MICRO FOCUS

CORBA

An OMG Standard

Discover the Future of CORBA

Orbacus
Version 4.3.6

Release Notes

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

http://www.microfocus.com

2021-12-16

Orbacus 4.3.6 Release Notes 1

Product Components 1

Migrating and Upgrading 1

New Features 1

New Features in Orbacus 1

New Features in Orbacus FSSL 2

New Features in Orbacus Notify 5

Supported Platforms 5

Discontinued Platforms 5

Known Issues 5

Known Issues in Orbacus FSSL C++ 5

Known Issues in Orbacus FSSL Java 6

Resolved Issues 6

Orbacus 4.3.6 Release Notes

Orbacus 4.3.6 is a service pack release of Orbacus 4.3 from Micro Focus.

These release notes contain information about the Orbacus 4.3.6 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 4.3.6 includes the following new product component versions:

- Orbacus Java 4.3.6
- Orbacus C++ 4.3.6
- Orbacus Notify Java 2.3.0
- Orbacus Notify C++ 2.3.0
- Orbacus FSSL Java 4.3.6
- Orbacus FSSL C++ 4.3.6

Migrating and Upgrading

In order to run existing applications against an Orbacus 4.3.6 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

The following new features are introduced for Orbacus 4.3.6:

- Orbacus C++ support for the GCC 8 C++ compiler

  Orbacus C++ can be built using the GCC 8 compiler.

  C++ 11 deprecates the use of exception specifications. C++ 17 does not support the use of exception specifications other than throw() and noexcept.

  All exception specifications, with the exception of throw(), have been removed from function declarations and definitions in the Orbacus C++ code base. Matching this, the Orbacus C++ idl compiler no longer generates exception specifications (other than the possible use of throw()) in header files, skeletons and stubs. This positions Orbacus to be built according to C++ 17.

Please consider how this may impact your current applications.

- Support for Java 11

  Orbacus Java supports Java 11:

  - The code base can be built with an earlier version of Java, and run in a Java 11 JVM.
  - The code base can be built with Java 11, and run in a Java 11 JVM.

New Features in Orbacus FSSL

The following new features are introduced for Orbacus FSSL 4.3.6:

- Support for Java 11.

  Orbacus FSSL Java supports Java 11. Please see the section "New Features in Orbacus" for more details.

- Orbacus FSSL C++ support for the GCC 8 C++ compiler.

  Orbacus FSSL C++ can be built using the GCC 8 compiler. Please see the section "New Features in Orbacus" for more details.

- Support for the OpenSSL version 1.1.1k security toolkit has been added.
- Support for the latest Java 11 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 v1.3

  Considerations when using TLS v1.3.

  When creating a context that does not specify secure protocols in the API call, and secure protocols are not specified in a configuration file, TLS v1.0, TLS v1.1, and TLS v1.2 will be set as default secure protocols. To create a context that supports TLS v1.3:

  - Explicitly set TLS v1.3 as a secure protocol in the API call, or in the configuration file.
  - Supply a TLS v1.3 supported cipher suite (or set of cipher suites) in the API call.

  The secure protocol can be a list that includes TLS v1.3 and other secure protocols. The cipher suite list can include cipher suites supported by TLS v1.3, as well as cipher suites supported by other protocols.

  If a context is created and TLS v1.3 is one of the secure protocols, but a TLS v1.3 cipher suite was not set in the cipher suite list:

  - For FSSL C++ a set of default TLS v1.3 cipher suites will automatically be used by OpenSSL. A message will be issued when this occurs, and the context is created successfully.

    - For FSSL Java no default TLS v1.3 cipher suites will be set. A message will be issued when this occurs, and the context is created successfully. There is a risk of the handshake failing if TLS v1.3 is selected as the protocol and no TLS v1.3 cipher suite is available.

  If a context is created that does not specify TLS v1.3 as a secure protocol, but a TLS v1.3 cipher suite is part of the cipher suite list, a message will be issued, and the context is created successfully.

  Anonymous contexts cannot be created with TLS v1.3, as TLSv1.3 does not contain anonymous cipher suites.

  TLS v 1.3 has certificate requirements that must be met, otherwise the handshake will fail:

    - DSA certificates are not supported
    - RSA certificates require a minimum private key size of 2048 bits
    - ECC certificates require a minimum private key size of 256 bits
    - Older signature algorithms are not supported - newer algorithms such as SHA256 etc are recommended

- Demos updated to use TLS v1.3.

  For both FSSL C++ and FSSL Java, the following demos have been updated to use TLS v1.3:

    - hello
    - hello_orbix
    - hello_pkcs12
    - secure_bank

- 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:

      `RSA_WITH_IDEA_CBC_SHA`

      `DHE_RSA_WITH_3DES_EDE_CBC_SHA`

      `DHE_DSS_WITH_3DES_EDE_CBC_SHA`

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

      `RSA_WITH_IDEA_CBC_SHA`

      `DHE_RSA_WITH_3DES_EDE_CBC_SHA`

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

      `DHE_DSS_WITH_3DES_EDE_CBC_SHA`

- The following C++ helper methods have been added for cipher suite processing:

  `FSSL::get_TLSV1_3_ciphers()`

  Return a sequence containing the following cipher suites:

    - AES_128_GCM_SHA256
    - AES_256_GCM_SHA384
    - CHACHA20_POLY1305_SHA256
    - AES_128_CCM_SHA256

  - AES_128_CCM_8_SHA256
- The following Java helper methods have been added for cipher suite processing:

  `com.ooc.FSSL.getTLSV13Ciphers()`

  Return a sequence containing the following cipher suites:

  - AES_128_GCM_SHA256
  - AES_256_GCM_SHA384
- The following helper methods have been added or changed for protocol processing.

  C++

  - `FSSL::SecureProtocolSeq* FSSL::getTLSProtocols()`
    Return a sequence containing the TLS protocols, excluding TLSv1.3.
  - `FSSL::SecureProtocolSeq* FSSL::getTLSV1_3_protocol()`
    Return a sequence containing the TLSv1.3 protocol.
  - `FSSL::getLatestTLSProtocols()`
    Return a sequence containing the TLS protocols, including TLSv1.3.

  Java

  - `static public int[]com.ooc.FSSL.getTLSProtocols()`
    Return a sequence containing the TLS protocols, excluding TLSv1.3.
  - `static public int[]com.ooc.FSSL.getTLSV1_3Protocol()`
    Return a sequence containing the TLSv1.3 protocol.
  - `static public int[] getLatestTLSProtocols()`
    Return a sequence containing the TLS protocols, including TLSv1.3.
- Support has been added for the following new cipher suites:
  - AES_128_GCM_SHA256
  - AES_256_GCM_SHA384
  - CHACHA20_POLY1305_SHA256 (C++ only)
  - AES_128_CCM_SHA256 (C++ only)
  - AES_128_CCM_8_SHA256 (C++ only)
- FSSL C++ has dropped support for the following cipher suites:
  - DHE_DSS_WITH_3DES_EDE_CBC_SHA
  - DHE_RSA_WITH_3DES_EDE_CBC_SHA
  - RSA_WITH_IDEA_CBC_SHA

  Note that while the following cipher suites are still supported by FSSL C++, their use is discouraged as they require the "@SECLEVEL=0" string be added to the list of cipher suites provided to OpenSSL, which drops the OpenSSL security level to '0':

  - RSA_EXPORT_WITH_NULL_MD5
  - RSA_EXPORT_WITH_NULL_SHA
  - RSA_WITH_RC4_128_MD5
  - DH_anon_WITH_RC4_128_MD5
  - DH_anon_WITH_3DES_EDE_CBC_SHA

New Features in Orbacus Notify

The following new features are introduced for Orbacus Notify 4.3.6:

- Support for Java 11.

  Orbacus Notify Java supports Java 11. Please see the section "New Features in Orbacus" for more details.
- Orbacus Notify C++ support for the GCC 8 C++ compiler.

  Orbacus Notify C++ can be built using the GCC 8 compiler. Please see the section "New Features in Orbacus" for more details.

Supported Platforms

The following platforms are supported in Orbacus 4.3.6:

- RedHat Enterprise Linux 7.x with GCC 4.8 (32-bit and 64-bit)
- RedHat Enterprise Linux 8.x with GCC 8 (32-bit and 64-bit)
- CentOS 7 with GCC 4.8
- Ubuntu 18 with GCC 7.5
- Windows with Microsoft Visual Studio 2015 (32-bit and 64-bit)
- Windows with Microsoft Visual Studio 2017 (32-bit and 64-bit)

The following JDKs are supported in Orbacus 4.3.6 for Java:

- Java 7
- Java 8
- Java 11

Discontinued Platforms

- Orbacus FSSL Java no longer supports the IAIK toolkit.

Known Issues

The following are known issues in Orbacus 4.3.6.

Known Issues in Orbacus FSSL C++

- An FSSL C++ server does not receive the certificate chain from the FSSL Java client when mutual authentication has been requested. This occurs when the FSSL Java client and FSSL C++ server use different CAs. This impacts the FSSL C++ server's trust decider as it requires the peer's certificate chain in order to determine the outcome in the trust decider. The workaround is to use the same CA for both the client and server.

- Running the bank server demo with an FSSL C++ server and an Orbix 6 C++ client, using TLS 1.3, can hang the Orbix 6 C++ client when the FSSL C++ server trust decider decides not to trust the peer. The workarounds are:
  - Ensure the peers trust each other when using TLS 1.3.

- Use TLS 1.2 if there is a chance the FSSL C++ server will not trust the Orbix 6 C++ client.
- When converting a PEM formatted private key to a DER formatted private key, the "openssl pkcs8" command is used. When using the 1.1.1k version of the "openssl" command, the "-v1PBE-MD5-DES" operand must be used in order to create a key that is compatible with the Orbacus FSSL C++ code. For example:

```
openssl pkcs8 -outform DER -v1 PBE-MD5-DES -inform PEM
    -in client.key.pem -out client.key -topk8
```

Known Issues in Orbacus FSSL Java

- Some of the very latest JDKs now disable TLS 1.0 and TLS 1.1 by default. More details and affected versions can be seen at https://bugs.openjdk.java.net/browse/JDK-8202343.

  In summary, the default java.security file for these later versions now includes something like the following:

```
jdk.tls.disabledAlgorithms=SSLv3, TLSv1, TLSv1.1, DES, \
    DH keySize < 1024, RC4_40, 3DES_EDE_CBC, anon, NULL, \
    include jdk.disabled.namedCurves
```

  This disables TLS1.0 and TLS1.1 by default. If you want to use these protocols you will need to remove them from this disabled list.

Resolved Issues

The reported issues resolved in Orbacus 4.3.6 are listed in this section. The numbers that follow each issue are the Reported Problem Incident number (RPI) or an Octane defect number followed by the Customer Incident Numbers (where applicable) in parentheses. RPIs/Octane defects that have numbers only and no text are included to confirm that the RPIs/Octane defects have been fixed, since no further information is required. The following issues have been fixed in Orbacus 4.3.6:

Resolved Issues in Orbacus C++

- IMR Segmentation Fault

  RPI 1113872

Resolved Issues in Orbacus FSSL C++

- Support for Version 1.0 IORs

  175028

  Orbacus FSSL C++ supports a version 1.0 IOR that does not contain any component data.

Resolved Issues in Orbacus FSSL Java

- Support for Version 1.0 IORs

  175028

  Orbacus FSSL Java supports a version 1.0 IOR that does not contain any component data.