Build Information - Build 112

From InfoBurst Support Wiki

Contents

Release Date

March 24, 2010

User Interface

Version

2.7.709.1

Document Preview Icons

Document preview icons now available for Business Objects documents.

Document Preview Icon

Excel Post Processing

Excel post processing functions now available for formatting Excel generated from Business Objects reports.

Excel Post Processing

Platform

Version

2009.14.111

Platform Features

This build adds the following core features to the platform:

- Document thumbnail images
- Document meta-data properties added
- XDC Cache Query
- Burst XDC and deliver self contained .swf (Offline)
- New Action “StartBurst”
- REST interface now supports get/post and xml/json
- New UserCache speeds up certain tasks

The following other features & fixes are also available

- XDC with 2 x parts using same WI Source generated runtime error, Fixed
- Email with Zip could result in multiple .zip extensions, Fixed
- CSV output for WI/DI docs with multiple Data Providers caused corrupt header, Fixed
- Added [$DP] support for Email delivery
- Crystal XIR2/XI3.x based on Universe would require DBLogon, Fixed
- XDC Query with decimal values was not formatted correctly, Fixed.
- Scheduling Instance of WebI Document with InList prompts could fail, Fixed.
- MultiColumnList Object row size increased from 256 to 1024 characters

Document Thumbnails & Preview

A new option 'CreatePreview' has been added to the Platform and Document Objects and an option 'ViewPreview' has been added to the User Object.

The default for all is True which means that the system will create preview images and each User will see them.

A thumbnail image will be created or updated when the Document is refreshed in a Burst and therefore as time goes on and Documents get processed the more thumbnails will become available (plus those Documents that do not have a thumbnail can easily be seen as not being processed!).

Thumbnails are created based on the platform logon so depending on how the BO security has been set and how each User connects to BO there could be one or more thumbnails created per Document. This means that a User who views a Thumbnail can only see the data that their BO credentials allow.

By default, the system will create a new Thumbnail and keep it for 30 days and then on the next Refresh a new one will be created.

The Document has the following new properties that control thumbnails:

- **CreatePreview** (Bool) - Set to False to disable Thumbs for this Document
- **PreviewRefresh** (Number) - Defaults to 30. Refresh every # Days
- **PreviewReport** (String) - The name of the Report-Tab to use for the Preview
- **PreviewPage** (Number) - The page # within the tab to use for the Preview
- **NewPreviewOnNextRefresh** (Bool) - Set to True to trigger a refresh of the Preview

Each thumbnail is created in 3 sizes, 128x128, 256x256 and 512x512 to allow for a UI that may wish to display a larger size.

The URI to retrieve the thumbnail image will be returned in IB_Object when calling functions such as GetObjectsInFolder and SearchForObjects. The URI will point to the REST service and therefore the call must be authenticated using either the current Soap token (passed as the User) or a valid IB user/password.

Rest Service URIs

Get/Preview/**id**
Get/Preview/**id**?**sz**=128/256/512
Get/Preview/**id**?**sz**=768&pg=2
Get/Preview/**id**?**sz**=768&rpt=Report&pg=2

id is a unique GUID that identifies the thumbnail.

The default size is 128 but you can request a larger size by using the ?sz parameter.

If you specify a size that is not pre-cached (128/256/512) a dynamic image will be created (but not cached).

You can also specify a page # to get a view of a different page or a report/page# to display a specific page from a report-tab.

The call will return a PNG Image (image/png).

The call will check that the authenticated user matches the preview image's security and so in the event that another user tries to view a preview image and supplied different credentials the image will not be displayed.

Script Commands

new/modify platform .. preview="false"

new/modify user .. preview="false"

New options for "modify document" :

- **preview**="false"
- **previewreport**="tab-name"

- **previewpage**=”1”
- **previewrefresh**=”30”
- **previewrefreshnext**=”true”

The current status of Preview can be seen using the display commands.

Document meta-data properties added

The following properties have been added to the Document Object :

- **LastRefreshed** (Date) - Date of last Refresh
- **RefreshCount** (Number) - # times this Document has been refreshed
- **LastRefreshType** (ObjectType) - The Object (Burst or XDC) that processed the Doc
- **LastRefreshObjectID** (Number) - The ID of the Burst/XDC
- **PreviewTotalPages** (Number) - The total # pages in the document preview
- **PreviewTabs** (Array) - A breakdown by Tab showing the start/end page

XDC Cache Query

IB 2.6 introduced ‘Intelligent Cache’ that allows for dashboard designers to pull a subset of data from a pre-built cache and has proved to be a very effective way of building complex drill-down style dashboards with blazing performance.

IB 2009 takes this approach and evolves the solution to provide the ultimate in flexibility and performance.

It is now possible to build one or more Queries that can be executed against any XDS in a Cache and each Query can use the full power of the SQL language to do almost anything. Placeholders are supported and so a Query can have one or more parameters that you supply when executing the Query.

In order to assist in writing the Query, each XDS has been extended to provide a list of all Column Names and Types.

The implementation of Cache Query uses a local high speed SQLite database per XDC and therefore the runtime performance is excellent. Metrics have been built in so for each Query the # Rows and the Query Execution time plus the Xml rendering time will be available.

Internal tests using a 100k row cache and a query with a WHERE using 4 columns showed a response time under 1 second (equivalent “intelligent cache” call using same cache was 6 seconds).

In an effort to handle a large cache effectively, any XDS that generates more than 2048 rows can ONLY be queried as it makes no sense to deliver such a large Xml stream(Xcelsius cannot handle more than this). If a call is made to Get/xdc to get the entire pre-built cache any XDS that exceeded the row limit will be returned as an empty variable. The “XDSReady” variable has been updated to include an additional row per XDS showing the Name and #Rows (so it is easy to see why a variable may be blank).

When the XDC is refreshed, the CacheQuery data will be populated and any indexes will be created. You should always refresh the Cache if you add a new XDS or if you set a new Query Index.

If the XDC is a multi-value Cache (e.g. by “State”) an extra column will be created call **CacheKey** that you can use in queries to filter by the desired value.

A Query is based on a single XDS within the Cache but it works on ANY Source so the XDS can be based on WebI, Query or Excel.

Again, to ensure the results of a Query are valid a limit has been placed on the #Rows and so if a Query generates more than 2048 rows an XDSError will be generated that shows the actual # rows.

Using the IBShell we can easily add a Query to an existing XDC:

add xcq xdc=”MyCache” name=”GetSales” sql=”select * from AllSales where SalesPrice>=@Price order by SalesPrice”

The SQL selects from a “table” called “AllSales” and this is important as the FROM must match an existing XDS so in this example the Cache “MyCache” has an XDS called “AllSales”.

SOAP APIs

```
Function CreateObject_XmlDataQuery(ByVal Token As String, ByVal ObjectInfo As IB_XmlDataCacheQuery)
As Boolean
```

```
Function GetObject_XmlDataQuery(ByVal Token As String, ByVal XDCID As Integer, ByVal QueryName As
String) As IB_XmlDataCacheQuery
```

```
Function ModifyObject_XmlDataQuery(ByVal Token As String, ByVal ObjectInfo As IB_XmlDataCacheQuery)
As Boolean
```

```
Function DeleteObject_XmlDataQuery(ByVal Token As String, ByVal ObjectInfo As IB_XmlDataCacheQuery)
As Boolean
```

```
Function XDC_GetDataSourceColumns(ByVal Token As String, ByVal ObjectInfo As IB_XmlDataCacheSource)
As IEnumerable(Of IB_XDCQryColumn)
```

```
Class IB_XmlDataCacheQuery
 Public Name As String
 Public XDCID As Integer
 Public ID As Integer
 Public SQL As String
 Public IndexColumn As String
 Public ReturnColumnNames As Boolean
```

```
Class IB_XDCQryColumn
 Public ColumnName As String
 Public ColumnType As String
```

Script Commands

```
add xcq xdc=”C1” name=”Query1” sql=”select * from XDSName .. “
modify xcq xdc=”C1” name=”Query1 sql=”...”
delete xcq xdc=”C1” name=”Query1”
```

Rest Service URIs

```
exec/xdcqry/id?qry=Q1
exec/xdcqry/id?qry=Q1&Price=25000
exec/xdcqry/id?qry=select * from XXX&name=MyQuery&cols=true&rows=2048
```

id is the ID for the XDC.

qry is the Name of the Query or a dynamic query to execute.

If you want to execute a dynamic query (one that has not been created in advance) you can supply the SQL and optionally use the **Name** parameter to control the resulting variable and optionally specify if you want the column names returned with the **Cols** parameter.

By default a dynamic Query will be limited to 2048 rows. To over-ride this specify **&rows**=nnn.

If the Query has parameters, supply them using &param=value

The resulting Xml will be formatted for Xcelsius and will contain a <variable> with a name set to the name of the Query.

An additional variable called "**QryInfo**" will also be returned and this contains the following metadata:

- # rows returned
- Query execution time in seconds and milliseconds
- Xml rendering time in seconds and milliseconds

The execution time can be used to fine tune the query.

The times do not include the time it takes for the consumer to display or process the Xml data.

The standard variable "**XDSReady**" is also returned and contains the date and time when the Cache was last refreshed. This can be useful to know how fresh the data is.

Burst XDC and deliver self contained .swf (Offline)

It is now possible to combine an XDC that has multiple Cache Keys with a Burst thatdelivers a .swf using the Native or HTML formats.

To 'burst' a .swf first the Xcelsius dashboard must be using the InfoSol "XDM Connector" and exported from the designer as a .swf.

The .swf can now be Cataloged with InfoBurst and a check will be made to see if the Connector has been used.

Add the .swf to a Burst and select either a Native format or HTML format for delivery and then use the new distribution option "XDCToEmbed" to choose the ID or Path for the XDC.

The Burst will extract each Xml Cache by Key and create a new .swf that contains this data. The resulting .swf can be delivered using any supported delivery (such as Network, Email etc).

If you wish to customize the name of the .swf you can use the delivery macro [=Key] for example "LatestSales_[=Key]" would generate "LatestSales_Arizona.swf" for a Cache key of "Arizona".

The resulting .swf is entirely self-contained and does not require anything else to run.

If you specify HTML and Email destination, a copy of the .swf is stored on the server and an in-line message is sent that links back to the server. The user will be prompted to authenticate (and the link uses the rest service).

Script Extensions

add delivery format=native .. **XDCToEmbed**=”\Caches\Revenue”

New Action “StartBurst”

A new Action has been added that makes it easier to kick off a Burst as a result of something else happening. A good example is where an XDC refresh schedule has just completed and we wish to start a Burst to deliver .swf with the latest data. This avoids having to setup a separate schedule with an event.

Script Command

new action type=”startburst” burst=”\Bursts\DeliverSWF”

REST interface now supports get/post and xml/json

In order to support almost any technology/platform the REST interface is now callable using either http GET or POST and can return data in 3 ways :

- XML with “verbose” node syntax
- XML with “attribute” syntax
- JSON

The parameter “attr” can be set to boolean true to use Xml attributes.

The parameter “json” can be set to boolean true to use Json.

The parameter “html” can be set to boolean true and combined with “json” to allow you to test json in a browser.

For example to search for all documents:

http://server:8551/infoburst/rest/**search/doc**

Returns: <objects><object><id>10</id> .. </object></objects>

/search/doc?attr=1

Returns: <objects><object id=”10” .. /></objects>

/search/doc?json=1

Returns: JSON with application/json mime type

/search/doc?json=1&html=1

Returns: JSON with texl/html mime type

New UserCache speeds up certain tasks

A new per-user Cache is now created and will dramatically speed up some tasks.

The focus of the Cache in this build is to make the XDC report part selection for WebI Documents operate much faster.

When a WebI report part is selected a number of calls are made to the Platform and due to the dynamic nature of IB09 the underlying WebI Document is opened in real time and the parts are discovered.

Now, the first time this is done the resulting parts are stored in the UserCache and so on subsequent calls to list or get parts for this Document there will be no need to connect/open/process the Document.

The UserCache will be extended in future builds to cache other types of content.

To allow for control over cached items some new API's are available.

SOAP APIs

Function GetUserCache(ByVal Token As String, ByVal UserID As Integer, ByVal ContentType As UserCacheContent, ByVal Options As String) As IB_UserCacheResult

Function GetNextUserCache(ByVal Token As String) As IB_UserCacheResult Function DeleteUserCache(ByVal Token As String, ByVal UserID As Integer, ByVal IDList() As Integer) As Integer

Function ClearUserCache(ByVal Token As String, ByVal UserID As Integer) As Boolean

```
Class IB_UserCacheResult
  Public Results() As IB_UserCache
  Public More As Boolean
  Public TotalResults As Integer
```

```
Class IB_UserCache
  Public ID As Integer
  Public Key As String
  Public ObjID As Integer
  Public DateCreated As Date
```

Script Commands

list usercache [user="x"] type="file" [opt="O:id"] [var="doc"]

delete usercache id="$doc"