

---

# Notes for Sirius Mods Release 7.5

**April, 2009**

---



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](mailto:support@sirius-software.com)  
World Wide Web: <http://sirius-software.com>

August 6, 2010

© 2010 Sirius Software, Inc.

---

## *Proprietary Notices*

The following products:

- *Janus SOAP*
- *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®** is a proprietary product of Computer Corporation of America, a wholly-owned subsidiary of Rocket Software, Inc., which owns the trademark:

**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

<b>Proprietary Notices</b> . . . . .	<b>ii</b>
<b>Contents</b> . . . . .	<b>iii</b>
<b>Chapter 1: Introduction</b> . . . . .	<b>1</b>
<b>Chapter 2: Maintenance and Support</b> . . . . .	<b>3</b>
Model 204 support . . . . .	3
<b>Chapter 3: All or Multiple Products</b> . . . . .	<b>5</b>
New Model 204 parameters . . . . .	5
GCSTATS . . . . .	5
AUTOGCN . . . . .	5
<b>Chapter 4: Janus SOAP ULI</b> . . . . .	<b>9</b>
New exception class: MaxDaemExceeded . . . . .	9
Unicode intrinsic methods . . . . .	9
UnicodeChar function . . . . .	10
UnicodeLeft function . . . . .	10
UnicodeLength function . . . . .	11
UnicodePositionIn and UnicodePositionOf functions . . . . .	11
UnicodeRight function . . . . .	12
UnicodeSubstring function . . . . .	13
Intrinsic conversion methods . . . . .	14
FloatToBinary and BinaryToFloat . . . . .	14
IntegerToHex and HexToInteger . . . . .	15
IntegerToHex . . . . .	16
HexToInteger . . . . .	16
New EbcDicToUnicode parameter: Untranslatable . . . . .	17
<b>Chapter 5: Janus SOAP Xml API</b> . . . . .	<b>19</b>
Support 1.1 as XML version . . . . .	19
The ClientCertificate method . . . . .	19
CrPreserve deserialization option . . . . .	21
<b>Chapter 6: Janus Web Server</b> . . . . .	<b>23</b>
TextUtf8 option on \$Web_*_Content functions . . . . .	23

<b>Chapter 7: Compatibility/Bug fixes</b> . . . . .	<b>25</b>
Backwards incompatibilities . . . . .	25
Janus SOAP ULI . . . . .	25
EbcDicRemoveNonUnicode and EbcDicTranslateNonUnicode methods removed . . . . .	25
ToXmlDoc default changed to AttributeNames=True . . . . .	25
Fixes in Sirius Mods 7.5 but not in 7.4 . . . . .	26
Version corequisites . . . . .	26

---

**CHAPTER 1** ***Introduction***

This document lists the enhancements and other changes contained in *Sirius Mods* version 7.5, which was released in April, 2009. The previous generally-released version of the *Sirius Mods*, 7.3, was available in October, 2008. An intermediate version, 7.4, was released to some customers in February, 2009.

Because 7.4 was an intermediate version, all customers running that version are expected to upgrade to *Sirius Mods* 7.5 as soon as possible. Also, because *Sirius Mods* 7.4 was not a general release, these release notes describe all enhancements to the *Sirius Mods* including those that were available in 7.4.



---

**CHAPTER 2** ***Maintenance and Support*****2.1 Model 204 support**

*Sirius Mods 7.5* supports only *Model 204* versions V6R1 and V7R1. The commercial release of *Model 204* V7R1 is supported only by *Sirius Mods 7.5* and higher.



### 3.1 New Model 204 parameters

Two new parameters that affect *Janus SOAP ULI* “garbage collection” are added.

#### 3.1.1 GCSTATS

Setting this User parameter reports garbage collection statistics in a message to the audit trail, to the terminal, or to both. The statistics include the number of objects discarded by the garbage collection process and the amount of time taken.

GCSTATS is a bitmask parameter whose settings are:

**X'00'** Do not send garbage collection statistics. This is the default.

**X'01'** Send statistics to the audit trail.

**X'03'** Send statistics to the audit trail and to the terminal.

The message that is sent is similar to the following, which reports the discarding of 10001 of the 10002 objects still occupying system resources for this thread and session after this request's normal cleanup:

```
*** MSIR.0995: (explicit) Garbage collection completed in  
16ms realtime with 16ms CPU time. Discarded 10001/10002  
objects.
```

The `(explicit)` tag in the message above signifies that garbage collection was initiated explicitly by a `%(object):GarbageCollect` call. An `(implicit)` tag displays if the garbage collection is invoked for a user by the system because the AUTOGCN (User parameter) threshold number (or more) of “orphaned” objects remained after the request's normal object cleanup.

For more information about garbage collection, see the *Janus SOAP Reference Manual*.

#### 3.1.2 AUTOGCN

The *Janus SOAP ULI* performs garbage collection automatically at user logout: that is, at logout it removes user-created objects that were not discarded by the user or by the *Janus SOAP ULI* during and after each request. This garbage collection process can

also be invoked explicitly (by `%(object):GarbageCollect`), and it may be time consuming. The `AUTOGCN User` parameter lets you invoke garbage collection automatically during a user session whenever normal post-request cleanup “leaves behind” a number of objects that meets or exceeds the `AUTOGCN` value.

At the end of a request, normal *Janus SOAP ULI* object cleanup discards all objects — if there are no global or session objects defined in the request. However, if the request created global or session objects, normal cleanup discards only objects that are not global and not session and have no references by other request object variables. Global and session objects and any other objects they reference or that are still referenced by other request object variables are **not** discarded.

Garbage collection sorts through these non-discarded objects and removes those that are not global or session objects or are not accessible, directly or indirectly, from a global or session object. An orphan cycle is a prime example of such an unreachable object. Orphans reference each other and may be complex, associated with many resources. By invoking garbage collection automatically, the `AUTOGCN User` parameter lets you keep the number of non-discarded, referenced but unreachable, objects below a threshold value.

If `AUTOGCN > 0`, garbage collection runs whenever the number of referenced, non-global, non-session objects created by the current request and not discarded after the current request's normal cleanup is at least the `AUTOGCN` threshold number.

If `AUTOGCN = 0`, automatic garbage collection is not invoked until user logout.

If garbage collection runs, whether `AUTOGCN`-invoked or via a `GarbageCollect` method call, you can be notified with a Sirius message if you set the `GCSTATS User` parameter to one of its non-default values. `GCSTATS` (“[GCSTATS](#)” on page 5) produces a message that indicates the number of objects the garbage collection discarded and how long it took to do so.

The request in the following example creates 50 self-referential, orphan objects which are not discarded by normal request cleanup because the request also creates a global object. The request does not trigger automatic garbage collection, however, because the `AUTOGCN` threshold is 70.

```
R GCSTATS X'03'  
R AUTOGCN 70  
Begin  
  
class Linked  
public  
    variable next is object Linked  
end public  
end class  
  
%cycle is object Linked  
%gl is object linked global  
%i is float
```

```
%gl = new

for %i from 1 to 50
  %cycle = new
  %cycle:next = %cycle
end for

print 'End of request'
End
```

If the request runs a second time, the result is the same as the first time, and now one hundred orphans remain in storage. If the request runs a third time but AUTOGCN is set to 30, the 50 orphans that are newly created and not discarded exceed the AUTOGCN threshold, and automatic collection runs, discarding the orphans from all three requests, but not discarding the global object. The third request produces the following result:

```
End of request
*** MSIR.0995: (implicit) Garbage collection completed in
0ms realtime with 0ms CPU time. Discarded 150/151 objects.
```

**Note:** If the second and third times the request ran it was modified to exclude the creation of the global object, the requests' orphans would still not be discarded, because the presence of the global object in the initial request affects object cleanup for the life of the user thread, as long as the global is not explicitly discarded.



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

## 4.1 New exception class: **MaxDaemExceeded**

The **MaxDaemExceeded** exception class indicates that the Daemon class constructor was invoked, but the calling thread is at its limit of daemon threads. The maximum number of daemons allowed per master thread is set by the **MAXDAEM** system parameter, whose default value is 1.

The only method of the **MaxDaemExceeded** exception class is the **New** constructor. The class has no properties.

**New** This callable constructor generates an instance of an **MaxDaemExceeded** exception.

To produce a **MaxDaemExceeded** exception, you typically use a User Language Throw statement with an **MaxDaemExceeded New** constructor. The **New** method format follows:

```
[%(MaxDaemExceeded):]New
```

### **MaxDaemExceeded constructor syntax**

#### **Example**

The following statement throws a **MaxDaemExceeded** exception:

```
throw %(maxDaemExceeded):new
```

## 4.2 Unicode intrinsic methods

Multiple new methods are added, as described in the following subsections. Many of the examples use the **PrintText** statement, which was introduced in *Sirius Mods* version 7.2 and is described in the ***Janus SOAP Reference Manual***.

### 4.2.1 UnicodeChar function

This function returns the string value of the single character at a specified position in the method object Unicode string.

```
%outUni = unicode:UnicodeChar(position)
```

#### UnicodeChar syntax

The *position* value must be a non-negative, non-zero number; a zero or negative number results in request cancellation. A *length* value greater than the declared length of the output string results in request cancellation.

For a given position, the UnicodeChar function returns the same value as the UnicodeSubstring function with a length argument of 1.

The UnicodeChar method is analogous to the String intrinsic Char method.

The following statement displays a g character:

```
printText {'inaugural':unicodeChar(5)}
```

### 4.2.2 UnicodeLeft function

This function returns the left-most characters of the method object string, possibly padding it on the right.

```
%outUni = unicode:unicodeLeft(length [, Pad=char])
```

#### UnicodeLeft syntax

The *length* value must be a non-negative number. A negative number results in request cancellation.

The pad parameter value must be either null or a single character. A longer value results in a compilation error. The default is a blank.

The UnicodeLeft method is identical to the UnicodeSubstring method ([“UnicodeSubstring function” on page 13](#)) with a 1 for the first argument.

The UnicodeLeft method is analogous to the String intrinsic Left method.

The following statement, which uses the U constant method, displays Wheeeee:

```
%u is unicode initial('Wh':U)
printText {'Wh':unicodeLeft(7, pad='e')}
```

### 4.2.3 UnicodeLength function

This function returns the number of characters in the method object string.

```
%len = unicode:unicodeLength
```

#### UnicodeLength syntax

The UnicodeLength method is analogous to the String intrinsic Length method.

The following example fragment prints a Unicode string, then its length:

```
%u unicode initial ('&#x40;xyz':U)
Print %u
Print %u:unicodelength
```

The result is:

```
@xyz
4
```

### 4.2.4 UnicodePositionIn and UnicodePositionOf functions

These functions return the numeric position of the first occurrence of one character string inside another (character case respected). The difference between the two methods is that for UnicodePositionIn, the method object string is located in the first argument string, whereas for UnicodePositionOf, the first argument string is located in the method object string. Which method is more convenient to use will be application dependent.

```
%pos = needle:unicodePositionIn(haystack [, Start=spos])
%pos = haystack:unicodePositionOf(needle [, Start=spos])
```

#### UnicodePositionIn and UnicodePositionOf syntax

The starting position must be a positive number. A zero or negative number results in request cancellation. If the starting position is greater than the length of the *haystack* plus one minus the length of the *needle*, a zero will always be returned since there aren't enough characters in the *haystack* to satisfy the search.

The UnicodePositionOf and UnicodePositionIn methods do exactly the same thing. The only difference between them is that in UnicodePositionOf, the *haystack* is the method object and the *needle* is the first argument. In UnicodePositionIn, the method object and first argument are reversed. Which method is preferable will depend on the application, and, in many cases, it will be quite arbitrary which one is used.

The `UnicodePositionOf` and `UnicodePositionIn` methods are analogous to the `String` intrinsic `PositionIn` and `PositionOf` methods.

The following example fragment contains three calls to `UnicodePositionOf`:

```
%s is unicode initial('This is a test')
printText {%s:unicodePositionOf('is')}
printText {%s:unicodePositionOf('is', start=4)}
printText {%s:unicodePositionOf('is', start=7)}
```

The result is:

```
3
6
0
```

The following fragment contains three calls to `UnicodePositionIn`:

```
%s is unicode initial('spl')
printText {%s:unicodePositionIn('splish splash')}
printText {%s:unicodePositionIn('splish splash', start=4)}
printText {%s:unicodePositionIn('splish splash', start=9)}
```

The result is:

```
1
8
0
```

### 4.2.5 **UnicodeRight function**

This function returns a specified number of the right-most characters of the method object string, possibly padding them on the left.

```
%outUni = unicode:unicodeRight(length [, Pad=char])
```

#### **UnicodeRight syntax**

The *length* value must be a non-negative number. A negative number results in request cancellation.

The *pad* parameter value must be either null or a single character. A longer value results in a compilation error. The default is a blank.

The `Right` method can be useful for right-justifying a value in a string, most typically by using a blank as *pad* character.

The UnicodeRight method is analogous to the String intrinsic Right method.

The following example statement places the right-most 5 characters of %x into %y. If %x is shorter than 6 characters, then all of %x is copied into %y:

```
%y = %x:unicodeRight(5)
```

The following request right-aligns a text string and a number, padding each on the left with a different character. This example also shows that the argument for the method's Pad parameter may be a Unicode string (which gets converted implicitly to String).

```
Begin
%u2 unicode initial('inaugural')
%pad is string len 4
%len is float
printText {%u2:unicodeRight(10, pad=' ')}
printText {123:unicodeRight(10, pad='&#x40;':U)}
End
```

The result is:

```
    inaugural
@@@@@@123
```

The U constant function used in the example is new as of *Sirius Mods* version 7.3, and it is described in the &JANSOAPR..

## 4.2.6 UnicodeSubstring function

This function returns a specific number of characters starting at a specific position in the method object Unicode string, possibly padding it on the right.

```
%outUni = unicode:unicodeSubstring(start, length [, Pad=char])
```

### UnicodeSubstring syntax

The length must be a non-negative number. A negative number results in request cancellation.

The pad parameter must be either null or a single character. A longer value results in a compilation error.

If the starting position is known to be 1, it might be tidier and slightly more efficient to use the UnicodeLeft method instead of UnicodeSubstring.

The UnicodeSubstring method is analogous to the String intrinsic Substring method.

The following example statement places into %y the 5 characters of %x starting at the third character. If %x is shorter than 7 characters, then all of %x is copied into %y:

```
%y = %x:unicodeSubstring(3, 5)
```

The following statement displays able!!!:

```
printText {'permeable':unicodeSubstring(6, 6, pad='!')}
```

## 4.3 Intrinsic conversion methods

Two new pairs of intrinsic conversion methods are added.

### 4.3.1 FloatToBinary and BinaryToFloat

The FloatToBinary function in the Float intrinsic class converts a numeric value to the binary string equivalent of its floating point representation. Since all Float types are processed in *Model 204* as Float Len 8, FloatToBinary always returns an 8-byte string.

The BinaryToFloat function in the String intrinsic class converts a binary representation of a floating point number to that number. The method object can be a string of length 4, 8, or 16.

The following program and its result demonstrate the new methods.

```
begin
  image silly
    a  is float len 4
    b  is float len 8
    c  is float len 16
  end image
  prepare image silly
  %i is float
  %s is string len 30

  for %i from -3.3 to 3.3 by 1.1
    printText {%i:right(5)}: {~} = {%i:floatToBinary:stringToHex}
    %silly:a = %i; %silly:b = %i; %silly:c = %i;
    %s = $!str_get_image('SILLY')
    printText {'':right(5)} {~} = {%s:stringToHex}
    printText {'':right(5)} {~} = {%s:substring(1, 4):binaryToFloat} -
    printText {'':right(5)} {~} = {%s:substring(5, 8):binaryToFloat} -
    printText {'':right(5)} {~} = {%s:substring(13, 16):binaryToFloat} -
  end for
end
```

The result is:

```
-3.3: %i:floatToBinary:stringToHex = C134CCCCCCCCCCCC
      %s:stringToHex =
C134CCDC134CCCCCCCCCCCCC134CCCCCCCCCCCC0000000000000000
      %s:substring(1, 4):binaryToFloat = -3.30000019073486
      %s:substring(5, 8):binaryToFloat = -3.3
      %s:substring(13, 16):binaryToFloat = -3.3
-2.2: %i:floatToBinary:stringToHex = C123333333333333
      %s:stringToHex =
C1233333C123333333333333C123333333333330000000000000000
      %s:substring(1, 4):binaryToFloat = -2.1999980926514
      %s:substring(5, 8):binaryToFloat = -2.2
      %s:substring(13, 16):binaryToFloat = -2.2
-1.1: %i:floatToBinary:stringToHex = C111999999999999
      %s:stringToHex =
C111999AC111999999999999C111999999999990000000000000000
      %s:substring(1, 4):binaryToFloat = -1.10000038146973
      %s:substring(5, 8):binaryToFloat = -1.1
      %s:substring(13, 16):binaryToFloat = -1.1
  0: %i:floatToBinary:stringToHex = 0000000000000000
     %s:stringToHex =
00000000000000000000000000000000000000000000000000000
     %s:substring(1, 4):binaryToFloat = 0
     %s:substring(5, 8):binaryToFloat = 0
     %s:substring(13, 16):binaryToFloat = 0
  1.1: %i:floatToBinary:stringToHex = 411199999999999
      %s:stringToHex =
4111999A4111999999999999411199999999990000000000000000
      %s:substring(1, 4):binaryToFloat = 1.10000038146973
      %s:substring(5, 8):binaryToFloat = 1.1
      %s:substring(13, 16):binaryToFloat = 1.1
  2.2: %i:floatToBinary:stringToHex = 4123333333333333
      %s:stringToHex =
4123333341233333333333334123333333333330000000000000000
      %s:substring(1, 4):binaryToFloat = 2.1999980926514
      %s:substring(5, 8):binaryToFloat = 2.2
      %s:substring(13, 16):binaryToFloat = 2.2
  3.3: %i:floatToBinary:stringToHex = 4134CCCCCCCCCCCC
      %s:stringToHex =
4134CCCD4134CCCCCCCCCCCC4134CCCCCCCCCCCC0000000000000000
      %s:substring(1, 4):binaryToFloat = 3.30000019073486
      %s:substring(5, 8):binaryToFloat = 3.3
      %s:substring(13, 16):binaryToFloat = 3.3
```

### 4.3.2 IntegerToHex and HexToInteger

These methods convert between hex strings and integers.

### 4.3.2.1 IntegerToHex

IntegerToHex converts an integer to its hexadecimal string representation.

The method's arguments are the requested hex-byte output size, and an optional, boolean, named-parameter: *Signed*

```
%str = number:IntegerToHex(num [, Signed=bool])
```

#### IntegerToHex syntax

Valid output sizes range from 1 to 4 bytes. Only integers that convert to four hex bytes or less (so a maximum output string length of eight) are allowed.

If the value of the *Signed* parameter is *True*, signed output is returned. The default is *False*.

Several examples follow.

- The following statement displays *00000000*:

```
printText {0:integerToHex(4)}
```

- The following statement displays *0000000A*:

```
printText {10:integerToHex(4)}
```

- The following statement displays *FF*:

```
printText {255:integerToHex(1)}
```

- The following statement displays *FF*:

```
printText {-1:integerToHex(1, signed=true)}
```

### 4.3.2.2 HexToInteger

HexToInteger returns the integer value of a hex encoded string. It has a single, optional, boolean named-parameter: *Signed*.

```
%num = string:hexToInteger([Signed=bool])
```

#### HexToInteger syntax

If the value of the *Signed* parameter is *True*, the given hex string is treated as signed data, and signed output is returned. The default is *False*.

The `HexToInteger` function requires a method object string with an even number of characters (hex digits), and it throws an `InvalidHexDat` exception, if the method object string does not contain a properly encoded hexadecimal value.

Some examples follow:

- The following statement displays 255:

```
printText {'FF':hexToInteger}
```

- The following statement displays -1:

```
printText {'FF':hexToInteger(signed=true)}
```

## 4.4 New `EbcdicToUnicode` parameter: `Untranslatable`

The `EbcdicToUnicode` method now takes the `Untranslatable` named argument, which lets you specify how to handle EBCDIC input characters that are not translatable to Unicode.

The `Untranslatable` argument is optional; if it is omitted and an EBCDIC character is encountered that is not translatable to Unicode, a `CharacterTranslationException` exception is thrown.

Otherwise, the value of the `Untranslatable` argument can be the null string or can be a single Unicode character.

- If it is the null string, any untranslatable EBCDIC characters are removed from the input string. The `EbcdicRemoveNonUnicode` method which formerly achieved this purpose has now been removed.
- If it is a single Unicode character, any untranslatable EBCDIC characters are replaced with that Unicode character. The `EbcdicTranslateNonUnicode` method which formerly achieved this purpose has now been removed.

As noted in “[EbcdicRemoveNonUnicode and EbcdicTranslateNonUnicode methods removed](#)” on page 25, removing the two methods mentioned above introduces a backwards compatibility issue.

For example, the following statement converts an EBCDIC string to Unicode, removing any EBCDIC characters that are not translatable to Unicode:

```
%u = %e:EbcdicToUnicode(Untranslatable='')
```

The following statement converts an EBCDIC string to Unicode, replacing with a question mark any EBCDIC characters that are not translatable to Unicode:

```
%u = %e:EbcDicToUnicode(Untranslatable='?')
```

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

## 5.1 Support 1.1 as XML version

The value of an `XmlDoc`'s `Version` property may now be set to '1.1':

```
%doc:Version = '1.1'
```

(The above change was introduced as part of maintenance in version 7.3 of the *Sirius Mods*.)

Also, 1.1 is accepted in the XML declaration in a deserialization operation:

```
%doc:LoadXml('<?xml version="1.1"?><docFoo/>')
```

## 5.2 The ClientCertificate method

This new shared method is a constructor that produces an `XmlDoc` object that contains detailed information from a client certificate received by a Janus web server, server socket, or Telnet server. The `XmlDoc` that is returned includes details about the client and about the certificate's signer.

The `ClientCertificate` method provides much of the functionality of functions in *Janus Web Server* (`$Web_Cert_Info` and `$Web_Cert_Levels`) and *Janus Sockets* (`$Sock_Cert_Info` and `$Sock_Cert_Levels`).

The `ClientCertificate` syntax is:

```
%doc = [% (XmlDoc):]ClientCertificate (           -
                                     [AttributeNames=bool] -
                                     [, AttributeValues=bool] )
```

The `AttributeNames` and `AttributeValues` arguments indicate how to display certificate detail names and values within their XML document elements: as the element name, or as the value of a "name" or "value" attribute. For example, for the certificate detail named "locality":

- Element-name format is:

```
<locality>Cambridge</locality>
```

- Attribute format is:

```
<info name="locality" value="Cambridge"/>
```

The default value for both is `False`, which produces element-name format.

The `XmlDoc` object produced by the `ClientCertificate` method is `Null` if:

- The method is invoked in a scenario where there is no client certificate.
- The method is invoked when you are not logged in on a Janus server port.
- The Janus port is not defined to use SSL.
- The client did not provide a certificate.

The information returned by this method is most useful in the server port's JANUS DEFINE NEWSDESCMD processing. The information method can be used anywhere: no information in the client certificate is considered "secure."

The following request prints the contents of a client certificate:

```
Begin
%doc Object XmlDoc
%doc = ClientCertificate
%doc:Print
End
```

A sample result follows:

```
<subject>
  <commonName>USER</commonName>
  <organizationalUnit>Local Services</organizationalUnit>
  <locality>Cambridge</locality>
  <state>MA</state>
  <country>US</country>
  <validityDate>20090125050000</validityDate>
  <validityTime>20090125050000</validityTime>
  <expirationDate>20100125050000</expirationDate>
  <expirationTime>20100125050000</expirationTime>
  <serialNumber>016437FF8A</serialNumber>
  <privateKeyLength>1024</privateKeyLength>
  <md5hash>7291D676CF47AF9578C94EEF029FA7BF</md5hash>
  <shaHash>8C969F3CCD1997082924EFD386B144917EF07F24</shaHash>
  <permanentKeyLength>1024</permanentKeyLength>
  <temporaryKeyLength>0</temporaryKeyLength>
  <issuer>
    <commonName>sis.sirius-software.com</commonName>
    <organizationalUnit>Local-Systems</organizationalUnit>
    <locality>Cambridge</locality>
```

```

<state>Massachusetts</state>
<country>USA</country>
<validityDate>20081210123238</validityDate>
<validityTime>20081210123238</validityTime>
<expirationDate>20110907123238</expirationDate>
<expirationTime>20110907123238</expirationTime>
<serialNumber>0160B99097</serialNumber>
<privateKeyLength>512</privateKeyLength>
<md5hash>681A76D6208C594D5BB08E816324E92B</md5hash>
<shaHash>513085FC5F652EDAC8759C4921842FF8B38ABEA2</shaHash>
<permanentKeyLength>512</permanentKeyLength>
<temporaryKeyLength>0</temporaryKeyLength>
<issuer>
  <commonName>sis.sirius-software.com</commonName>
  <organizationalUnit>Local-Systems</organizationalUnit>
  <locality>Cambridge</locality>
  <state>Massachusetts</state>
  <country>USA</country>
  <validityDate>20081210123238</validityDate>
  <validityTime>20081210123238</validityTime>
  <expirationDate>20110907123238</expirationDate>
  <expirationTime>20110907123238</expirationTime>
  <serialNumber>0160B99097</serialNumber>
  <privateKeyLength>512</privateKeyLength>
  <md5hash>681A76D6208C594D5BB08E816324E92B</md5hash>
  <shaHash>513085FC5F652EDAC8759C4921842FF8B38ABEA2</shaHash>
  <permanentKeyLength>512</permanentKeyLength>
  <temporaryKeyLength>0</temporaryKeyLength>
</issuer>
</issuer>
</subject>

```

### 5.3 CrPreserve deserialization option

The following option is available for all deserialization methods:

**CrPreserve** All whitespace characters in Element content are preserved, including carriage return. Unlike all other deserialization options, a carriage return in Element content does **not** undergo the normalization specified in the XML standard.

CrPreserve is mutually exclusive with all other Wsp\* options, and with the LinefeedNoTrailingTabs option.

The CrPreserve option was also implemented with a maintenance zap to version 7.4 of the *Sirius Mods*.



The following features are new or changed in *Janus Web Server*.

## 6.1 TextUtf8 option on \$Web\_\*\_Content functions

A new option has been added to the `$Web_File_Content`, `$Web_Input_Content`, and `$Web_Output_Content` functions. The name of this option is `TextUtf8` and it is mutually exclusive with the `Text` and `Binary` options.

The `TextUtf8` option causes `$Web_Input_Content` to first convert each two-byte UTF-8 sequence to the corresponding ASCII character, which is then translated to EBCDIC using the translation table in effect for the *Janus Web Server* connection.

For example:

```
%string = $Web_Input_Content('TextUtf8')
```

This feature was also provided as part of maintenance to version 7.3 of *Janus Web Server*.



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 immediately prior version (7.4).

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 SOAP ULI

#### 7.1.1.1 **EbcdicRemoveNonUnicode and EbcdicTranslateNonUnicode methods removed**

The following methods have been removed, and, as discussed in “[New EbcdicToUnicode parameter: Untranslatable](#)” on page 17, their purpose can now be achieved using the `Untranslatable` named argument of the `EbcdicToUnicode` function:

##### **EbcdicRemoveNonUnicode**

Removing untranslatable EBCDIC characters can be achieved with `EbcdicToUnicode(Untranslatable='')`.

##### **EbcdicTranslateNonUnicode**

Translating untranslatable EBCDIC characters (for example, to a question mark) can be achieved with `EbcdicToUnicode(Untranslatable='?')`.

#### 7.1.1.2 **ToXmlDoc default changed to AttributeNames=True**

The default value of the `AttributeNames` argument of the `ToXmlDoc` method in the `Record` class has been changed.

Formerly, the default was `False`; the default now is `True`.

The reason for this change is that the `AttributeNames=True` format is more well suited to operations on the `XmlDoc`, particularly a record copying operation.

This change was actually introduced as part of maintenance in the 7.5 *Sirius Mods*. It was also introduced as part of maintenance in the 7.3 and 7.4 *Sirius Mods*.

## **7.2 Fixes in Sirius Mods 7.5 but not in 7.4**

This section lists fixes to functionality existing in the *Sirius Mods* version 7.4 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* 7.5 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 7.5 of *Sirius Mods*.

- There are no corequisites associated with *Sirius Mods* 7.5.