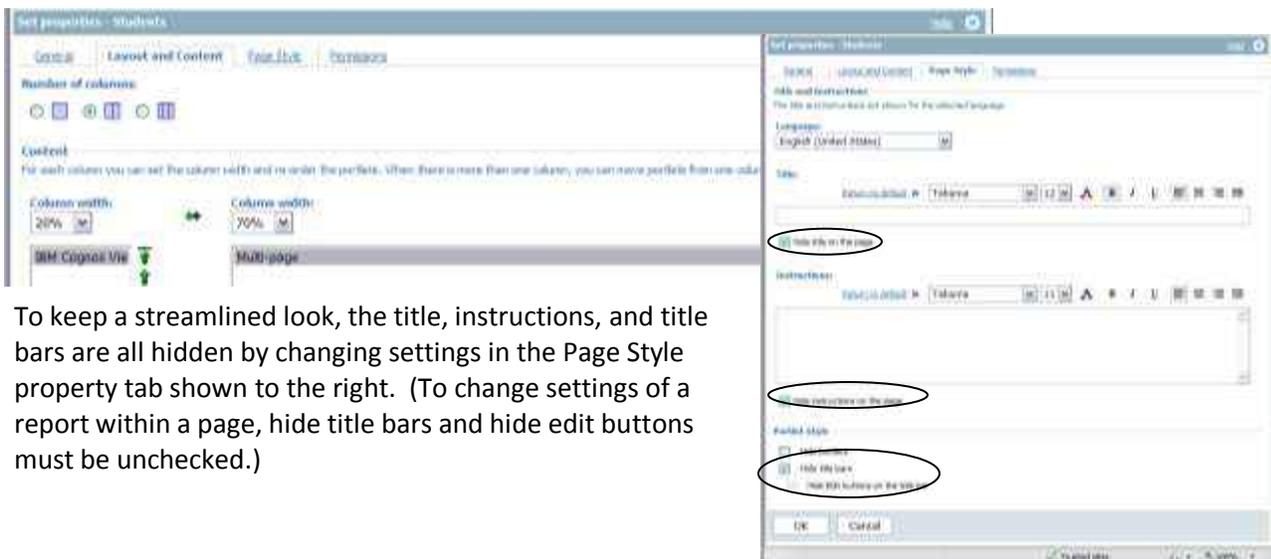
Folder	Description and Contents
<p> Reports</p>	<p>The actual reports are saved in the folders shown below. Reports are generally saved into the folder that corresponds to the page the report is displayed in or the folder that shows on the I*Reports page (Custom, Other.) If a report is shown within a tabbed page, the HTML view of the report is saved in the same folder as the report itself.</p> 
<p> Tabs</p>	<p>Pages show a prompt report on the left and “multi-page dashboards” on the right. The multi-page dashboard setting simply points to one of the folders shown below. Within each of these folders, there is a page for each tab. This is explained further in the following section.</p> 

## Interface Design / Pages

To provide a cohesive environment for users to work within, the main pages circled below have been developed. First time set-up instructions are provided to end-users to add these pages as tabs in their Cognos Connection portal.



The “Students” page shown above is composed of a Cognos Viewer portlet on the left and a multi-page dashboard on the right—as shown in the Page properties below. This is the standard approach used in I\*Reports. The end-user chooses one or more parameters from the prompt report in the left pane, then clicks on a tab to see the report. The reports do not run until the tab is selected.



To keep a streamlined look, the title, instructions, and title bars are all hidden by changing settings in the Page Style property tab shown to the right. (To change settings of a report within a page, hide title bars and hide edit buttons must be unchecked.)

## Reports (tabs) in a Multi-Page Dashboard

A multi-page dashboard points to one folder that contains a page for each tab. In the figure below, the pages correspond to the tabs in the main Student page.



To add a tab in a multi-page dashboard, add a page into the corresponding folder. To change the order of how the tabs appear, use the Change Order button circled above. These pages typically include one Cognos Viewer portlet that shows an HTML report view. (HTML report views are used in tabbed reports because if a user changes their default view to PDF, the report will open in a new window without the drill-through links available.)

## Channels

To send parameter values from a prompt page to the corresponding reports within that page, “Channels” are used. Channel settings are set in the ‘report properties’ within the Cognos Viewer properties on the page, not in the report itself.

### Editing the channel and other properties of how a report appears within a page/tab:

1. Find the page that contains the report. For prompt reports, use the page in Pages; for tabbed reports, locate the corresponding page under Tabs. Temporarily change the page properties to unhide the Title Bar: Set Properties, Page Style, Uncheck Hide Title Bars and Hide Edit Buttons.
2. Open the page and click on the Edit Button in the now visible title bar. 
3. The Entry property sets the link to the appropriate report view as shown in the example below. Reports that show up within pages are HTML views of the report – this forces the view for the report on that page in case a user has set their default report view preference to PDF.

#### Entry:

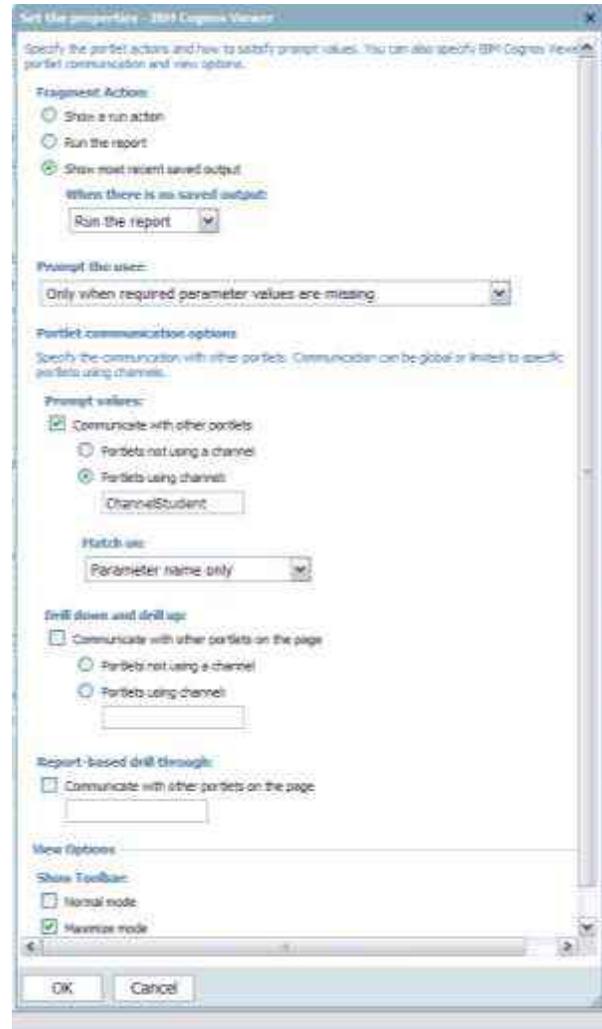


- To set the properties that determine how the report will appear within this portlet, click on Report Properties from the Entry link.

The properties shown to the right are for the reports showing in the Student tabs.

Note that the Prompt Values are set so the report will communicate with other portlets on the page. This report is using "ChannelStudent" to get the parameter ?FindStudent? from the FindStudentPrompt report showing in the left pane (portlet) of the Student Page.

To test changes made to a report or report properties, refresh the page AND change the prompt selection. If a new parameter is added to the prompt page, the link to the report needs to be broken (Entry/Clear) then added again.



### Editing existing reports

All reports were created using Report Studio. All edits to existing reports should be done in development environment). This will automatically update any corresponding report views.



**Report views link to Source report, therefore they are automatically updated when the source report has been edited.**

Changes will not be immediately evident if a report appears within a page. The page needs to be refreshed (F5) and the report needs to be run with at least one selection parameter changed. The updated report should then appear. If a report is accessed through a hyperlink (not embedded in a page), then the new version will show when the report is run.

## REPORTS

The focus of this section is to explain some of the nuances on how I\*Reports has been put together and how to make changes within this environment. Click the “?” in the toolbar within Report Studio to access the Report Studio User Guide. There is also a wealth of information available online – including tutorials posted in You-tube and many user-group bulletin boards.



Reports are directly edited within the Development environment. If a report is copied, edited, then copied over the original (or if a backup copy of a report is restored over a new copy), this will break the connection to associated report views and corresponding portal links. New report views will need to be created (see Creating Report Views below) and/or the portlet entry properties will need to be cleared and re-established.

To test report changes for reports that are inside pages, press Refresh (F5) AND change a prompt value to see the revised report.

### Prompts (User-Selections)

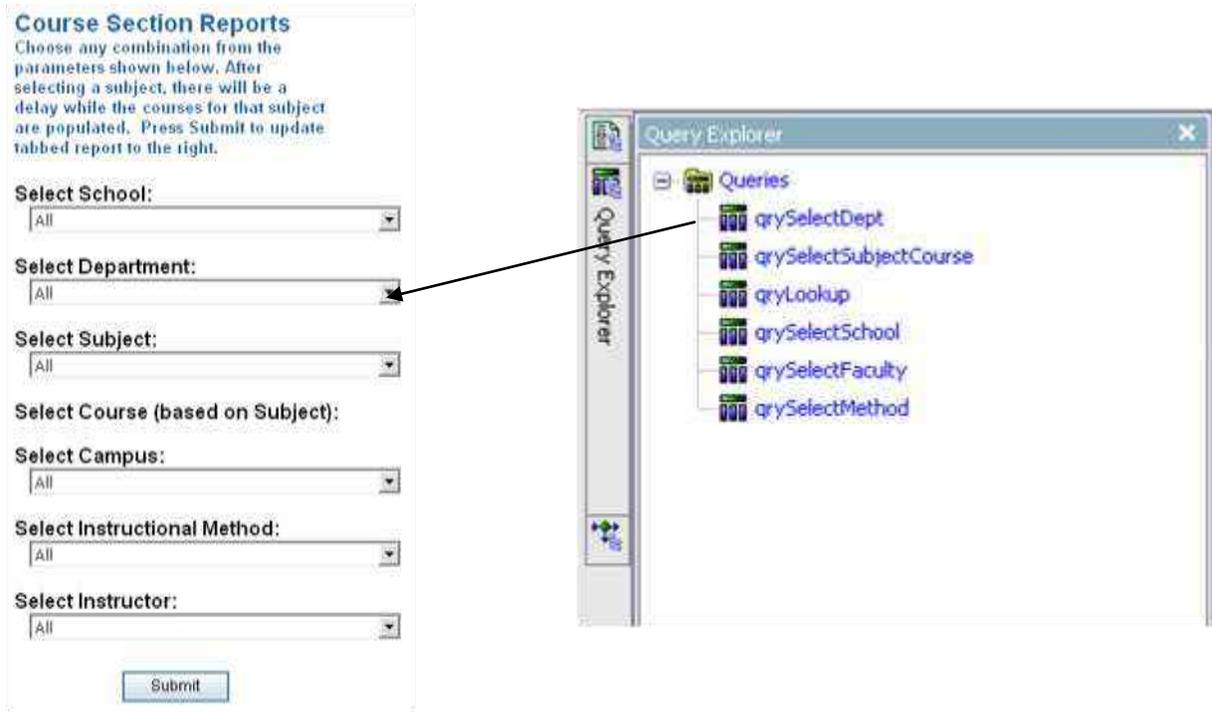
User selections for reports are done either through the Prompt Reports (left-side portlets within the main I\*Report pages) or within the report itself. I\*Reports uses several different types of prompts, depending on the context. Within report studio, a prompt can be added to a report from the Toolbox as shown below.

 A screenshot of the Report Studio Toolbox, which lists various prompt types. Callouts provide detailed information about several of these prompts:
 

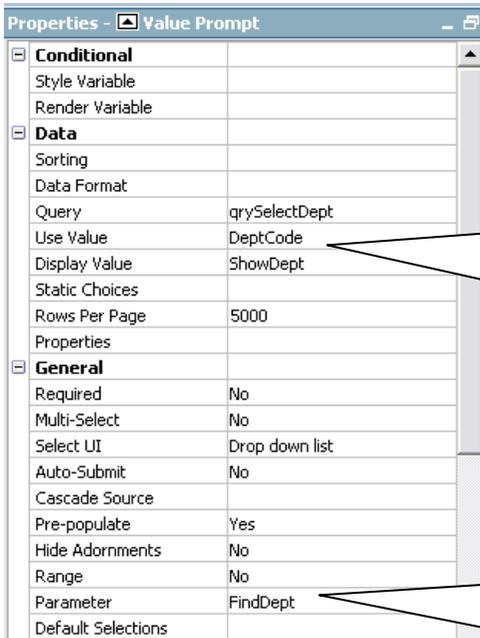
- Text Box Prompt:** Accepts typed values.
- Value Prompt:** Drop-down list, list box, or radio button group based on values in a query or a list of static choices. Limit of 5000 rows.
- Select & Search Prompt:** User types in 'keyword' then presses enter to choose from resulting values. Used in Student Search.
- Date & Time Prompt, Date Prompt, Time Prompt:** Dates can also be entered using a text box prompt, which take less space and can be easier/faster for the end-user.
- Prompt Button:** Button "types" used: Cancel, Next, Finish. To change text of a prompt button, drop "Text Item" from Toolbox onto button.

### Prompt Properties

Queries are built to generate the values that appear in Value prompts. As shown below, the value prompts in the PromptFindCourses report are based on the queries stored in that same report.



Prompts for the Select Department Value Prompt are shown below:



**Use Value:** field that is assigned to Parameter

**Display Value:** value shown to the user. In this case, the following query expression was created to concatenate the dept code to the dept description.

```
[All Students].[Course Schedule].[Department Code] || ' - ' || [All Students].[Course Schedule].[Department Description]
```

The user's selection is saved to the parameter named FindDept. This parameter is then available within a report or to reports sharing a channel.

The syntax for including this parameter in the filter for a course report is:

```
[All Students].[Course Schedule].[Department Code]=?FindDept?
```

Each prompt runs a query against the database, so these prompt reports (and prompt pages) within a report can be slow to run before appearing on the screen. Some queries include hard-coded values. For example, qrySelectFaculty selects where [All Students].[Course Schedule].[Schedule Term Code]>='200701'. There is a calculated field in qryLookup called FiveYearsBack that could be used rather than hard-coding the '200701' but, in this case, the combined performance amongst these queries degraded significantly, so hard-coding was used.



Caching the reports was attempted to speed up prompt reports (Advanced Report Properties), and this worked well. However, some users received errors when the cache ran out, so this approach was reversed. This is an area worth pursuing in the future. If the reports were cached for general use, then some of the hard-coding that was used could be eliminated as the report itself would only need to be run once a month or so.



## Queries and Filters in Reports

If a report includes prompts, there will generally be a query for each prompt, although cascading prompts (a prompt that is populated based on the value of another selection) may be based upon a shared query. These queries use the naming convention qrySelectname. Filters applied to these queries most often limit the prompt dataset to current or recently used values. In addition to better performance, this eliminates old values from showing up in the drop-down list.

Additionally, there will be at least one primary query to select the columns that appear within the report (often uses the default Query1 name.) Multiple filters are typically applied to the main query of a report. As an example, the filters for the Course Schedule Report are shown below.

```
[Schedule Term Code]>=[qryLookup].[Current Term]
*[All Students].[Course Schedule].[College Code]=?FindSchool?
*[All Students].[Course Schedule].[Department Code]=?FindDept?
*[All Students].[Course Schedule].[Subject Code]=?FindSubject?
*[All Students].[Course Schedule].[Course Number]=?FindCourseNum?
*[All Students].[Course Schedule].[Schedule Term Code] = ?FindTerm?
*[All Students].[Course Schedule].[Instructor ID Primary]=?FindFaculty?
*[All Students].[Course Schedule].[Campus Description]=?FindCampus?
*[All Students].[Course Schedule].[Instructional Method Code]=?FindMethod?
[Subject Code] not in ('CC', 'NUR', 'RCC', 'TRAN')
[CRN]<>'TRAN'
```

Filters with an \* above are Optional filters, which is designated by the Usage properties for the filter. Optional filters are ignored if there is no value available for the parameter. The Application property determines if the filter is applied before or after the data is aggregated.

Properties - Detail Filter	
<b>Data</b>	
Definition	[All Students].[Course Schedule].[Department Code]=?FindDept?
<b>General</b>	
Usage	Optional
Application	Before Auto Aggregation

## Render Variables

A render variable determines whether an object is displayed or not. I\*Reports makes extensive use of Boolean Variables to control whether objects are shown. To apply a render variable to an object, add the variable name in the Render Variable property for that object.

Examples of variables used as Render Variables (View, Variables in Report Studio menu)	Assigned as Render Variables in the properties of an Object.																												
<table border="1"> <thead> <tr> <th colspan="2">Properties - Variable</th> </tr> </thead> <tbody> <tr> <td colspan="2"><b>General</b></td> </tr> <tr> <td>Type</td> <td>Boolean</td> </tr> <tr> <td>Report Expression</td> <td>ReportOutput() = 'PDF'</td> </tr> <tr> <td colspan="2"><b>Miscellaneous</b></td> </tr> <tr> <td>Name</td> <td>PDF</td> </tr> </tbody> </table> <p><b>The PDF variable is used in all reports that can be printed to render the footer table that shows page number.</b></p>	Properties - Variable		<b>General</b>		Type	Boolean	Report Expression	ReportOutput() = 'PDF'	<b>Miscellaneous</b>		Name	PDF	<table border="1"> <thead> <tr> <th colspan="2">Properties - Table</th> </tr> </thead> <tbody> <tr> <td colspan="2"><b>Conditional</b></td> </tr> <tr> <td>Conditional Styles</td> <td></td> </tr> <tr> <td>Style Variable</td> <td></td> </tr> <tr> <td>Render Variable</td> <td>PDF</td> </tr> </tbody> </table>	Properties - Table		<b>Conditional</b>		Conditional Styles		Style Variable		Render Variable	PDF						
Properties - Variable																													
<b>General</b>																													
Type	Boolean																												
Report Expression	ReportOutput() = 'PDF'																												
<b>Miscellaneous</b>																													
Name	PDF																												
Properties - Table																													
<b>Conditional</b>																													
Conditional Styles																													
Style Variable																													
Render Variable	PDF																												
<table border="1"> <thead> <tr> <th colspan="2">Properties - Variable</th> </tr> </thead> <tbody> <tr> <td colspan="2"><b>General</b></td> </tr> <tr> <td>Type</td> <td>Boolean</td> </tr> <tr> <td>Report Expression</td> <td>ParamValue('FindStudent') is not null</td> </tr> <tr> <td colspan="2"><b>Miscellaneous</b></td> </tr> <tr> <td>Name</td> <td>PIDMAvailable</td> </tr> </tbody> </table>	Properties - Variable		<b>General</b>		Type	Boolean	Report Expression	ParamValue('FindStudent') is not null	<b>Miscellaneous</b>		Name	PIDMAvailable	<table border="1"> <thead> <tr> <th colspan="2">Properties - Page</th> </tr> </thead> <tbody> <tr> <td colspan="2"><b>Conditional</b></td> </tr> <tr> <td>Conditional Styles</td> <td></td> </tr> <tr> <td>Style Variable</td> <td></td> </tr> <tr> <td>Render Variable</td> <td>PIDMAvailable</td> </tr> </tbody> </table> <p><b>The report page will not be rendered unless PIDMAvailable is Yes.</b></p>	Properties - Page		<b>Conditional</b>		Conditional Styles		Style Variable		Render Variable	PIDMAvailable						
Properties - Variable																													
<b>General</b>																													
Type	Boolean																												
Report Expression	ParamValue('FindStudent') is not null																												
<b>Miscellaneous</b>																													
Name	PIDMAvailable																												
Properties - Page																													
<b>Conditional</b>																													
Conditional Styles																													
Style Variable																													
Render Variable	PIDMAvailable																												
<table border="1"> <thead> <tr> <th colspan="2">Properties - Variable</th> </tr> </thead> <tbody> <tr> <td colspan="2"><b>General</b></td> </tr> <tr> <td>Type</td> <td>Boolean</td> </tr> <tr> <td>Report Expression</td> <td>[ClassInfo].[Student Level Code]= 'UG' and [ClassInfo].[Under100Total] &gt;0</td> </tr> <tr> <td colspan="2"><b>Miscellaneous</b></td> </tr> <tr> <td>Name</td> <td>HoursUnder100</td> </tr> </tbody> </table> <p><b>This variable is used to determine whether a warning message shows up at the bottom of the transcript. A "singleton" is a data container that can be used to insert data from a query into a table.</b></p>	Properties - Variable		<b>General</b>		Type	Boolean	Report Expression	[ClassInfo].[Student Level Code]= 'UG' and [ClassInfo].[Under100Total] >0	<b>Miscellaneous</b>		Name	HoursUnder100	<table border="1"> <thead> <tr> <th colspan="2">Properties - Singleton</th> </tr> </thead> <tbody> <tr> <td colspan="2"><b>Conditional</b></td> </tr> <tr> <td>Render Variable</td> <td>HoursUnder100</td> </tr> <tr> <td colspan="2"><b>Data</b></td> </tr> <tr> <td>Query</td> <td>ClassInfo</td> </tr> <tr> <td>Properties</td> <td></td> </tr> <tr> <td colspan="2"><b>Miscellaneous</b></td> </tr> <tr> <td>Name</td> <td>Singleton28</td> </tr> </tbody> </table>	Properties - Singleton		<b>Conditional</b>		Render Variable	HoursUnder100	<b>Data</b>		Query	ClassInfo	Properties		<b>Miscellaneous</b>		Name	Singleton28
Properties - Variable																													
<b>General</b>																													
Type	Boolean																												
Report Expression	[ClassInfo].[Student Level Code]= 'UG' and [ClassInfo].[Under100Total] >0																												
<b>Miscellaneous</b>																													
Name	HoursUnder100																												
Properties - Singleton																													
<b>Conditional</b>																													
Render Variable	HoursUnder100																												
<b>Data</b>																													
Query	ClassInfo																												
Properties																													
<b>Miscellaneous</b>																													
Name	Singleton28																												

Additional examples of the use of Render Variables in I\*Reports:

### List Columns

- Columns containing hyperlinked fields are rendered only if the "HTML" variable is set to yes.
- Some columns (such as attributes in Student List) only appear if a student has selected an attribute

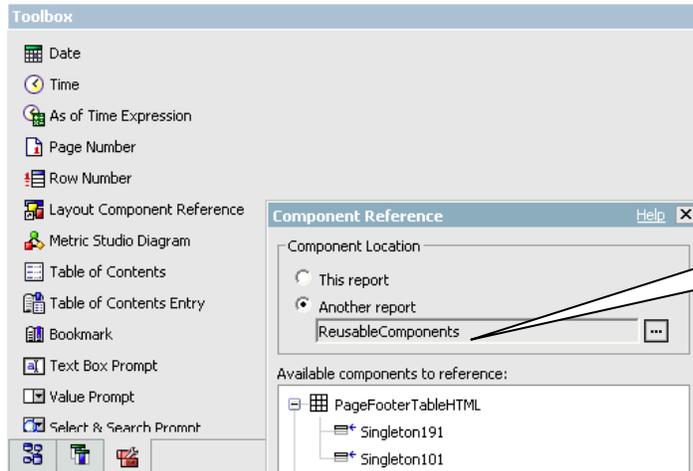
### Lists

- Some reports have multiple "lists" where rendering is determined by the ReportOutput() form. (See first example in table above for PDF variable.) An "HTML" version of a list will show hyperlinked columns and use multiple vertical blocks within a column to limit the width of the report on the screen. An Excel list for the same report, on the other hand, will present all the fields in separate columns and eliminate subtotals and totals to allow for easier manipulation of the data in Excel. This approach was established late in the game and there may be additional opportunities to adjust some of the reports that may be exported to Excel with grouping, sections, and/or subtotals intact.



## Reusable Components

Each report contains the same footer 'table' that is stored in the 'Reusable Components' report. The Toolbox option **Layout Component Reference** is used to point to this component.

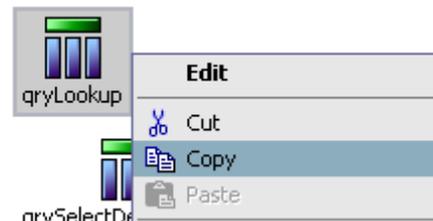


To add the standard footer to a new report, drag the **Layout Component Reference** object into the Report footer and browse to find the Report named **Reusable Components** and select **PageFooterTableHTML**.



The page numbering component within this footer is attached to a PDF render variable (only shows on PDF versions of reports) and the **<Activity Date>** comes from a query called **QryLookup** in the same report. Because of this, each report must have a PDF variable AND a copy of the **qryLookup** query.

There are also several queries in the Reusable Component report that can be copied into new reports for prompts. To copy a query from one report to another, it is easiest to copy and paste from the Query Explorer which can be accessed from the menu under View, Queries.



Once the query has been copied into the new report, it may need to be modified to match the context of the report. For example, there is a calculated field in **qryLookup** that is named **FiveYearsBack**. For the 10 year graphs, this was modified to a calculated field called **TenYearsBack** that can be used to filter the info within a report, such as:

```
[All Students].[Degrees].[Graduation Term Code]>=[qryLookup].[TenYearsBack]
```

## Drill-through links

Report-based drill-through links are used throughout I\*Reports to open the same report in a different format or a new report. The drill-through definition is used to control the report that will be opened, the format of the report, and the data item(s) or parameter values to be passed.

Drill-through links are used for the PDF and EXCEL links in the upper right hand corner of most reports. To create a drill-through hyperlink that passes parameters, type the text to use for the link, right-click on the text, and choose Drill-Through Definitions to get to the drill-through definition page.

To edit/view the data items or parameters passed to a report click on the edit button.

For the Excel link on the student list, all parameter values are passed as shown below.

Target Report: Bookmark Label

Report: StudentList

Action: View most recent report

Format: Excel 2007

Open in new window

Parameters:

- ClassGroup
- FindConc
- FindDept
- FindMajor
- SortOrder

Name	Type	Required	Multi-select	Method	Value	Property to Pass
ClassGroup	String	<input checked="" type="checkbox"/>		Pass parameter value	ClassGroup	(Default)
FindAdvisor	String			Pass parameter value	FindAdvisor	(Default)
FindAttribute	String			Pass parameter value	FindAttribute	(Default)
FindCohort	String			Pass parameter value	FindCohort	(Default)
FindConc	String			Pass parameter value	FindConc	(Default)

Drill-through links can also be used to pull up a report specific to a data value in a list. For example, the student name in the student list and rosters can be clicked on to open the StudentTranscript report for the selected student. The Drill-through definition for passing data values from a list is a property for the **List Column Body**.

In this case, the PIDM from the student list is used as the data item value to pass to the StudentTranscript report. The PIDM field must be in the query source for the list, but does not need to appear on the list itself. NOTE: PIDM is set to automatically aggregate (Count), therefore the Aggregate Function property of that field in the query must be set to None (or the value passed will be a "1").

Target Report: Bookmark Label

Report: StudentTranscript

Action: Run the report using dynamic filtering

Format: (Default)

Open in new window

Parameters:

- FindStudent

Display prompt pages: Only when required parameter values are miss

Name	Type	Required	Multi-select	Method	Value	Property to Pass
FindStudent	Number	<input checked="" type="checkbox"/>		Pass data item value	PIDM	(Default)
YearsBack	String	<input checked="" type="checkbox"/>		(Default)		(Default)

## Report Views

As mentioned, HTML report views are created for reports that appear in the multi-page dashboard. Report views are also used to add links to folders on the quick links page, (For example, there is a view of the StudentTranscript report labeled Student Transcript (PDF) in the Popular Reports folder.)

To create a View of a report, go to the folder where the source report is located and click on the Create a Report View button as circled to the right:



Specify the Name and Location of the Report view, then click Finish. After this, go to the Report View and change the properties of the view as needed. Properties in the Report View that can be set include the report output format, description, user permissions, and default prompt values. These properties are set for the view only and will not affect the original report properties. As mentioned, if a report is overwritten, then the link to views will be broken. If a report is renamed, the link will remain intact.



End-Users have the capability to create their own report views and save them to their MyFolder location. From there, they could change the view properties to pre-select the prompts they would use. With that done, they could simply click on the report view to see the updated version with their prompts pre-selected. This would streamline the use of reports that end-users run on a regular basis.

## Custom Reports

In some cases, custom reports have been developed for specific end-users. These reports are saved in:  
**Public Folders > SOU > I\*Reports > IR Development > IR > Reports > Custom**

To 'distribute' the report to selected end-users, a report **view** should be created in the Custom Report folder that appears on the I\*Reports Page:

**Public Folders > SOU > I\*Reports > IR Development > IR > ReportFolders > Custom Reports**

To limit access to a report to a specific user, go to the Permissions tab in the Set Properties page of the report view. Check the Override the access permissions option.

To add a user, click on Add, Check Show Users in the list, choose SOULDAP, Employees, and a list of employee names and login addresses will be shown. Click Search in the upper right hand corner of the screen and type any part of the users name or login address. Choose the user(s) from the Results: and add to the Selected entries.

Give the new selected user "Read, Execute, and Traverse" privileges to the report.

Remove access to any remaining users (labeled as unavailable) to take this off of their menu.

To see/edit the properties for each user, click in the check box and Grant access to the following actions:

Set properties - McVayDyche-OnLineStates

General Report view **Permissions**

Specify access permissions for this entry. By default, an entry acquires its access permissions from a parent. You can override those permissions with the permissions set explicitly for this entry.

Override the access permissions acquired from the parent entry

<input type="checkbox"/>	Name	Permissions	<input type="checkbox"/> Grant	<input type="checkbox"/> Deny
<input checked="" type="checkbox"/>	...>Jennifer McVay-Dyche	Read, Write, Execute, Set Policy, Traverse	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	...>Katie Pittman	Read, Write, Execute, Set Policy, Traverse	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	...>Shane Hunter	Read, Write, Execute, Set Policy, Traverse	<input type="checkbox"/>	<input type="checkbox"/>

Read  Deny   
 Write  Deny   
 Execute  Deny   
 Set Policy  Deny   
 Traverse  Deny

**Option**

Select this option if you want to override the existing access permissions of all child entries.

Delete the access permissions of all child entries

## PERIODIC MAINTENANCE

To speed up the running of some of the reports, default prompt values have been hard-coded for selected reports -- these values need to be reset at the beginning of a new term or the beginning of the academic year. *(Note: in most cases, this can be done programmatically, but the performance delay is not acceptable. It is hoped that over time, an alternative approach can be determined.)*



Whenever possible, prompt selections are set in a “view” of the report. This has been done for reports where all remaining prompt values are optional. Saving prompt values for a report view cannot be done if there are required filters on the other fields—in this case, the report itself can be edited and the default value property can be set on the prompt. Example: in StudentSchedule, the current term is set as a default in the value prompt of the report page because PIDM is required.

### To save prompt values in Report VIEWS,

- Find the report view in the appropriate folder
- Click on the Set Properties icon
- Go to Report View Tab and Set/Edit... the Prompt Values
- Enter values for prompts, then Submit
- Uncheck the Prompt for values box
- Click on OK to save

### To set default values for the prompts within a report,

- Find the report in the appropriate folder
- Click on the Edit Report icon
- Click on the prompt to set
- Change the Default Selection Prompt property



## At the beginning of the Academic Year

### Change defaults in the following reports:

#### Public Folders > SOU > I\*Reports > IR Development > IR > ReportFolders > Enrollment Reports

- Retention Details – Term to Term Comparison
- Find Term One: Fall term of prior academic year
- Find Term Two: Fall term of current academic year

## At the beginning of each term

### *Save prompt values for the following report views:*

#### **Public Folders > SOU > I\*Reports > IR Development > IR > Reports > Courses**

CourseHistoryGroup - HTML

From Term: Fall term of Prior Academic Year

Through Term: Current Term

#### **Public Folders > SOU > I\*Reports > IR Development > IR > Reports > Enrollment**

StudentDemographics-HTML

Registration Term Code: Current Term

Find Level: All Admitted

### *Change default values for prompts in the following reports*

#### **Public Folders > SOU > I\*Reports > IR Development > IR > Reports > Other**

ClassRosters

Change TermCode prompt in PromptPage1 to current term

#### **Public Folders > SOU > I\*Reports > IR Development > IR > Reports > Students**

StudentSchedule

Term Code Prompt: Current Term

#### **Public Folders > SOU > I\*Reports > IR Development > IR > Reports > StudentLists**

StudentList

Change TermCode prompt on Report Page 1 to current term

*(setting prompts in StudentList-HTML complicated due to sort and class standing options)*  
*(FYI: StudentList-HTML view set to not prompt)*

StudentListMinors

Change TermCode prompt on Report Page 1 to current term (view set to prompt: no)

#### **Public Folders > SOU > I\*Reports > IR Development > IR > Reports > Faculty**

FacAdvisees

Change TermCode prompt on Report Page 1 to current term (view set to prompt: no)

FacAdviseesNotRegistered

Change TermCode prompt on Prompt Page to next term (view set to prompt: no)

FacStudentsNotRegistered

Change Detail Filter lines in "StudentList" query to select Current Term:

[Enrollment Term Code]='201101' (use term code)

[All Students].[Course Registration and Grades].[Registration Term Code]='201101'

After all prompts and defaults have been changed, update the production version as described on the following page.

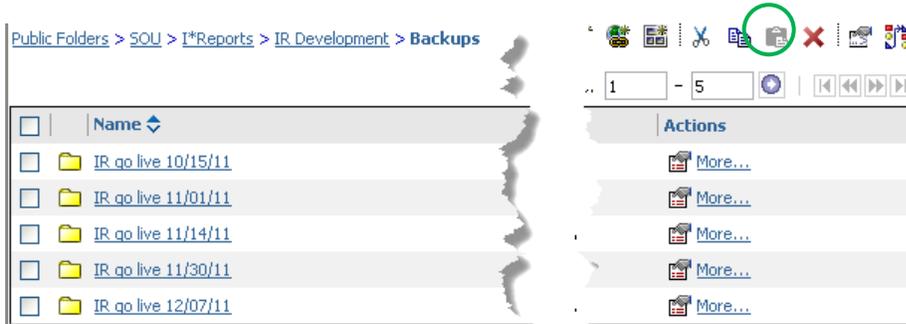
## UPDATING I\*Reports

To update I\*Reports, the following process has been established.

1. Copy the IR folder from Public Folders>SOU>I\*Reports>IR Development into Clipboard



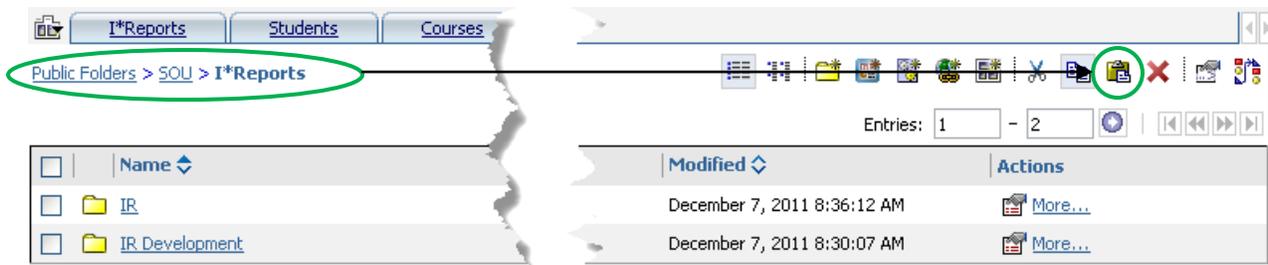
2. Paste Copy as Backup: Open Backups Folder, Paste,



3. Rename backup copy to "IR go live mm/dd/yy" as shown above. To rename, go to Properties and change the name.

4. Again, Copy the IR folder from Public Folders>SOU>I\*Reports>IR Development

5. Go Live with new version: Go to the I\*Reports folder and paste over the Live Version:



4. The following warning will appear. Click on Yes to finish.

