MICRO FOCUS

Orbacus Version 4.3.5

Release Notes

Micro Focus
The Lawn
22-30 Old Bath Road
Newbury, Berkshire RG14 1QN
UK

http://www.microfocus.com

2016-09-28

Contents

Orbacus 4.3.5 Release Notes

Orbacus 4.3.5 is a service pack release of Orbacus 4.3 from Micro Focus.

These release notes contain information about the Orbacus 4.3.5 release. They contain information that might not appear elsewhere in the documentation. Read them in their entirety before you install the product.

Product Components

Orbacus 4.3.5 includes the following new product component versions:

- Orbacus Java 4.3.5
- Orbacus C++ 4.3.5
- Orbacus Notify Java 2.2
- Orbacus Notify C++ 2.2
- Orbacus FSSL Java 4.3.5
- Orbacus FSSL C++ 4.3.5

Migrating and Upgrading

In order to run existing applications against an Orbacus 4.3.5 C++ installation:

- If the application was built statically, you need to re-link the application.
- If the application was built with shared libraries, then name changes resulting from version number changes to the OpenSSL shared libraries as well as the Orbacus shared libraries mean that a re-link will be required.

New Features

New Features in Orbacus FSSL

The following new features are introduced for Orbacus FSSL 4.3.5:

- The Orbacus FSSL version number is now in line with the Orbacus version: 4.3.5.
- Support for the OpenSSL version 1.0.2j security toolkit has been added.
- Support for the latest Java 8 JDK JSSE security toolkit has been added.

- The following new secure communications protocols can be used when installing the client side FSSL plug-in:
  - TLS v1.0
  - TLS v1.1
  - TLS v1.2

- It is now possible to specify via a configuration file which communications protocols are to be supported. The newly-supported protocols listed above and the already-supported SSL v3 can be specified.

  For example, to specify support for the TLS protocols using the configuration file:

```
ooc.oci.client=fssliop --secure_protocol "TLSv1,TLSv1.1,TLSv1.2"
```

  Note that since servers must also install the client side plug-in, both clients and servers can be configured this way.

  For an example of using `--secure_protocol` see the files `fssl/demo/hello2_rsa/client.cfg` and `fssl/demo/hello2_rsa/server.cfg` provided with the installation.

- The FSSL module **Manager** interface has been extended with the **Manager2** interface. This new interface enables you to create contexts programmatically, while specifying the secure protocols to use, as follows:

```
ContextID create_context_with_protocols(in CertificateSeq chain,
                                        in OctetSeq pkey,
                                        in PassPhrase pass,
                                        in TrustDecider decider,
                                        in CipherSeq ciphers,
                                        in SecureProtocolSeq protocols)
     raises (BadCertificate, BadKey, BadCipher);

ContextID create_pkcs12_context_with_protocols(in OctetSeq cert
                                               in PassPhrase pass,
                                               in TrustDecider decider,
                                               in CipherSeq ciphers,
                                               in SecureProtocolSeq protocols)
     raises (BadCertificate, BadKey, BadCipher);

 ContextID create_anon_context_with_protocols(in CipherSeq ciphers,
                                              in SecureProtocolSeq protocols)
     raises (BadCipher);
```

  For a C++ example of using these interfaces see the file `fssl/test/TestContext.cpp` provided.

  For a Java example of using these interfaces see the file `fssl/test/TestContext.java` provided.

- The FSSL module **Current** interface has been extended with the **Current2** interface. This new interface provides the following methods to obtain the negotiated secure protocol:

```
SecureProtocol get_peer_protocol()
     raises(NoContext, NoPeer);
```

  For a C++ example of using this interface see the file `fssl/demo/hello2_rsa/Hello_impl.cpp` provided.

  For a Java example of using this interface see the file `fssl/demo/hello2_rsa/Hello_impl.java` provided.

- The `OCI::FSSLIOP` module **TransportInfo** interface has been extended with the **TransportInfo2** interface. The new interface provides the following attribute which holds the negotiated secure protocol:

```
readonly attribute FSSL::SecureProtocol negotiated_protocol;
```

  For a C++ example of using this interface see the file `fssl/demo/hello2_rsa/Client.cpp` provided.

  For a Java example of using this interface see the file `fssl/demo/hello2_rsa/Client.java` provided.

- The C++ helper methods that return sequences of cipher suite identifiers have been updated so that they no longer return identifiers that are not supported by OpenSSL.

  - Helper `FSSL::get_non_export_ciphers()` no longer returns these identifiers:

    `DHE_RSA_WITH_DES_CBC_SHA`

    `DHE_DSS_WITH_DES_CBC_SHA`

  - Helper `FSSL::get_export_ciphers()` no longer returns these identifiers:

    `RSA_EXPORT_WITH_RC4_40_MD5`

    `RSA_EXPORT_WITH_RC2_CBC_40_MD5`

    `RSA_EXPORT_WITH_DES40_CBC_SHA`

    `DHE_RSA_EXPORT_WITH_DES40_CBC_SHA`

    `DHE_DSS_EXPORT_WITH_DES40_CBC_SHA`

  - Helper `FSSL::get_RSA_ciphers()` no longer returns these identifiers:

    `RSA_EXPORT_WITH_RC4_40_MD5`

    `RSA_EXPORT_WITH_RC2_CBC_40_MD5`

    `RSA_EXPORT_WITH_DES40_CBC_SHA`

    `RSA_WITH_DES_CBC_SHA`

    `DHE_RSA_EXPORT_WITH_DES40_CBC_SHA`

    `DHE_RSA_WITH_DES_CBC_SHA`

  - Helper `FSSL::get_DSS_ciphers()` no longer returns these identifiers:

DHE_DSS_EXPORT_WITH_DES40_CBC_SHA

DHE_DSS_WITH_DES_CBC_SHA

- Helper `FSSL_get_ADH_ciphers()` no longer returns these identifiers:

  DH_anon_EXPORT_WITH_RC4_40_MD5

  DH_anon_EXPORT_WITH_DES40_CBC_SHA

  DH_anon_WITH_DES_CBC_SHA

- The Java helper methods that return sequences of cipher suite identifiers have been updated so that they no longer return identifiers that are not supported by Java.

  - Helper `com.ooc.FSSL.get_non_export_ciphers()` no longer returns these identifiers:

    com.ooc.FSSL.RSA_WITH_IDEA_CBC_SHA.value

  - Helper `com.ooc.FSSL.get_export_ciphers()` no longer returns these identifiers:

    com.ooc.FSSL.RSA_EXPORT_WITH_RC2_CBC_40_MD5.value

  - Helper `com.ooc.FSSL.get_RSA_ciphers()` no longer returns these identifiers:

    com.ooc.FSSL.RSA_EXPORT_WITH_RC2_CBC_40_MD5.value
    com.ooc.FSSL.RSA_WITH_IDEA_CBC_SHA.value

- The following helper methods have been added for protocol processing.

  C++

  - `FSSL::SecureProtocolSeq* FSSL::getTLSProtocols()`
    Return a sequence containing the TLS protocols.
  - `FSSL::SecureProtocolSeq* FSSL::getTLSV1_2_protocol()`
    Return a sequence containing the TLSv1.2 protocol.
  - `FSSL::SecureProtocolSeq* FSSL::getTLSV1_1_protocol()`
    Return a sequence containing the TLSv1.1 protocol.
  - `FSSL::SecureProtocolSeq* FSSL::getTLSV1_protocol()`
    Return a sequence containing the TLSv1 protocol.
  - `FSSL::SecureProtocolSeq* FSSL::getSSLV3_protocol()`
    Return a sequence containing the SSLv3 protocol.
  - `char* FSSL::Protocol_to_string(SecureProtocol protocol)`
    Convert a protocol to a string.

  Java

  - `static public int[]com.ooc.FSSL.getTLSProtocols()`
    Return a sequence containing the TLS protocols.
  - `static public int[]com.ooc.FSSL.getTLSV1_2Protocol()`
    Return a sequence containing the TLSv1.2 protocol.
  - `static public int[]com.ooc.FSSL.getTLSV1_1Protocol()`
    Return a sequence containing the TLSv1.1 protocol.
  - `static public int[]com.ooc.FSSL.getTLSV1Protocol()`
    Return a sequence containing the TLSv1 protocol.

- `static public int[]com.ooc.FSSL.getSSLV3Protocol()`
  Return a sequence containing the SSLv3 protocol
- `static public String com.ooc.FSSL.Protocol_to_string(int protocol)`
  Convert a protocol to a string.

- Support has been added for the following new cipher suites:
  - RSA_WITH_AES_128_GCM_SHA256
  - RSA_WITH_AES_128_CBC_SHA256
  - RSA_WITH_AES_128_CBC_SHA
  - RSA_WITH_AES_256_GCM_SHA384
  - RSA_WITH_AES_256_CBC_SHA256
  - RSA_WITH_AES_256_CBC_SHA
  - DHE_DSS_WITH_AES_128_GCM_SHA256
  - DHE_DSS_WITH_AES_128_CBC_SHA256
  - DHE_DSS_WITH_AES_128_CBC_SHA
  - DHE_DSS_WITH_AES_256_GCM_SHA384
  - DHE_DSS_WITH_AES_256_CBC_SHA256
  - DHE_DSS_WITH_AES_256_CBC_SHA
  - DHE_RSA_WITH_AES_128_GCM_SHA256
  - DHE_RSA_WITH_AES_128_CBC_SHA256
  - DHE_RSA_WITH_AES_128_CBC_SHA
  - DHE_RSA_WITH_AES_256_GCM_SHA384
  - DHE_RSA_WITH_AES_256_CBC_SHA256
  - DHE_RSA_WITH_AES_256_CBC_SHA
  - ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
  - ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
  - ECDHE_ECDSA_WITH_AES_128_CBC_SHA
  - ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
  - ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
  - ECDHE_ECDSA_WITH_AES_256_CBC_SHA
  - ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
  - ECDHE_ECDSA_WITH_RC4_128_SHA
  - ECDHE_RSA_WITH_AES_128_GCM_SHA256
  - ECDHE_RSA_WITH_AES_128_CBC_SHA256
  - ECDHE_RSA_WITH_AES_128_CBC_SHA
  - ECDHE_RSA_WITH_AES_256_GCM_SHA384
  - ECDHE_RSA_WITH_AES_256_CBC_SHA384
  - ECDHE_RSA_WITH_AES_256_CBC_SHA
  - ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
  - ECDHE_RSA_WITH_RC4_128_SHA

- The default backend for Orbacus FSSL Java is now `com.ooc.FSSL.jsse.FSSLImpl`. Support for all other backends has been discontinued.

New Features in Orbacus Notify

- The persistence database has been upgraded to Berkeley DB version 4.5.20.

Supported Platforms

The following platforms are supported in Orbacus 4.3.5:

- Solaris 10 x86_64 with Oracle Studio 12 Update 3 (32-bit and 64-bit)
- Solaris 11 x86_64 with Oracle Studio 12 Update 3 (32-bit and 64-bit)
- Solaris 10 SPARC with Oracle Studio 12 Update 3 (32-bit and 64-bit)
- Solaris 11 SPARC with Oracle Studio 12 Update 3 (32-bit and 64-bit)
- RedHat Enterprise Linux 7.x with GCC 4.8 (32-bit and 64-bit)
- Windows 7 with Microsoft Visual Studio 2015 (32-bit and 64-bit)
- Windows 10 with Microsoft Visual Studio 2015 (32-bit and 64-bit)
- Windows Server 2012 R2 with Microsoft Visual Studio 2015 (32-bit and 64-bit)

The following JDK is supported in Orbacus 4.3.5 for Java:

- Java 8

For the latest information on supported platforms, compilers, and Java versions, see the Product Availability page.

Discontinued Platforms

Orbacus FSSL Java no longer supports the IAIK toolkit.

Known Issues

The following are known issues in Orbacus 4.3.5.

Known Issues in Orbacus Java

- You cannot use the Orbacus IMR Console to modify an existing service or POA. You must make any modification using the command-line interface. The change will be visible in the IMR Console once it has been refreshed.

  (RPI 622953)

Resolved Issues

The reported issues resolved in Orbacus 4.3.5 are listed in this section. The numbers that follow each issue are the Reported Problem Incident number followed by the Customer Incident Numbers (in parentheses). RPIs that have numbers only (and no text) are included to confirm that the RPIs have been fixed, since no further information is required.The following issues have been fixed in Orbacus 4.3.5:

Resolved Issues in Orbacus C++

- Orbacus was unresponsive if `gethostbyname` could not resolve the host.

  (OB-91)
- It was possible for requests to overlap in multi-threaded clients.

  (OB-92)
- The idl command was crashing and core dumping.

  (OB-93)
- IMR was not able to launch processes if the client is sending requests with `enableLocateRequest=true`.

  (OB-95)
- Multiple threads in a client received a timeout at the same time when invoking on the same endpoint.

  (OB-96)
- Incorrect exception handling in the method `GIOPConnectionThreaded::sendReceive` could cause a crash.

  (OB-97)
- Building Orbacus with Active Perl 5.14.* did not work.

  (OB-98)
- There was a problem with the `InputStreamImpl` class, when using GCC optimization and strict aliasing.

  (OB-99)
- Orbacus 4.3.5 has been certified for Solaris SPARC 11.

  The configure menu is updated.

  A workaround has been added for a `pthread_setschedparam()` EPERM error which could happen when running in a Solaris non-global zone.

  Careful setting of minimum and maximum values in `findUnusedDiscriminator()` of `DynamicAny_impl.cpp`.

  An updated UDP file transfer demo means that a statically

linked sender/receiver will take less time to run.

(RPI 1092784)

Resolved Issues in Orbacus Notify C++

- Orbacus Notify has been certified for Solaris SPARC 11.

  The configure menu is updated.

  Building Orbacus Notify with Active Perl 5.14.* does not work.

  (RPI 1092784)

Resolved Issues in Orbacus FSSL C++

- FSSL multi-threaded server crashes on Windows during ORB shutdown.

  (RPI 622951)
- Orbacus FSSL 4.3.5 has been certified for Solaris SPARC 11.

  The configure menu is updated.

  Building Orbacus FSSL with Active Perl 5.14.* does not work.

  Updated expired demo and test certificates.

  Updated the Orbacus FSSL test.

  Fixed a compile error when building static libraries in the secure bank demo server.

  (RPI 1092784)

Resolved Issues in Orbacus Java

- Orbacus Java 4.3.5 has been certified for Windows Server 2012. JDK 1.7 is supported.

  Tests can be run under Cygwin.

  (RPI 1092784)

Resolved Issues in Orbacus Notify Java

- Orbacus Notify Java has been certified for Windows Server 2012. JDK 1.7 is supported.

  A problem in the simple demo structured pull consumer has been fixed.

  A null pointer exception problem with the Notify console has

been fixed.

(RPI 1092784)

- Icons were not displaying correctly in the Orbacus Notify Java console.

  (RPI 622767)

Resolved Issues in Orbacus FSSL Java

- Orbacus FSSL Java has been certified for Windows Server 2012. JDK 1.7 is supported.

  Updated expired demo and test certificates.

  Updated the Orbacus FSSL test to better handle a JSSE backend.

  Updated the demos to use a JSSE backend.

  Added a fix for server endpoints that support both insecure and secure connections.

  Run the install target from the "all" target.

  (RPI 1092784)

- The FSSL Java test was throwing a Null Pointer Exception

  (RPI 603156)