Progress®



New and Revised Features for OpenEdge 11.7.6

OpenEdge®

Copyright

© 2021 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

These materials and all Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. The information in these materials is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear therein. The references in these materials to specific platforms supported are subject to change.

DataDirect Cloud, DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, DataRPM, Defrag This, Deliver More Than Expected, DevReach (and design), Icenium, Inspec, Ipswitch, iMacros, Kendo UI, Kinvey, MessageWay, MOVEit, NativeChat, NativeScript, OpenEdge, Powered by Chef, Powered by Progress, Progress, Progress Software Developers Network, SequeLink, Sitefinity (and Design), Sitefinity, Sitefinity (and design), SpeedScript, Stylus Studio, Stylized Design (Arrow/3D Box logo), Styleized Design (C Chef logo), Stylized Design of Samurai, TeamPulse, Telerik, Telerik (and design), Test Studio, WebSpeed, WhatsConfigured, WhatsConnected, WhatsUp, and WS_FTP are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries.

Analytics360, AppServer, BusinessEdge, Chef Automate, Chef Compliance, Chef Desktop, Chef Habitat, Chef WorkStation, Corticon.js, Corticon Rules, Data Access, DataDirect Autonomous REST Connector, DataDirect Spy, DevCraft, Fiddler, Fiddler Everywhere, FiddlerCap, FiddlerCore, FiddlerScript, Hybrid Data Pipeline, iMail, JustAssembly, JustDecompile, JustMock, KendoReact, NativeScript Sidekick, OpenAccess, PASOE, Pro2, ProDataSet, Progress Results, Progress Software, ProVision, PSE Pro, Push Jobs, SafeSpaceVR, Sitefinity Cloud, Sitefinity CMS, Sitefinity Digital Experience Cloud, Sitefinity Feather, Sitefinity Insight, Sitefinity Thunder, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, Supermarket, SupportLink, Unite UX, and WebClient are trademarks or service marks of Progress Software Corporation and/or its subsidiaries or affiliates in the U.S. and other countries. Java is a registered trademark of Oracle and/or its affiliates. Any other marks contained herein may be trademarks of their respective owners.

May 2020

Last updated with new content: Release 11.7.6

Updated: 2021/06/28

Table of Contents

What's New in 11.7.6.....9

Support to run PAS for OpenEdge in Docker Containers.....9
Deferred logging in PAS for OpenEdge.....10
Updated Spring Security JARs.....11
Upgrade OpenEdge REST applications.....15
Upgrade OpenEdge SOAP applications.....15
Upgrade ABL applications 15
Updated GENSPRINGPWD utility for PAS for OpenEdge.....16
Updated EncodePassword utility for the classic AppServer.....16
Set hashed and salted passwords using EncodePassword.....16
Encrypt a password shared by REST Management Agent and REST web application.....18

Preface

Purpose

The New and Revised Features briefly describes both new features and changes to existing features introduced in OpenEdge® Release 11.7.6.

Audience

This guide is primarily intended for OpenEdge application developers and system administrators who are upgrading their OpenEdge license to Release 11.7.6.

Organization

What's New in 11.7.6

Presents descriptions of new and enhanced features introduced in Release 11.7.6.

Documentation conventions

See Documentation Conventions for an explanation of the terminology, format, and typographical conventions used throughout the OpenEdge content library.

1

What's New in 11.7.6

This topic describes the new features included in OpenEdge Release 11.7.6.

For details, see the following topics:

- Support to run PAS for OpenEdge in Docker Containers
- Deferred logging in PAS for OpenEdge
- Updated Spring Security JARs
- Updated GENSPRINGPWD utility for PAS for OpenEdge
- Updated EncodePassword utility for the classic AppServer

Support to run PAS for OpenEdge in Docker Containers

Progress Application Server (PAS) for OpenEdge Release 11.7.6 is available as a Docker image where applications can be developed, deployed, and run in an adaptive and isolated environment.

For more information, see Run PAS for OpenEdge in a Docker Container. The Docker container is available at the Docker Hub.

Deferred logging in PAS for OpenEdge

Deferred logging was added to OpenEdge 11.7.6.

Deferred logging in PAS for OpenEdge opens an in-memory buffer to record information immediately preceding a multi-session agent crash. It also lets system administrators run an on-demand monitoring check on the instance. The in-memory buffer is configured independently from the agent log and automatically writes to the agent log file in the event of a crash or when it is called through an API.

If a PAS for OpenEdge multi-session agent crashes with deferred logging enabled, the buffer memory is automatically printed to the abl-app-name.agent.datestamp.log file. Additionally, a system administrator or DevOps engineer can use a REST or JMX API to dump the in-memory buffer to the agent log file for close monitoring of a production system.

The following video introduces deferred logging in PAS for OpenEdge:

The deferred logging is set in the openedge.properties file.There are three properties that allow system administrators to control how much in-memory logging information is stored: `defrdLogNumLines`, `defrdLoggingLevel`, `defrdLogEntryTypes`. The deferred logging buffer is enabled if all three properties are set to a non-zero value.

Deferred logging keeps a separate memory buffer from the regular agent logging. Deferred logging can be set to record different log entry types and to record at different logging levels. A separate memory buffer can help keep the regular agent log to a minimal size, while providing another type of logging for reconstructing agent crashes or monitoring the application without affecting performance.

For example, if you ran 200 sessions on each agent and ran multiple agents, it would require thousands of buffer lines to capture all the debugging information. This circumstance would create a log file that would be difficult to parse and could affect performance. Deferred logging allows you to create a separate logging buffer that saves only the specified number of log number lines and log entry types. This gives a higher logging level only immediately preceding an agent crash or when it is invoked through the API. Deferred logging allows for only the context of the crash to display, while allowing the regular agent log to print other messages or at a lower logging level.

To enable deferred logging on your PAS for OpenEdge instance:

1. Use the OEPROP utility to specify the number of lines to be saved in the deferred logging buffer:

```
oeprop.{bat|sh} AppServer.SessMgr.defrdLogNumLines=500
```

   There are two types of log entries that can be written to the buffer: Unconditional entries, including start-up and error events, and conditional entries, based on the logging level and startup type. The unconditional entries use some lines specified with this property. The default value is 0, which disables the feature.

2. Use the OEPROP utility to specify the logging level of the deferred logging buffer:

```
oeprop.{bat|sh} AppServer.SessMgr.defrdLoggingLevel=4
```

   You could use the deferred logging buffer to act as a stack trace preceding crash events, or set it to record a separate log entry type from the agent log. If this property is set to 0, then no deferred log file buffer is written to the agent log file.

3. Use the OEPROP utility to specify deferred log entry type:

```
oeprop.{bat|sh} AppServer.SessMgr.defrdLogEntryTypes=4GLTrace
```

   The default is the `4GLTrace` log entry type. The following agent log entry types are supported for deferred logging:

   - `4GLTrace`
   - `ASPlumbing`
   - `UBNET`
   - `DB.Connects`

   Note: Deferred logging does not currently support all log entry types that can be configured for the PAS for OpenEdge agent. If deferred logging is turned on, and an unsupported log entry type is specified in the `deferredLogEntryTypes` property, log messages for the unsupported log entry type will be written to the agent log file. At present, `DYNOBJECT` is not supported for deferred logging.

4. When an agent crashes, the deferred logging buffer is printed to the agent log file. Alternatively, you can invoke an API to print the deferred logging buffer to the agent log file or clear the deferred log buffer. For more information, see:

   - REST APIs: Flush deferred log buffer to agent log file and Reset deferred log buffer.
   - OEJMX queries: Flush deferred log buffer to agent log file query and Reset deferred log buffer query.

Updated Spring Security JARs

To mitigate the possibility of security vulnerabilities, OpenEdge 11.7.6 includes updated Spring Security JARs. This update allows OpenEdge users to build more secure web applications for PAS for OpenEdge.

To update existing 11.7.*x* PAS for OpenEdge instances and web applications to use the updated Spring Security JARs:

1. Package the existing OpenEdge 11.7.*x* instance in a ZIP file, for example, `1175_instance.zip`.
2. Copy the ZIP file into the `$WRK` directory for the PAS for OpenEdge 11.7.6 instances, for example: `C:\OpenEdge\WRK1176`.
3. Extract the file in the OpenEdge 11.7.6 `$WRK` directory.
4. For each deployed ABL web application deployed to the *`instance-name`*`/webapps` directory, create a backup of the `WEB-INF/spring` directory, for example:
   - *`instance-name`*`/webapps/ROOT/WEB-INF/spring`
   - *`instance-name`*`/webapps/`*`webapp-name`*`/WEB-INF/spring`

5. For each file described in the following table, replace the existing entry with the new entry:

Filename	Existing entry	New entry
instance-name/bin/deploysvc.sh	DEPLOYSVC_JAR= "$CATALINA_HOME"/bin/ deploysvc.11.7.5.jar	DEPLOYSVC_JAR= "$CATALINA_HOME"/bin/ deploysvc.11.7.6.jar
instance-name/bin/deploysvc.bat	set DEPLOYSVC_JAR= %CATALINA_HOME%\bin\ deploysvc.11.7.5.jar	set DEPLOYSVC_JAR= %CATALINA_HOME%\bin\ deploysvc.11.7.6.jar
instance-name/bin/deploysvcCmd.bat	set DEPLOYSVC_JAR= %CATALINA_HOME%\bin\ deploysvc.11.7.5.jar	set DEPLOYSVC_JAR= %CATALINA_HOME%\bin\ deploysvc.11.7.6.jar
instance-name/bin/deploysvcCmd.sh	DEPLOYSVC_JAR= "$CATALINA_HOME"/bin/ deploysvc.11.7.5.jar	DEPLOYSVC_JAR= "$CATALINA_HOME"/bin/ deploysvc.11.7.6.jar
instance-name/bin/oeprop.bat	set OEPROP_JAR= %CATALINA_HOME%\bin\ oeprop.11.7.5.jar	set OEPROP_JAR= %CATALINA_HOME%\bin\ oeprop.11.7.6.jar
instance-name/bin/oeprop.sh	OEPROP_JAR= "$CATALINA_HOME"/bin/ oeprop.11.7.5.jar	OEPROP_JAR= "$CATALINA_HOME"/bin/ oeprop.11.7.6.jar
instance-name/bin/oepropCmd.bat	set OEPROP_JAR= %CATALINA_HOME%\bin\ oeprop.11.7.5.jar	set OEPROP_JAR= %CATALINA_HOME%\bin\ oeprop.11.7.6.jar
instance-name/bin/oepropCmd.sh	OEPROP_JAR= "$CATALINA_HOME"/bin/ oeprop.11.7.5.jar	OEPROP_JAR= "$CATALINA_HOME"/bin/ oeprop.11.7.6.jar
instance-name/webapps/ ROOT/META-INF/MANIFEST.MF	OpenEdge-version: 11.7.5	OpenEdge-version: 11.7.6

Note: You can update only the `.bat` or `.sh` files based on your operating system.

6. Complete the following step based on the condition that applies to your environment:

   - Case 1—If you did not make any changes to XML files for any of the web applications in the `/webapps` directory, then follow these steps:

1. Replace all of the files in the *instance-name*/webapps/ROOT/WEB-INF/spring directory with the files from the OpenEdge 11.7.6 installation directory ($DLC/servers/pasoe/webapps/ROOT/WEB-INF/spring).
2. Replace all of the files in the *instance-name*/webapps/*webapp-name*/WEB-INF/spring directory with the files from the OpenEdge 11.7.6 installation directory ($DLC/servers/pasoe/webapps/ROOT/WEB-INF/spring) for each OpenEdge ABL WebApp deployed to the instance.
3. If you migrated an oeauthserver instance into the OpenEdge 11.7.6 working directory, then replace all the files under *1176_WRKDIR*/oeauthserver/webapps/ROOT/WEB-INF with latest files from the $DLC/servers/pasoe/extras/oests/WEB-INF directory.
4. If you have the oemanager web aplication deployed on the migrated instance, then extract the $DLC/servers/pasoe/extras/oemanager.war file into the webapps/oemanager directory. In the WEB-INF directory of the oemanager application, replace the following files with the latest versions that you extracted from $DLC/servers/pasoe/extras/oemanager.war:
   - oemgrSecurity-anonymous.xml
   - oemgrSecurity-basic-ldap.xml
   - oemgrSecurity-basic-local.xml
   - oemgrSecurity-basic-oerealm.xml
   - oemgrSecurity-container.xml
   - oemgrSecurity-form-local.xml
5. If you have the wscgi web application deployed on the migrated instance, then extract the $DLC/servers/pasoe/extras/wscgi.war into the webapps/wscgi directory. Replace all the files (cgiSecurity-anonymous.xml, cgiSecurity-basic-local.xml, and cgiSecurity-form-local.xml) in webapps/wscgi/WEB-INF with the corresponding latest files from the $DLC/servers/pasoe/extras/wscgi/WEB-INF directory.

- Case 2—If you changed any of the XML files in the web applications in the /webapps directory, follow these steps:
  1. Perform all the steps described in Case 1.
  2. After replacing all Spring XML files with the latest XML files from the OpenEdge 11.7.6 $DLC, manually merge the customizations from your file backups into the updated XML files.

7. Register the 11.7.*x* instance with OpenEdge 11.7.6 using the PASMAN register command, for example:

```
pasman register 1176_instance C:/OpenEdge/WRK1176/1175_instance
```

Note: For more information about registering instances, see "Register an instance for tracking (register)" in *PAS for OpenEdge Administration Tools Reference*.

8. Verify that the instance is registered:

```
pasman instances
```

9. Start the instance using OpenEdge 11.7.6, for example:

```
pasman start -I 1176_instance
```

Upgrade OpenEdge REST applications

Progress recommends using Progress Developer Studio for OpenEdge to migrate existing REST applications from 11.7.*x* to 11.7.6. Progress Developer Studio for OpenEdge automatically provides the new Spring updates, which are less vulnerable.

All existing applications in 11.7.*x* must be redeployed to the instance in Progress Developer Studio for OpenEdge 11.7.6.

Upgrade OpenEdge SOAP applications

To migrate existing 11.7.*x* PAS for OpenEdge SOAP applications to release 11.7.6, update the hostname under the `soap:address` section in the *`instance-name`*`/webapps/ROOT/WEB-INF/adapters/soap/ROOT/`*`Soapservice-name`*`.wsdl` file. The WSDL file contains the old machine (11.7.*x*) hostname. Replace it with the new machine (11.7.6) hostname.

Upgrade ABL applications

To mitigate security vulnerabilities found in third-party JARs that are in earlier OpenEdge 11.7 releases, these third-party JARs were upgraded in OpenEdge 11.7.6. The upgraded JARs allow you to build more secure web applications with the classic OpenEdge AppServer.

To make this upgrade, Spring XML configurations were updated. This upgrade does not affect existing applications or migrated applications. If you want to use the updated Spring XML configurations, you will need to regenerate and republish your REST application to the classic OpenEdge AppServer.

This document provides detailed steps that help you migrate your REST applications in 11.7.*x* to 11.7.6 (the latest OpenEdge release).

There are three use cases for REST application migration from previous releases of OpenEdge:

- Case 1—Create a new REST application in Progress Developer Studio for OpenEdge to be deployed on the classic AppServer.

  There is no effect on the creation of a new REST application with Progress Developer Studio for OpenEdge, and the REST application can be deployed to the classic AppServer.

- Case 2—Migrate an existing REST application from 11.7.*x* to 11.7.6.

  There is no effect on deploying an existing REST application WAR file in OpenEdge 11.7.6 classic AppServer.

- Case 3—Update services in existing (11.7.*x*) WAR file. You must do the following:

  1. Regenerate the REST application WAR file from Progress Developer Studio for OpenEdge.
  2. Republish the application using Progress Developer Studio for OpenEdge.

Updated GENSPRINGPWD utility for PAS for OpenEdge

In OpenEdge 11.7.6, the GENSPRINGPWD utility was updated to the same encryption algorithm and syntax as the GENSPRINGPWD utility in OpenEdge 12.

For more information, see "Generate encrypted passwords with GENSPRINGPWD" in *Manage Progress Application Server (PAS) for OpenEdge*.

Updated EncodePassword utility for the classic AppServer

In OpenEdge 11.7.6, the EncodePassword utility was updated and requires different configuration steps from the previous versions of OpenEdge.

For more information, see:

- Set hashed and salted passwords using EncodePassword on page 16
- Encrypt a password shared by REST Management Agent and REST web application on page 18

Set hashed and salted passwords using EncodePassword

Progress recommends that you use the EncodePassword utility to generate encrypted passwords. You can add user accounts with hashed and salted passwords in the `users.properties` file.

EncodePassword is only available for REST web applications developed in OpenEdge 11.4 or later releases.

For more information about adding user accounts, see Adding and modifying a user. For more information about hash and salt, see the hash and salt cryptography references online.

For more information about adding user accounts, see Adding and modifying a user. For more information about hash and salt, see the hash and salt cryptography references online.

To use the EncodePassword utility to implement hashed user passwords for a REST web application, `myApplication`:

1. Navigate to the `WEB-INF` directory of the REST web application, `myApplication`:

```
CD path-to\Progress\OpenEdge\servers\tomcat\webapps\myApplication\WEB-INF\
```

2. Execute the EncodePassword located in the REST web application and specify the password that you want to be encrypted, `newpassword`:

```
EncodePassword -A bcrypt -P newpassword
```

   You receive the following output:

```
appSecurity bcrypt password:
$2a$10$z1U2TV4csxAGRbZEeFeOounDhy65RTAmM9k7QJb6txz8XUIYg8spC
OERM oech1 password : 3e373837333620243f202b
```

   The encoded password for the user, `newuser`, is `$2a$10$z1U2TV4csxAGRbZEeFeOounDhy65RTAmM9k7QJb6txz8XUIYg8spC`. This example uses the bcrypt algorithm to hash the password string.

   The EncodePassword utility generates another password, `3e373837333620243f202b`, using the OpenEdge encryption algorithm, `oech1`, for encrypting the shared password used between the OpenEdge REST Management Agent and the OpenEdge REST web applications. For more information about where to use an encrypted password and how to encrypt a password, see Encrypt a password shared by REST Management Agent and REST web application on page 18.

3. Open the security configuration file you chose in Choosing and applying a security configuration and uncomment the code that implements password hashing in the REST web application.

For example, assuming that you are using the Basic security configuration, `appSecurity-basic-local.xml`, do the following:

1. Navigate to and uncomment the code in bold in `appSecurity-basic-local.xml`:

```
<authentication-manager id="RestApplicationtAuth"
   <authentication-provider>
<password-encoder ref="noopEncoder" />
      <!-- Uncomment following and comment above password encoder to
        add strong password encoder users.properties (bcrypt password)
use EncodePassword for the same
            <password-encoder ref="bcryptEncoder" />
            -->
            <user-service properties="/WEB-INF/users.properties" />
        </authentication-provider>
    </authentication-manager>
```

   After uncommenting, the code must look like the following:

```
<authentication-manager id="RestApplicationtAuth"
   <authentication-provider>
<!-- <password-encoder ref="noopEncoder" /> -->
       <password-encoder ref="bcryptEncoder" >
       </password-encoder>
       <user-service properties="/WEB-INF/users.properties" />
   </authentication-provider>
</authentication-manager>
```

2. Save the security configuration file.

4. Add the new user, `newuser`, and the encoded password you generated in Step 2, to the `users.properties` file by appending the following line of code to the file:

```
newuser=$2a$10$z1U2TV4csxAGRbZEeFeOounDhy65RTAmM9k7QJb6txz8XUIYg8spC,ROLE_PSCAdmin,enabled
```

   In this example, you assigned the existing `ROLE_PSCAdmin` role to the user, `newuser`, and enabled `newuser` to work with the REST Management Agent.

Note: The `users.properties` file cannot store plain text and encoded passwords. Therefore, before adding a new user with an encoded password, you must ensure that all the other users have encoded passwords.

---

5. Save and close the `users.properties` file.

Encrypt a password shared by REST Management Agent and REST web application

The REST Management Agent uses the `/adapterman` URI to communicate with REST web applications. The `/adapterman` URI is to be used only in OpenEdge as a cross-context communication port. For security, the `/adapterman` URI has its own security configuration that contains a shared password used between the OpenEdge REST Management Agent and OpenEdge REST web applications.

In a production environment, the `/adapterman` URI is secured with the shared password. The password does not have to be modified as it is consumed by the OpenEdge REST Management Agent and OpenEdge REST web application. However, as part of advanced security measures for the REST web application, you can encrypt the shared password.

To encrypt the shared password:

1. Generate a hashed and salted password for a REST web application user using the EncodePassword utility. The utility provides two passwords: one for the REST web application user and another for the REST Management Agent.

   For more information about generating hashed passwords, see Set hashed and salted passwords using EncodePassword on page 16.

   For example, if the EncodePassword utility gives the following output when generating a password for a user:

```
appSecurity bcrypt password:
$2a$10$z1U2TV4csxAGRbZEeFeOounDhy65RTAmM9k7QJb6txz8XUIYg8spC
OERM oech1 password: 3e373837333620243f202b
```

   The password, `3e373837333620243f202b`, generated using the OpenEdge encryption algorithm, `oech1`, is required for encrypting the shared password used between the OpenEdge REST Management Agent and the OpenEdge REST web applications.

2. Open the `web.xml` file in the REST Management Agent in a text editor, and look for the following code:

```
<context-param>
     <param-name>adaptermanID</param-name>
     <param-value>C62384a0F1516B00</param-value>
</context-param>
   <context-param>
     <param-name>adaptermanSecret</param-name>
     <param-value>1f177e7600202027731f2e293322362174</param-value>
</context-param>
```

   Replace the `param-value` of `adaptermanID` with the new username and `adaptermanSecret` with the new `oech1` encoded password. Extending the example discussed in Step 1, you must make changes, as shown in bold:

```
<context-param>
     <param-name>adaptermanID</param-name>
     <param-value>newuser</param-value>
</context-param>
   <context-param>
     <param-name>adaptermanSecret</param-name>
     <param-value>3e373837333620243f202b</param-value>
</context-param>
```

3. Open the security configuration file, `appSecurity-xxxx.xml`, of your REST web application, and uncomment the code to implement password hashing in it.

   Extending the example discussed in Step 1, if you use the security configuration, `appSecurity-anonymous.xml`, for your REST web application. Do the following:

   a. Find `<authentication-manager id="RestManagerAuth">` in the code and make the updates shown in bold. The encryption algorithm employed by default is `bcryptEncoder`. The newly-generated

username is newuser and password is $2a$10$z1U2TV4csxAGRbZEeFeOounDhy65RTAmM9k7QJb6txz8XUIYg8spC:

```
<!-- REST Manager access -->
    <authentication-manager id="RestManagerAuth" >
        <authentication-provider>
             <password-encoder ref="bcryptEncoder" />
            <user-service>
                 <user name="newuser"
                       password="
$2a$10$z1U2TV4csxAGRbZEeFeOounDhy65RTAmM9k7QJb6txz8XUIYg8spC"
                       authorities="ROLE_PSCAdapter" />
            </user-service>
       </authentication-provider>
     </authentication-manager>
```

b. Save the security configuration file.

4. Restart the web server, and then re-enable the deployed REST web application.