

Methods for Extracting and Storing Records

SUG 10: Dave Evans



Agenda

- Basically, it's easy
 - 3 extraction methods, basically the same
 - AddToRecord method to, well add to record
 - Several details
- Example procs can be e-mailed
 - Not sure how much time we'll have, but will try to look at them and run them
 - You may find some of them are good examples of working with objects & methods & XmlDocs & XPath



Basic use: copying record, basically easy:

```

In INPUTFIL FRN %inRec
  %doc:LoadFromRecord
End For
In OUTPUTFL Store Record
Then Continue
  %doc:AddToRecord
End Store

```

- Beyond basic use: LoadFromRecord/NewFrom... is general-purpose, nicely structured record extract tool (debugging, data discovery, ...)
- Can customize data added by AddToRecord by manipulating XmlDoc; will show later in *CPYR_ALTER**
- Many details, which we hope LoadFromRecord/AddToRecord usually hides, but we'll discuss, in interest of a good nap



Using LoadFromRecord/... for debugging/data discovery

- Advantage over PAI:
 - Ability to manipulate structured extract (XmlDoc)
 - For example, you can exclude parts of a record you don't care about
 - See *CPYR_DEBUG_DATA NOPAI*
- Won't go deeply into debugging usage, but keep it in mind
- "Convenient for debugging" also applies to viewing input to AddToRecord



There are 3 extraction methods

- NewFromRecord - XmlDocument factory method
 - Nice alternative to LoadFromRecord, avoid %doc=New
- LoadFromRecord - generally refer to this
 - As NewFromRecord, convenient in record context, like we've already shown in FRN loop
 - Need to "clear" XmlDocument first
 - XmlNode class LoadFromRecord, to "gather" from several records
- ToXmlDoc - Record method object



Versions

- Sirius Mods version 7.8 - basis of presentation
 - Doc is in 7.8 Release Notes
- Sirius Mods 7.7
 - In this version of the mods, AddToRecord requires M204 V7R2
- Record copy methods pretty compelling for LOBs
- Fieldgroups and other V7R2 features
 - Make record copy methods even more compelling



Let's look at an XmlDoc

- See *CPYR_COPY...*

```

$Lstr_Set_UserBuffer('Value of
...':Left(500))
Store Record
  FOO = 'Value of FOO'
  LOB = Buffer, 1, 500
End Store
%rn = $CurRec
FRN %rn
  %doc = %doc:NewFromRecord
  %doc:Print
End For

```



Let's look at an XmlDoc (*continued*)

```

<Record version="1" file="QAWORK" number="0">
  <field name="FOO">
    Value of FOO
  </field>
  <field name="LOB">
    Value of LOB field, obviously can be
very...
  </field>
</Record>

```



Method object/result of LoadFromRecord/etc.

- %newDoc = %rec:**ToXmlDoc**
- %newDoc = %anyDoc:**NewFromRecord**
 - Must be inside "record loop", e.g., FRN
 - Method object can be null
 - Shared method, so can also do
 - ▶ %newDoc = %(XmlDoc):**NewFromRecord**



Method object/result of LoadFromRecord/etc. (continued)

- %docOrNode:**LoadFromRecord**
 - Must be inside "record loop", e.g., FRN
 - If %doc, or %rootNode, XmlDocument must be empty, 'Record' added as top level element
 - Otherwise %node must be an Element, and 'field' and 'fieldgroup' elements representing record added as children of %node



NewFromRecord/ToXmlDoc AllowNull

```
%doc = %(XmlDoc):NewFromRecord(-  
    [AllowNull=bool])
```

Since `NewFromRecord` and `ToXmlDoc` create an `XmlDoc`, this argument is propagated to created `XmlDoc`

`AllowNull` property of `XmlDoc` determines whether field values with `X'00'` are base 64 encoded



LoadFromRecord/etc. - CodepageTable

```
%doc:LoadFromRecord(CodepageTable=bool)
```

False is default; with True, attribute of Record element contains name of base codepage at time of record extraction, and `AddToRecord` will use the same table.

Addresses a probably rare translation issue, discussed later.



"Non-copy" arguments of LoadFromRecord/etc. - AttributeNames

```
%doc:LoadFromRecord(AttributeNames=False)
```

* True is default; with False you get:

...

```
<FIELD A>value of FIELD A</FIELD A>
```

```
<FIELD B>value of FIELD B</FIELD B>
```

...



"Non-copy" arguments of LoadFromRecord/etc. - AttributeValues

```
%doc:LoadFromRecord(AttributeValues=True)
```

* False is default; with True you get:

...

```
<field name="FIELD A" value="value of FIELD A"/>
```

```
<field name="FIELD B" value="value of FIELD B"/>
```

...



"Non-copy" arguments of LoadFromRecord/etc. - NamesToLower

```
%doc:LoadFromRecord(NamesToLower=True)
```

* False is default; with True you get:

...

```
<field name="fielda">value of FIELDA</field>
```

```
<field name="fieldb">value of FIELDB</field>
```

...



"Non-copy" arguments of LoadFromRecord/etc. - AllowUnreversible

* False is default and results in request

* cancellation if field name needs to be changed,

* e.g. with AttributeNames=False, blank in field

* translated to '.' for valid XML element name:

```
%doc:LoadFromRecord(AllowUnreversible=True, -  
AttributeNames=False)
```

...

```
<DUE.DATE>value of DUE DATE</DUE.DATE>
```



"Non-copy" arguments of LoadFromRecord/etc. - combined

```
%doc:LoadFromRecord(NamesToLower=True, -
AttributeValues=True, AttributeNames=False)
```

...

```
<fielda value="value of FIELDA"/>
```

```
<fieldb value="value of FIELDB"/>
```

...



AddToRecord syntax

```
%doc:AddToRecord(-
  [IgnoreUndefinedFields=bool,] -
  [DisableFieldConstraints=bool,] -
  [CopyIDs=bool,] -
  )
```

Adds field/fieldgroups contained in XmlDoc to current record - must be in record loop



IgnoreUndefinedFields

- AddToRecord(IgnoreUndefinedFields=False) -- default
 - If a 'field' or 'fieldgroup' element in the XmlDocument names a field or fieldgroup not defined in the file, request is cancelled
- AddToRecord(IgnoreUndefinedFields=True)
 - If a 'field' or 'fieldgroup' element in the XmlDocument names a field or fieldgroup not defined in the file, contents of 'field' or 'fieldgroup' element (entire subtree, in case of 'fieldgroup') is ignored



"Automatically generated" fields

- Fieldgroup IDs: preserved if AddToRecord(CopyIDs=True)
 - If 'groupID' attribute missing, new fieldgroup ID assigned
 - If CopyIDs=True, max fieldgroup ID of record also set from attribute on 'Record' element
- Derived fields (concatenated and CTO):
 - Like everything, LoadFromRecord extracts into XmlDocument
 - AddToRecord discards derived fields



"Automatically generated" fields (continued)

- Update tracking (e.g., CREATE-USER):
 - Like everything, LoadFromRecord extracts into XmlDoc
 - AddToRecord stores tracking fields (over-riding automatic value)



DisableFieldConstraints

- Since REDEFINE allows you to change field value constraints (e.g., LENGTH-EQ constraint), you may have "old" data in an XmlDoc which violates the current constraint
- If you have done that, then when copying a record, AddToRecord must ignore these field constraints
- This can be specified with DisableFieldConstraints=True
- See *CPYR_CONSTR*



Structure of the XmlDoc

```
<Record [version="1"] [file="---fileName---"]
  [number="---record number---"]
  [codepage="---codepage name---"]
  [maxGroupID="---max fieldgroup ID---"]>
  <field name="---field name---">
    ---field value---
  </field>
  <field name="---field name---"
    encoding="base64">
    ---base64 encoded field value---
  </field>
```



Structure of the XmlDoc (*continued*)

```
<fieldgroup name="---fieldgroup name---"
  [groupID="---fieldgroup ID---"]>
  <field ...
  <fieldgroup ...
  ...
</fieldgroup>
...
</Record>
```



Structure of the XmlDoc (*continued*)

- Top level element must be 'Record'
- 'Record' children may **only** be:
 - 'field' or 'fieldgroup' element
 - Any other element in non-null namespace
 - Comment or PI
 - no Text



Structure of the XmlDoc (*continued*)

- 'Record' attributes may **only** be:
 - 'version' - if present, value must be 1
 - 'maxGroupID' - if present, value must be nonnegative integer or null string
 - 'codepage' - if present, must be name of codepage
 - 'file' or 'number'
 - any additional attributes in a non-null namespaces



Structure of the XmlDoc (*continued*)

- 'field' element may have no element children
 - may have any number of PI/comment children
 - may have at most one text node child
- 'field' attributes may **only** be:
 - 'name' - required
 - 'encoding' - if present, must be 'base64'
 - any additional attributes in a non-null namespace



Structure of the XmlDoc (*continued*)

- 'fieldgroup' children may **only** be:
 - 'field' or 'fieldgroup' element
 - Any other element in non-null namespace
 - Comment or PI
 - no Text
- 'fieldgroup' attributes may **only** be:
 - 'name' - required
 - 'groupID' - optional
 - any additional attributes in a non-null namespace



Altering the XmlDocument

- You can use AddSubtree to accomplish the XmlNode class and LoadFromFieldgroup possible methods
- Caution with "hand-rolling" - you can lose the translation protection offered by base 64 encoding
- See *CPYR_ALTER**, *CPYR_NODE*



AddToRecordError exception class

- Thrown by AddToRecord; properties are:

Reason	Enumeration	AddToRecordErrorReason
Description	Longstring	
UntranslatableHexValue	Longstring	
FieldOrFieldgroupName	Longstring	
NodeName	Longstring	
NodeType	Enumeration	XmlNodeType
Value	Longstring	



AddToRecordErrorReason enumeration

InvalidNode
UntranslatableFieldName
UntranslatableFieldgroupName
UntranslatableValue
InvalidBase64
FieldNameTooLong
FieldgroupNameTooLong
ValueTooLong - e.g., > 255 for non-LOB
UnknownFieldName
UnknownFieldgroupName
ExpectedField
ExpectedFieldgroup
ErrorAddingField



AddToRecordErrorReason enumeration (continued)

ErrorAddingFieldgroup
ErrorObtainingRecord
InvalidFieldgroupID
InvalidCodepage
ErrorAddingMaxFieldgroupID
InsufficientStorageForLOB
InvalidVersion



Translation issues

- Field[group] names are EBCDIC
- Field values are generally EBCDIC characters
 - But can be arbitrary bytestreams
- Design is to preserve byte values
- XmlDocument strings are Unicode, so...
 - LoadFromRecord translates from EBCDIC to Unicode
 - AddToRecord translates from Unicode to EBCDIC
 - What if a character (either way) is untranslatable?
 - What if EBCDIC -> Unicode -> EBCDIC "round trip" results in different character?



Translation issues (*continued*)

- If field contains "problem" character
 - Field value is base 64 encoded
 - Attribute in 'field' element is used to flag this
 - See *CPYR_BASE64...*



Translation issues (*continued*)

```

%s Longstring Initial('08':x)
Store Record
  FOO = %s
End Store
%rn = $CurRec
FRN %rn
  %doc:NewFromRecord:Print
End For

      --- Result: ---
<Record version="1" file="QAWORK" number="0">
  <field name="FOO" encoding="base64">
    CA==
  </field>
</Record>

```



Translation issues (*continued*)

- Should we base 64 encode non-displayable characters?
- Seems that the choice is based on what's best for debugging (i.e., using LoadFromRecord for data debugging/discovery, or examining data while debugging applications which use LoadFrom/AddToRecord to copy records)
- Some data is translatable to Unicode control characters; for example, EBCDIC X'05' translates to Unicode U+0009 but that'll be serialized as '	' - maybe confusing for use in debugging. Print for EBCDIC X'00' - X'3F' show as &#x<hh>;
 - See *CPYR_TRANS*



Translation issues (*continued*)

- So - fields containing EBCDIC X'01'-X'3F' are all base 64 encoded
- Per XML Recommendation: XML document may not contain null (X'00') character
 - XmlDocument has AllowNull=True property to indicate null character is allowed
 - We won't base 64 encode null (X'00') if XmlDocument has AllowNull=True
 - NewFromRecord & ToXmlDoc create XmlDocument object - so AllowNull= argument is propagated to AllowNull property



Less likely translation issue

- What if standard Unicode <-> EBCDIC translation table used by LoadFromRecord differs from that used by AddToRecord?
- Solution
%doc:LoadFromRecord(CodepageTable=True)
uses **base** translation table for codepage in LoadFromRecord, and base of same codepage in AddToRecord



Possible Additional Methods

- XmlNode class AddToRecord
 - With XmlNode class LoadFromRecord, another way to create a "container" XmlDocument for merging several records, perhaps doing each in a Try block
 - Until implemented, can work around (see "Altering the XmlDocument")



Possible Additional Methods (*continued*)

- LoadFromFieldgroup
 - Like LoadFromRecord, but issued in a For Fieldgroup block, and extracts only the contents of the fieldgroup
 - Note this is a variant on the XmlDocument structure: fieldgroup element is top level element
 - Workaround requires extracting entire record and selecting fieldgroup



Possible Additional Methods (*continued*)

- LoadFromFieldgroup - in fieldgroup context
- AddToFieldgroup - in fieldgroup context

