
Notes for Sirius Mods Release 7.7

March, 2010



Sirius Software, Inc.
875 Massachusetts Avenue, Suite 21
Cambridge, MA 02139

Telephone: (617) 876-6677
FAX: (617) 234-1200
E-mail: support@sirius-software.com
World Wide Web: <http://sirius-software.com>

August 17, 2010

© 2010 Sirius Software, Inc.

Proprietary Notices

The following products:

- *Fast/Reload*
- *Janus SOAP*
- *Janus Sockets*
- *Janus Web Server*
- *Sirius Functions*

are proprietary products of Sirius Software, Inc.:

Sirius Software, Inc.
875 Massachusetts Avenue, Suite 21
Cambridge, Massachusetts 02139
USA

Model 204® and **Connect *™** are proprietary products of Computer Corporation of America, a wholly-owned subsidiary of Rocket Software, Inc., which owns the trademarks:

Rocket Software Corporate Office
M204 Division
275 Grove Street
Suite 3-410
Newton, Massachusetts 02466-2272
USA

SoftSpy™ is a proprietary product of Information Technology Systems:

Information Technology Systems
95 Wells Avenue
Newton, Massachusetts 02459-3216
USA

Contents

Proprietary Notices	ii
Contents	iii
Chapter 1: Introduction	1
Chapter 2: Maintenance and Support	3
Documentation	3
Chapter 3: All or Multiple Products	5
Model 204 V7R2 LOB field syntax	5
The User Language ADDCA utility	5
SSL certificate renegotiation	7
Chapter 4: Fast/Reload	9
Chapter 5: Janus SOAP ULI	11
New attribute for method declarations: CurrentRecord	11
New Unicode intrinsic methods: UnicodeToUpper and UnicodeToLower	12
New system class: Journal	13
Journal example	14
Journal methods	16
Discard subroutine	16
IsMerged property	16
MergeCount property	16
New constructor	17
StringFirstDateTime property	17
StringLastDateTime property	18
New System methods: CurrentRecordsUpdated and RecordsUpdated	18
New StringTokenizer method: SkipTokens	20
New parameter (Skip) for StringTokenizer NextToken method	21
New arguments for Record class ToXmlDoc method	22
Chapter 6: Janus SOAP XmlDoc API	23
AddTopElement method	23
ReplaceUnicode deserialization option	24
AddNamespace allowed on top level element with children	25
Length limit of XML names changed to 127	26

NewFromRecord, LoadFromRecord, and AddToRecord methods	26
Chapter 7: Compatibility/Bug fixes	27
Backwards incompatibilities	27
Janus SOAPXmlDoc API	27
DefaultURI argument of AddSubtree	27
Fixes in Sirius Mods 7.7 but not in 7.6	28
Version corequisites	29

CHAPTER 1 ***Introduction***

This document lists the enhancements and other changes contained in *Sirius Mods* version 7.7, which was released in March, 2010. The previous version of the *Sirius Mods*, 7.6, was available in November, 2009.

CHAPTER 2 ***Maintenance and Support***

Sirius Mods version 7.7 supports *Model 204* versions V7R2, V7R1, and V6R1.

2.1 Documentation

In addition to the incorporation of the new features described in “[All or Multiple Products](#)” on [page 5](#) and clarification of the information about client certificates, the organization of the ***Janus Network Security Reference Manual*** has been simplified and some redundancies have been removed.

3.1 Model 204 V7R2 LOB field syntax

When the *Sirius Mods* version 7.7 or greater are used with *Model 204* version V7R2 or greater, references to LOB fields without the “Buffer” syntax are now allowed (as long as the LSTRLOB parameter is 1, which is the default). For example, all of the following are now allowed using V7R2/7.7:

```

DEFINE FIELD LOB WITH BLOB
Begin
%s Longstring
...
Store Record
  LOB = %s
End Store
...
FRN %rn
  Add LOB = %s
  Insert LOB = %s
  Change LOB To %s
  %s = LOB
End For
End

```

3.2 The User Language ADDCA utility

The JANUS ADDCA command adds a “trusted” certifying authority's certificate (encrypted public key) to a Janus SSL server or client port. The User Language ADDCA utility, new in version 7.7 of the *Sirius Mods*, does the work of a series of ADDCA commands to let you add any number of the CA certificates that Sirius has already obtained and saved in the *UL/SPF SIRIUS* file.

Although a server port is not likely to need to use the JANUS ADDCA command more than once, a client port may want to use it to add multiple certificates. As a convenience especially for a CLSOCK port connecting to an SSL server, Sirius pre-loads a set of standard certifying authorities' certificates to the *SIRIUS* procedure file (as of *UL/SPF* 7.3).

Janus Sockets SSL client ports must add the certificate(s) of the CA that signed the certificate of the SSL server or servers to which the client will connect. (Janus does not allow a port to specify exceptions, that is, signed certificates that a port accepts without having the signing CA's certificate.)

The SIRIUS file certificates can be loaded to a port by the JANUS ADDCA command or by the ADDCA utility. Adding this set of certificates equips the client port in much the same way that internet browsers are equipped with multiple CA certificates.

The certificates are procedures whose names have the prefix “CA_” so they are easy to scan using a DISPLAY PROCEDURE command or by browsing the procedures in *SirPro*. Sirius will periodically review the certificates loaded to SIRIUS, eliminate any that have expired, and load new ones when they are made available by various well-accepted CAs.

The User Language ADDCA utility lets you add some or all of these certificates at once to an SSL port. The utility is invoked at the command level via INCLUDE:

```
IN SIRIUS INCLUDE ADDCA portname certificate
```

User Language ADDCA utility syntax

Where each parameter is positional and required:

portname The defined JANUS SSL port to which the certificate is to be added. The name can include wildcards.

Non-SSL ports may *not* have trusted CA certificates added to them.

certificate The name of the procedure that contains the base64 encoded CA certificate.

This parameter may contain wildcards. To load all certificates in the SIRIUS file whose names begin with “CA_AOL”, you can use: CA_AOL*

This ADDCA utility invocation loads (at least) the CA_ThawtePremiumServerCA certificates, for example:

```
IN FILE SIRIUS INCLUDE ADDCA WEBBY CA_T*
```

The utility internally runs a series of ADDCA commands like the following to load to WEBBY each certificate in file SIRIUS that matches the specified pattern:

```
JANUS ADDCA WEBBY SIRIUS CA_ThawtePremiumServerCA
JANUS ADDCA WEBBY SIRIUS CA_ThawteServerCA
JANUS ADDCA WEBBY SIRIUS CA_ThawteTimestampingCA
JANUS ADDCA WEBBY SIRIUS CA_ThawteSGCCA
```

Note: The CA certificates are stored as procedures with mixed-case names, which makes them much easier to scan visually, but a little more difficult to manipulate.

For instance, if the above INCLUDE command is used, all the intended certificates are loaded, because the “CA_T” in uppercase matches the case of the corresponding

characters in the certificate names. But if you want a more precise search and the name to be matched by the pattern includes mixed-case characters, the INCLUDE command must be bracketed with a set of *LOWER/*UPPER commands, as in:

```
JANUS DEFINE WEBBY * CLSOCK 10
OPEN FILE SIRIUS
*LOWER
IN SIRIUS INCLUDE ADDCA MYSSLPORT *America*
*UPPER
```

The above sequence of statements would load the certificate named “CA_AmericaOnlineRootCertificationAuthority1” from file SIRIUS.

3.3 SSL certificate renegotiation

As of *Sirius Mods* version 7.7, after the initial “handshake” between a Janus SSL server and a client to establish their secure connection, the server will start an SSL renegotiation (that is, a new handshake) that requests a digital certificate from the client if the following are both true:

- The server program requests information from a client certificate (via \$web_cert_levels, \$web_cert_info, the Socket class's CertInfo and CertLevels methods, or the XmlDoc class ClientCertificate method).
- The server port definition does *not* specify a SSLCLCERT or SSLCLCERTR parameter (so it does not cause a client certificate to be requested in the initial handshake).

This new ability to renegotiate allows a port to require a client certificate for some content, but not for other content.

In this renegotiation, the server requests a certificate, but does **not** insist that the client present one. It is the responsibility of the server application to take appropriate action if no client certificate is presented. Specifying that action depends on how the \$function or method that causes the renegotiation indicates that no certificate is returned.

Note: A client certificate will only ever be requested once for an SSL session. If the initial server-client handshake requested a certificate because the port definition included SSLCLCERT or SSLCLCERTR, a subsequent \$web_cert_info (for example) will not request a certificate again.

Similarly, if a \$web_cert_levels call (for example) caused a renegotiation by requesting a client certificate, a subsequent \$web_cert_info will not cause another request for a client certificate, whether or not a client certificate was returned for the initial renegotiation. This is because:

All or Multiple Products

- There is no reason a client would not return a certificate on an initial renegotiation, but return a certificate on a later renegotiation.
- There is no reason a client would return a certificate on an initial renegotiation, then return a different client certificate on a subsequent SSL renegotiation.

CHAPTER 4 ***Fast/Reload***

The following are new or changed features in *Fast/Reload*:

- NEWfgid option on LAI statement

The NEWfgid option on the LAI statement means that fieldgroup IDs from the TAPEI input file are **not** copied to fieldgroups created in the target file, but rather that fieldgroup IDs are recreated such that within each target record the ID numbers begin with 1 and are incremented by 1 for each subsequent fieldgroup.

Fieldgroups are a new feature as of *Model 204 V7R2*.

The following sections describe changes in the *Janus SOAP ULI* in this release.

5.1 New attribute for method declarations: CurrentRecord

A `CurrentRecord In File name` or a `CurrentRecord In Group name` clause on a method declaration indicates that:

- The method may only be invoked in a record context (established by a For Each Record loop, for example) for the declared file or group.

For example, the `printField` method, defined with the `CurrentRecord` attribute, is successfully invoked within an FR loop:

```
local subroutine (Record in file myproc):printField -
    currentRecord in file myproc
    for record currentRecord
        print NAME
    end for
end subroutine

for each record in %recset
    %rec = CurrentRecord
    %rec:printField
end for
```

- Statements within the method definition, even a `CurrentRecord` method call, may reference the record without having to be wrapped inside a record For loop.

For example, the `XmlDoc` class `LoadFromRecord` method, which normally requires a containing record loop, is valid as specified in the following method:

```
local function getDoc is object xmlDoc -
    currentRecord in file parents
    %doc is object xmlDoc
    %doc = new
    %doc:loadFromRecord
    return %doc
end function
```

Note: Under *Sirius Mods* 7.7, field references are an exception to this rule. You may not reference a record field from within a method declared with `CurrentRecord` without being inside a record `For` loop, unless you are running under *Sirius Mods* version 7.8 or higher.

For example, the `For Record currentRecord` and `End For` statements within the `printField` method in the preceding bullet item are not necessary in *Sirius Mods* version 7.8 or higher. In those versions, the `print NAME` statement succeeds by itself.

5.2 New Unicode intrinsic methods: `UnicodeToUpper` and `UnicodeToLower`

The `UnicodeToUpper` and `UnicodeToLower` functions return the Unicode alphabetic characters in the method object string as all-uppercase or all-lowercase characters, respectively. Non-alphabetic characters are returned as is, and the input string undergoes no other change.

Character references are returned for characters that do not translate to an EBCDIC character. The default substitute for non-displayable characters in your environment, say a question mark (?), is returned for characters that translate to an EBCDIC character that is not displayable.

These methods produce a one-character for one-character translation, which will fail to correctly handle unusual characters whose uppercase versions have two-characters, for example.

```
%outUni = unicode:UnicodeToUpper
%outUni = unicode:UnicodeToLower
```

UnicodeToUpper and UnicodeToLower syntax

Example `UnicodeToUpper` and `UnicodeToLower` statements follow:

```
. . .
%u is unicode
%u = ' } abC5xyZ! &#x394; &#x531; &#xDF; ':u
printText {~} = '{%u:toUpper}'
printText {~} = '{%u:toLower}'
. . .
```

The results are:

```
%u:UnicodeToUpper = ' } ABC5XYZ! &#x0394; &#x0531; ?'
%u:UnicodeToLower = ' } abc5xyz! &#x03B4; &#x0561; ?'
```

5.3 New system class: Journal

The Journal class provides an object-oriented interface to CCAJLOG or CCAJRNL datasets and streams, including those that have been aggregated by the MERGEJ utility. A *Model 204* stream is a datastream comprising one or more sequential journals which is created by the *Model 204* DEFINE STREAM command (see the ***Model 204 Command Reference Manual***).

To extract information programmatically about the contents of a Journal object, you use the methods of the Journal class. To view such a journal's actual data, you use the Stringlist class AppendJournalData method — or you use *SirScan*, which implicitly calls AppendJournalData.

The Journal class operates on CCAJRNL or CCAJLOG datasets or streams, which you make available to your Online by any of the following:

- An MVS DD card.
- A VSE DLBL card.
- A CMS FILEDEF command.
- A *Model 204* ALLOCATE command (see the ***Model 204 Command Reference Manual***).
- A *Model 204* DEFINE STREAM command (see the ***Model 204 Command Reference Manual***).

5.3.1 Journal example

The following example is a program that displays the properties and contents of a historical CCAJRNL. The request makes use of a low setting of the parameter option MAXREC of the AppendJournalData method to control the amount of journal data that is displayed:

```
Begin
%list is object stringlist
%rc is float
%jStartTime is longstring
%jEndTime is longstring
%journal is object Journal
%msg is longstring
%stime is longstring
%etime is longstring
%jhold is longstring
%parm is longstring
%threads is longstring

* Get old journal and view its properties
%journal = new('OLDJRNL')
if %journal eq Null then
    %msg = 'Journal is invalid or not allocated.'
    jump to someErrorCondition
end if

printText {~} is {%journal:isMerged:toString}
printText {~} is {%journal:MergeCount}
printText {~} is {%journal:stringFirstDateTime}
printText {~} is {%journal:stringLastDateTime}

* Set values for appendJournalData call
%list = new
%parm = 'AA USER MAXREC=100'
%threads = '*'
%stime = '1012202370444'
%etime = '1012302373097'

* Test for time values that are out of range
%jStartTime = %journal:stringFirstDateTime
%jEndTime = %journal:stringLastDateTime
if %stime ge %jEndTime:subString(2) Then
    %jhold = %jEndTime
    %msg = 'Requested start time is after end of journal.'
    jump to someErrorCondition
elseif %etime lt %jStartTime:subString(2) then
    %jhold = %jStartTime
    %msg = 'Requested end time is before start of journal.'
    jump to someErrorCondition
end if
```

```

* Call the method
%rc = %list:appendJournalData( -
    StartTime=%sTime, -
    EndTime=%eTime, -
    Options=%parm, -
    Threads=%threads, -
    journal=%journal)

Print %list:count AND 'list items:'
If %list:count eq 0 then
    Print %rc And 'is return code'
End If
%list:print

someErrorCondition:
    * do some error processing here
    Print %msg

End

```

The result is:

```

%journal:isMerged:toString is False
%journal:MergeCount is 0
%journal:stringFirstDateTime is 11012202370444
%journal:stringLastDateTime is 11012509464897
100 list items:
0
*****
**
0 * MODEL 204 IS PROPRIETARY TO AND A REGISTERED TRADEMARK
OF COMPUTER
*
0 * CORPORATION OF AMERICA. ALL TITLE, COPYRIGHT AND OTHER
PROPRIETARY
*
0 * RIGHTS SHALL BE RETAINED BY COMPUTER CORPORATION OF
AMERICA. MODEL
204 *
0 * HAS BEEN LICENSED BY THIS CUSTOMER AND THE USE AND
PROTECTION OF
*
0 * MODEL 204 IS GOVERNED BY THE LICENSE AGREEMENT BETWEEN
THE CUSTOMER
*
0 * AND COMPUTER CORPORATION OF AMERICA.
*
0 *
*
0 * COPYRIGHT (C) 2010 COMPUTER CORPORATION OF AMERICA
*
0 * ALL RIGHTS RESERVED
*

```

```
Ø
*****
**
Ø M204.0060: MODEL 204 INITIALIZATION.  VERSION = 7.2.0D
EVCP/RSQL
    VERSION = 7.2.0D  03/01/10 23.11
Ø M204.0061: SMF SYSTEM ID = CMS, JOB NAME = ULSPFPRO,
STEP NAME =
    02?|?2, JES ID =
Ø M204.0062: EXECUTE PARAMETERS:
SYSOPT=187,LIBUFF=1600,NJBUFF=31,
    RESTARTU=1,SIRTUNE=0
Ø M204.1119: READING PARAMETERS
. . . (70+ lines not shown)
```

5.3.2 Journal methods

The individual Journal methods are described in the following sections.

5.3.2.1 Discard subroutine

This method deletes a Journal object. It takes no arguments.

```
%jrn1:discard
```

Discard syntax

5.3.2.2 IsMerged property

This read-only property indicates with a boolean result whether the journal is a merged journal, that is, produced by the MERGEJ utility.

```
%bool = %jrn1:isMerged
```

IsMerged syntax

5.3.2.3 MergeCount property

This read-only property indicates the number of journals that were input to the MERGEJ utility to produce the method object journal.

```
%num = %jrn1:mergeCount
```

MergeCount syntax

5.3.2.4 New constructor

This method creates a new instance of a Journal object. Its required parameter is a string that is the DDname of the journal dataset or the name of the stream that is to be accessed.

```
%jrn1 = new(name)
```

New syntax

Notes:

- In *Sirius Mods* version 7.7, the New method returns a Null object if the journal is not present or is not a valid journal. In *Sirius Mods* version 7.8 and later, New also throws a BadJournal exception on an error, and it will not return a Null Journal object without throwing an exception.

Consequently, invocations of New in *Sirius Mods* version 7.7 need to include a test with an On Error unit and a test for a Null result.

- In *Sirius Mods* version 7.7, New does not require the input journal to be from the same *Model 204* release as the Online. However, if the releases are different, an AppendJournalData call that references this journal will silently fail to add any data to its output Stringlist. As of *Sirius Mods* version 7.8, an attempt to instantiate a journal object variable that references a journal from a different version than the Online produces a BadJournal exception.

5.3.2.5 StringFirstDateTime property

This read-only property returns the datetime stamp from the first entry in the journal stream. This 14-character value is in CYYDDHMISSXX format: CYY = year - 1900, DDD = Julian day number, HH = hour, MI = minutes, SS = seconds, XX = hundredths of seconds)

```
%string = %jrn1:stringFirstDateTime
```

StringFirstDateTime syntax

If you want to use the result of this method as, or in connection with, the `StartTime` parameter of the AppendJournalData method, be sure to remove the leading century indicator, because that parameter must be specified in YYDDHMISSXX format. For example, you can use the intrinsic Substring method, as shown in the “test for time values that are out of range” in [“Journal example” on page 14](#).

5.3.2.6 **StringLastDateTime property**

This read-only property returns the datetime stamp from the last entry in the journal stream. This 14-character value is in CYYDDHMMSSXX format: CYY = year - 1900, DDH = Julian day number, MM = hour, SS = minutes, XX = seconds, XX = hundredths of seconds

```
%string = %jrn1:stringLastDateTime
```

StringLastDateTime syntax

If you want to use the result of this method as, or in connection with, the `EndTime` parameter of the `AppendJournalData` method, be sure to remove the leading century indicator, because that parameter must be specified in YYDDHMMSSXX format. For example, you can use the intrinsic `Substring` method, as shown in the “test for time values that are out of range” in [“Journal example” on page 14](#).

5.4 **New System methods: CurrentRecordsUpdated and RecordsUpdated**

These System class shared functions return a Boolean value that indicates whether a record has been updated in the current transaction. The methods apply, respectively, to the current record in a record-oriented loop, or to a record you identify by file name and record number.

The syntax of the methods follows:

```
%bool = %(system):CurrentRecordIsUpdated
```

```
%bool = %(system):RecordIsUpdated(filename, recnum)
```

The following example suggests how the methods might be used. In this request fragment, a record updating loop reports to the audit trail if it will be the initial update for each record in the transaction:

```
%recs is object recordset in file myfile
%rec is object record in file myfile

fd to %recs
end find
for each record in %recs
  %rec = CurrentRecord
  if %(system):currentRecordIsUpdated ne true then
    Audit 'First update of Record num: ' %rec:RecordNumber
  end if
  ...
  * perform record update
end for
```

A similar task can be done using the RecordIsUpdated method:

```
%recs is object recordset in file myfile
%rec is object record in file myfile

fd to %recs
end find
for each record in %recs
  ...
  * if some condition is satisfied
    %rec = CurrentRecord
    if %(system):RecordIsUpdated($curfile, $currec) ne true then
      Audit 'First update of Record num: ' %rec:RecordNumber
    end if
    ...
    * perform record update
  end if
end for
```

5.5 New StringTokenizer method: SkipTokens

This method moves the current token and tokenizing position forward in the tokenizer string the number of tokens specified in the method argument. Skipping one token is equivalent to executing the NextToken method; skipping three tokens is equivalent to executing NextToken three times consecutively.

If a token is found after advancing the designated number of tokens, the tokenizing position in the string is moved after the found token, and the CurrentToken method will return the found token. If the end of the tokenizer string occurs before the specified number of tokens, the request is cancelled.

`%tok:SkipTokens(num)`

SkipTokens syntax

The value of *num* must be greater than 0 and less than the number of tokens remaining in the tokenizer string.

For a new StringTokenizer instance, issuing NextToken, SkipTokens, or FindToken is required before CurrentToken may be issued without error.

The following code fragment shows multiple SkipTokens calls. The second call follows a move of the tokenizing position to the middle of the initial token in the tokenizer string.

```
begin
  %tok is object StringTokenizer
  %tok = new
  %tok:string = 'Here is the skipTokens example.'
  printtext {~} = '{%tok:string}'

  %tok:skipTokens(3)
  printText {~} = {%tok:currentToken}
  printText {~} = {%tok:nextPosition}
  %tok:nextPosition = 3
  %tok:skipTokens(1)
  printText {~} = {%tok:currentToken}
  printText {~} = {%tok:nextPosition}
  %tok:skipTokens(4)
  printText {~} = {%tok:currentToken}
end
```

The result is:

```
%tok:string = 'Here is the skipTokens example.'  
%tok:currentToken = the  
%tok:nextPosition = 12  
%tok:currentToken = re  
%tok:nextPosition = 5  
%tok:currentToken = example.  
%tok:nextPosition = 32
```

5.6 New parameter (Skip) for StringTokenizer NextToken method

The NextToken Skip parameter specifies the number of tokens to skip before returning the next token. This optional, name required parameter gives the NextToken method the flexibility to advance more than a single token at a time.

The new syntax of NextToken is:

```
%string = %tok:NextToken([Skip=num])
```

The value of *num* must be greater than 0, and it must be less than the number of tokens remaining in the tokenizer string.

The following request fragment shows the effect of the Skip parameter:

```
%tok:string = ' * 0 * 1 * 2 * 3 * 4 * 5 * 6 * 7'  
repeat while %tok:notAtEnd  
  printtext {~} = '{%tok:nextToken(skip=1)}'  
end repeat
```

The result is:

```
%tok:nextToken(skip=1) = '0'  
%tok:nextToken(skip=1) = '1'  
%tok:nextToken(skip=1) = '2'  
%tok:nextToken(skip=1) = '3'  
%tok:nextToken(skip=1) = '4'  
%tok:nextToken(skip=1) = '5'  
%tok:nextToken(skip=1) = '6'  
%tok:nextToken(skip=1) = '7'
```

5.7 New arguments for Record class ToXmlDoc method

The ToXmlDoc method in the Record class has the following new arguments:

CodepageTable=False|True

This Boolean argument, which defaults to `False`, specifies whether to use the base codepage translation table when creating the XmlDoc.

This argument was actually introduced in version 7.6 of the *Sirius Mods*, but as explained in the Note below, you must see the **Notes for Sirius Mods Release 7.8** for further discussion of this argument.

AllowNull This value of this Boolean argument, which defaults to `False`, is copied to the AllowNull property of the XmlDoc created by ToXmlDoc. The XmlDoc's AllowNull property, in turn, determines whether field values that contain the 'X'00' character are stored in the XmlDoc with base64 encoding. Such values are base64 encoded if the AllowNull value is `False`.

As explained in the Note below, you must see the **Notes for Sirius Mods Release 7.8** for further discussion of this argument.

Note: ToXmlDoc and AddToRecord (introduced in *Sirius Mods* 7.3 and 7.6, respectively) and version 7.7's NewFromRecord and LoadFromRecord ([“NewFromRecord, LoadFromRecord, and AddToRecord methods” on page 26](#)) are the “record copying” methods. Except for ToXmlDoc, their full documentation is provided in the **Notes for Sirius Mods Release 7.8**, because 7.8 is the first release in which their full functionality is available to all customers. For example, only customers using *Model 204 V7R2* can use the AddToRecord method with the *Sirius Mods* prior to version 7.8.

The following sections describe changes in the *Janus SOAP XmlDocument API* in this release.

6.1 AddTopElement method

```
%newElementNode = %xmlDoc:AddTopElement(elementName, [URI])
```

This method adds a new Element node as the new top-level element of the XmlDocument that is the object method, making all of the former children of the XmlDocument Root node into children of the new element.

The *elementName* argument, a Unicode string, is the name to be given to the new element; it is required.

The *URI* argument is optional and defaults to the null string.

- If *elementName* contains a prefix, then *URI* specifies the URI associated with that prefix for the element, so it must be a non-null string that is a valid URI.
- If *elementName* does not contain a prefix, then *URI* must be omitted or must be the null string. This rule means that this URI argument is more restrictive than the URI argument of the AddElement method.

When you need to change the structure of an XML document and that need can be accomplished with AddTopElement, it is a more efficient approach than using AddSubtree. For example, if you want to make a copy of an XML document adding “SOAP wrappers” to it, you can use the following:

```
%doc2 = %doc:DeepCopy
%doc2:AddTopElement('soap:Body', -
    'http://schemas.xmlsoap.org/soap/envelope/')
%doc2:AddTopElement('soap:Envelope', -
    'http://schemas.xmlsoap.org/soap/envelope/')
```

Notes:

- The order of the AddTopElement invocations is the reverse of the final order of the elements, that is, the second invocation creates the first element in the final document.

- If you have no need to preserve the original XmlDocument, then in the above example, both AddTopElement invocations can use %doc as the object method, and the DeepCopy can be omitted.
- In the above example, there will be a redundant namespace declaration on the soap:Body element; this is unavoidable.

6.2 ReplaceUnicode deserialization option

The ReplaceUnicode option for deserialization methods (for example, LoadXml) specifies that Unicode characters are to be converted using the replacements specified for your site by UNICODE updating commands that use the Rep subcommand (for example, UNICODE Table Standard Rep U=2122 '(TM)').

The replacement is performed on all names, element and attribute values, comments, and PI “values” in the document, after any entity and character references have been converted to characters.

For example, assume the following command is in CCAIN:

```
UNICODE Table Standard Rep U=2122 '(tm)'
```

Given the above command, the following fragment:

```
%u Unicode Initial('<a>')
%u = %u:UnicodeWith('2122':X:Utf16ToUnicode)
%u = %u:UnicodeWith('</a>':U)
%d:LoadXml(%u, 'ReplaceUnicode')
%d:Print
```

results in:

```
<a>(tm)</a>
```

In the preceding example, the stream of input characters to LoadXml contains the Unicode character U+2122. Since the ReplaceUnicode option applies to both the stream of input characters and to the character value of character references, the following fragment (assuming the same CCAIN line as above):

```
%d:LoadXml('<a>&#x2122;</a>', 'ReplaceUnicode')
```

also results in:

```
<a>(tm)</a>
```

In this case, U+2122 does not occur in the input character stream, but it is the value of the character reference.

Notes:

- It is an error to be processing a replacement string within a character reference. For example, assume the following two lines are in CCAIN:

```
* Replace superscript 2 with digit '2':  
UNICODE Table Standard Rep U=00B2 '2'
```

Given the above command, the following fragment gets a parse error, because the replacement string is being used as part of a character reference:

```
%d:LoadXml('<a>&#x' With '&#xB2;':U With ';'</a>', -  
'ReplaceUnicode')
```

As a consequence of this rule, a replacement string should not contain an ampersand character (assuming that the ReplaceUnicode option will be used).

- As mentioned, replacement of a Unicode character due to the ReplaceUnicode option is only done while processing names and values in the XML document. It is an error if the end of the name or value occurs and the replacement string has not been exhausted. In other words (again assuming that the ReplaceUnicode option will be used), a replacement string should not have “XML markup” which might end a string, such as a quotation mark or a left angle bracket (<). For example, assume the following line is in CCAIN:

```
UNICODE Table Standard Rep U=2122 '(trademark)<tm>'
```

Given the above command, the following fragment gets a parsing error, because when the '<' is encountered in the replacement string, that ends the element content:

```
%d:LoadXml('<a>&#x2122;</a>':U, 'ReplaceUnicode')
```

- If a parsing error occurs after processing a Unicode character that has been replaced, the error display of the input stream will contain the replacement string, and the replaced character will not be displayed. However, if the character being replaced was introduced as a character reference, the character reference remains in the display of the input stream.

6.3 AddNamespace allowed on top level element with children

The AddNamespace method has been enhanced to let you add non-default namespace declarations to the top level element, even if the top level element has element children.

6.4 Length limit of XML names changed to 127

The length limit of XML names has been changed from 100 to 127. For example, the prefix part must be shorter than 128 characters, and the local name part must be shorter than 128 characters, for the “name” argument of the `AddElement` method.

Also, the *Janus SOAP Reference Manual* “XML syntax” section has been corrected. It previously stated that the maximum length of an XML name is 650 characters, which was not correct.

6.5 NewFromRecord, LoadFromRecord, and AddToRecord methods

The `NewFromRecord` and `LoadFromRecord` methods are implemented. They perform the same operation (copying the contents of a record to an `XmlDoc`) as the `Record` class `ToXmlDoc` method.

The `AddToRecord` method has been updated. `AddToRecord` copies the fields and fieldgroups contained in an `XmlDoc` (such as that created by `LoadFromRecord`, `NewFromRecord`, or `ToXmlDoc`), storing into the current record. In the version 7.7 *Sirius Mods*, `AddToRecord` requires version V7R2 of *Model 204*.

Note: For further discussion of these “record copying” methods, you must see the *Notes for Sirius Mods Release 7.8*. The full documentation of these methods is provided there, because 7.8 is the first release in which their full functionality is available to all customers (that is, those not using *Model 204 V7R2*).

This chapter lists any compatibility issues with prior versions of the *Sirius Mods* and any bugs which have been fixed in this version of the *Sirius Mods* but had not, as of the date of this release, been fixed in the previous generally available version (7.6).

In general, backward incompatibility means that an operation which was previously performed without any indication of error, now operates, given the same inputs and conditions, in a different manner. We may not list as backwards incompatibilities those cases in which the previous behaviour, although not indicating an error, was “clearly and obviously” incorrect, and which are introduced as normal bug fixes (whether or not they had been fixed with previous maintenance).

7.1 Backwards incompatibilities

Backwards incompatibilities are described per product in the following sections.

7.1.1 Janus SOAPXmlDoc API

The following backwards compatibility issues have been introduced in the *Janus SOAP XmlDOC* API.

7.1.1.1 DefaultURI argument of AddSubtree

In some cases, an Element in a default namespace, which was added to the XmlDoc by a deserialization method, will not get the correct namespace URI when it is copied using the DefaultURI argument of the AddNamespace subroutine.

This problem was fixed, and a resulting incompatibility introduced in the version 7.7 *Sirius Mods* by maintenance supplied by ZAP77A4 on 17 August, 2010.

For example:

```
Text To %s1
<a:a xmlns:a="http://aaa" xmlns="http://ddd">
  <b:b xmlns:b="http://bbb">
    <c>123</c>
  </b:b>
</a:a>
End Text
%in:LoadXml(%s1)
%n Object XmlNode
%n = %in:SelectSingleNode('/**/*')
%out:AddSubtree(%n, DefaultURI='u:who')
```

Prior to applying ZAP77A4, the above results in:

```
<b:b xmlns:b="http://bbb">
  <c xmlns="http://ddd">
    123
  </c>
</b:b>
```

The correct result, as produced with ZAP77A4 applied, is as follows (note the namespace for element 'c'):

```
<b:b xmlns:b="http://bbb">
  <c xmlns="u:who">
    123
  </c>
</b:b>
```

7.2 Fixes in Sirius Mods 7.7 but not in 7.6

This section lists fixes to functionality existing in the *Sirius Mods* version 7.6 but which, due to the absence of customer problems, have not, as of the date of the release, been fixed in that version.

- There are no fixes in *Sirius Mods* 7.7 that are not available in previous versions.

7.3 Version corequisites

This section lists any restrictions on usage of various products (including *Sirius Mods* itself) that will be imposed by use of version 7.7 of *Sirius Mods*.

- There are no corequisites associated with *Sirius Mods* 7.7.

