Cúram 8.2

Business Intelligence Development Compliancy Guide

Note

Before using this information and the product it supports, read the information in

Edition

This edition applies to Cúram 8.2.

© Merative US L.P. 2012, 2025

Merative and the Merative Logo are trademarks of Merative US L.P. in the United States and other countries.

Contents

Notices...

Chapter 1 Developing Compliantly with Business Intelligence and Analytics

When you develop Cúram applications, you must comply with certain guidelines to ensure that you can easily upgrade to future versions without affecting your custom functionality. Complying with these guidelines is essential to ensure that the support team can better support your custom implementation.

There are changes to the guidelines since version 6.0.3. While all application customization mechanisms that were already implemented continue to be supported, some customization mechanisms are now discouraged for new development.

1.1 Starting a new project

When you start a new project, it is important to review the development directory structure and place the appropriate files under source code control so that you can track all changes.

When the files are under source code control, tag all development artifacts. Ensure that the tag reflects to the version of the application. At any point, you can then produce a report to identify all files that were added or changed by using file comparison tools. This report is useful when you are upgrading the application.

Review the client and server development directory structure

Review the development directory structure to understand where development artifacts are located, how they are organized, and where to store changes to these artifacts.

The client and server development artifacts are installed in the following directories:

- Client development artifacts are installed into the *webclient* directory.
- Server development artifacts are installed into the *EJBServer* directory.

Within both the *webclient* directory and the *EJBServer* directory, there is a *components* subdirectory, which has a further subdirectory called *custom*. The *custom* subdirectory is where all project-specific development artifacts should be placed. The other *components* subdirectories contain all of the application development artifacts that are delivered with the product.

> **Important:** The *custom* folder contains a starter structure for first usage and is referred to throughout developer documentation as the area in which all artifacts are developed. This convention is not mandatory and it is a project choice to develop within this component or create a new named component appropriate for your project.

Within the *EJBServer\components\custom\model* directory, there is a starter model file and some model fragments.

Business Intelligence: Development Directory Structure

Business Intelligence development artifacts are installed into the Reporting directory.

There is a components subdirectory, which has a further subdirectory named *custom*. Place all client-based project-specific development artifacts in the *custom* subdirectory. The other subdirectories within the components folder contain all the BI application development artifacts that are delivered with the product.

The development directory structure is as follows:

<component_name>

- *data_manager*

  - *initialdata*
  - *demodata*
- *ddl*

  - *oracle*

    - *staging*
    - *central*
    - *datamarts*
  - *db2*

    - *staging*
    - *central*
    - *datamarts*
- *etl*

  - *oracle*

    - *source*
    - *staging*
    - *central*
    - *datamarts*
    - *configtest*
  - *CuramBIWarehouse*
- *jar*
- *run*

  - *orcl*
- *sample*
- *source*

> **Important:** The *custom* folder contains a starter structure for first usage and is referred to throughout developer documentation as the area for customizing all artifacts. This rule is not enforced and it is a project choice to develop within this component or to create a new named component appropriate for your project. For example, within the *\components\custom* folder, there are starter files and some file fragments.

Related concepts

For more information about the reporting directories, see the Business Intelligence Reporting Developer Guide.

Source Code Control

To track all changes to source artifacts, place the development directory structure under source code control. When under source code control, tag all development artifacts.

Ensure that the tag refers to the version of the application. At any point, you can then produce a report by using file comparison tools to identify all files that were added or changed. This report is useful when you are upgrading the application.

1.2 Business Intelligence: Changing Source Artifacts

Direct customer modification of all default Business Intelligence artifacts is now prohibited. Service Packs and Emergency Patches need to be able to safely move, restructure, or overwrite default files.

If default files are modified, Service Packs or Emergency Patches can overwrite them without notice with changes that might not be compatible with the modifications. Reapplying the in-place changes afterward might not be possible.

There are many types of artifacts. Some of these artifacts are critical to the execution of the system and must not be changed. Others can be changed without impacting the overall systems integrity. It is important to be able to distinguish between the two.

Write Source Code for Database Entities

Write all new customer-specific entities in new source files. Place all new source files in the *DDL* subdirectory in the `components\custom` directory.

Write Source Code for Initial and Demo Data

Write new customer specific initial and demo data items in new source files. Place all new source files in the `data_manager` subdirectory of the `components\custom` directory.

Writing Source Code for IBM®Infosphere™ Warehouse and Oracle Warehouse Builder

The development tools that you use can enforce certain rules on how code is written, referenced, and used. Do not customize any default Business Intelligence metadata. Write new customer-specific metadata in new source files.

The IBM®Infosphere™ Warehouse is the design and runtime environment for DB2® clients. Oracle Warehouse Builder provides the design and runtime environment for Oracle clients.

Place all new source files in the `ETL` subdirectory of the `components\custom` directory.

New Build Environment Commands

The `scripts` directory contains most of the default Apache Ant build scripts. Do not modify these scripts directly. Updates to these scripts can be made by creating new custom Ant scripts and by using the Ant inheritance capability.

1.3 Source Code and APIs

APIs are marked as internal or external by annotations. External operations constitute the official API, which you are encouraged to use and call from your own code. Classes with no annotations are considered internal by default.

All Cúram Java functionality is shipped as pre-built JAR files. This functionality is in the Javadoc documentation. Further documentation is available as sample source code for some areas of Java functionality. Taken together, the Javadoc documentation and the sample source provide a complete description of APIs that you can invoke in any custom code you need to write.

The built versions of each component can be found in the following location: `components \<component name>\lib\<component name>.jar`.

Sample source code is available as zip files in: `components\<component name>\sample \src.zip`

Internal APIs

Internal APIs are annotated with `@Accesslevel(INTERNAL)`.

Some Internal APIs are configured to produce 'access restriction' errors in Eclipse if they are referenced. These APIs are annotated with: `@Accesslevel(INTERNAL)`. Internal APIs are not for customer use, and may change without warning in future versions of the product.

If you attempt to reference a restricted API in your code, you will see errors in Eclipse. References to Internal APIs produce Eclipse warnings. Note that neither of these are supported for customer use.

External APIs

External APIs can be referenced directly in customer project code. Such APIs are annotated with `@Accesslevel(EXTERNAL)`. Javadoc information is provided for External APIs on a per-component basis.

The Javadoc information for each component can be found at `components\<component name>\doc\api.zip`. Some components might not have any Javadoc information if they have no External APIs. Only class operations that are in the Javadoc documentation should be referenced from your code; referencing other classes will produce discouraged warnings or access restricted errors and are not supported.

As with all APIs, it is expected that classes that are marked as External will evolve over time, while remaining compatible with previous versions. If you require some capability that cannot be fulfilled through a combination of External APIs and supported extension mechanisms, raise your requirements through Cúram Support. If appropriate, a new API, customization hook, strategy pattern or configuration-based approach can be made available, and such new APIs can be delivered in feature packs. Alternatively, an existing internal API might be re-designated as external, if appropriate.

1.4 Business Intelligence: Recommended Extension Mechanisms

Direct customer modification of all default Business Intelligence artifacts is now discouraged.

Service Packs and Emergency Patches need to be able to safely move, restructure, or overwrite default files. If default files are modified, Service Packs or Emergency Patches can overwrite them without notice with changes that might not be compatible with the modifications made. Reapplying custom changes afterward might not be possible.

Initial Data

Direct customer modification of initial data is now discouraged.

Through the supported customization mechanisms, customers can change the following data items. These overrides must be created in new files within the custom component.

1. Customers can update any "description" data items for initial data records and last written values.
2. The default "Warehouse language" property value can be updated. That is, the default value for the property *BI.BILOCALE* is allowed to change for each customer configuration.

Demo Data

Direct customer modification of demo data is now discouraged.

Demo data is included by default as sample data. Create custom demo data in new files in the custom component.

Entities (DDL)

To add data, add new customer-specific entities and wrap custom BI maintenance operations in their own processes to maintain both tables atomically.

You can then change other application or BI objects, such as BIRT reports, to point to the new entities or views.

You can change the following data definition language objects through the supported customization mechanisms. Create these objects in new files within the custom component.

1. Sequences: Add new sequences, or change the minimum or maximum values of existing sequences.
2. Entities: Add new data items to new customer-specific entities. Customers are discouraged from adding new columns to default entities.
3. Views: Do not change the default views. Add new data items to a new view. Overriding views to redefine the business meaning of the view is discouraged. That is, do not change WHERE or JOIN clause conditions so that it materially changes the business meaning of the view. Create a view if there is a need to define a new business flow.
4. Foreign keys: Creating foreign keys from default entities to custom entities is discouraged. Instead, create keys from custom entities to default entities.
5. Functions and procedures: Create new functions and procedures for any custom transformations.

ETL Metadata

Direct customer modification of BI ETL metadata is now discouraged.

As with the extension of BI Entities, direct extension or override of BI ETL metadata is discouraged. Specifically, direct extension or override of ETL processing is now potentially unsafe. Customers can no longer necessarily have full visibility of where such data is used.

Similar to Entity classes, model and code your own ETL metadata, either wrapping existing ETL processes or implementing new functionality. Otherwise, you might have problems during upgrades.

Related concepts

For a list of file types that are supported by Oracle Warehouse Builder, IBM InfoSphere Warehouse, and for the recommended naming conventions, see the Business Intelligence Reporting Developer Guide.

Other Scripts - Run and Grant Scripts

Direct customer modification of all files within the `run` directory is now discouraged.

Similar to entities, customers must instead code their own run and grant scripts. Customers must create their own custom entries. Custom grant scripts can override the default grant scripts. Custom run scripts can extend and override the default entries.

Java Code

Direct customer modification of Java is now discouraged.

Similar to entities, customers must instead code their own Java processes, either wrapping existing Java processes or implementing new functionality. Direct modification can cause problems during upgrades.

Build Environment

Direct customer modification of the build environment is now discouraged.

Similar to entities, customers must instead code their own build extensions, either wrapping existing build commands or implementing new processes. Direct modification can cause problems during upgrades. Customer can use the Apache Ant inheritance mechanisms to extend or customize the build environment.

1.5 Avoiding common server compliancy issues

Follow the guidelines to avoid these common compliance issues. Following these guidelines from the early stages of a project is relatively easy. However, if you do not, it can result in serious disruptions later and fixing these disruptions can be both costly and difficult.

Use project-specific prefixes in custom artifact names

Avoid naming collisions when you upgrade by ensuring that you always name new, custom artifacts with a consistent prefix for your project. Naming collisions can be difficult to fix afterward. Prefix all new source artifact names with a relevant acronym or abbreviated word to prevent naming collisions from occurring between your custom artifacts and artifacts that Merative ™ might add over time.

Use the same acronym or abbreviated word throughout. As the project progresses, this prefix makes project additions to core artifacts more obvious. This distinction becomes more useful as the development effort grows. Most projects are described by some kind of acronym and this acronym is a good candidate to use as the prefix.

Project-specific prefixes might not apply when you override some application artifacts. Where supported, override mechanisms typically require the custom artifacts to have the same name as the default artifacts that they override, but some exceptions exist.

Some further considerations are as follows:

* There are many different types of identifiers. For example, a file name, an XML ID, a Java class name, or a combination of identifiers.
* A short prefix is advisable because there might be restrictions on name lengths. For example, some types of database identifiers have length restrictions.

> **Note:** In addition to source artifacts, it is also important to consider identifier values that might conflict with values that are used by Merative ™.

Some artifact types have more than one identifier. Remember this when you name your custom artifacts. The following list describes examples of common development artifacts that can cause naming collisions when you take on a new release.

* **Database fields**
  New database fields can be delivered in fix packs. Use project prefixes for database fields to prevent duplicating the names delivered in the fix packs.
* **Application code table items**
  New application code table items can be delivered in fix packs. Use a project prefix when you name custom code table items to prevent duplicating the names delivered in the fix packs. Custom code table items have a value and a Java identifier, and both share a flat namespace with application items in the same code table.

- **Entity classes**

  Custom Entity classes have a table name that shares the flat namespace and database schema with application tables and must have a unique table name within that namespace. It also has a Java class name, which shares a hierarchical namespace and package structure with application Java classes. Use project-specific prefixes for custom entity classes to prevent duplication of the names of the new entity classes.

- **Identifier values that conflict with values used by Merative ™**

  Consider identifier values that might conflict with values used by Merative ™.
  For example, the `TransactionInfo.setFacadeScopeObject` and `TransactionInfo.getFacadeScopeObject` APIs enable developers to access objects that are associated with the current transaction. When you use these APIs, use a String as your object identifier and prefix this string with an appropriate project-specific word to ensure that your data for the transaction does not conflict with Merative ™ data.

Business Intelligence: Examples of Project-specific Prefixes in Artifact Names

These types of collisions can be avoided by ensuring that you always name new, custom artifacts with a consistent prefix.

- **Example**

  For example, you install a Service Pack. You then discover that a new Business Intelligence application ETL process with a custom data item that was also added with the same name, but with a different meaning.

Some artifact types have more than one identifier and you must remember this fact when you are naming them.

- **Example**

  For example, ETL processes.

Related concepts

For more information about prefixes for Business Intelligence artifacts, see the Business Intelligence Reporting Developer Guide.

Business Intelligence: Use Numeric Identifiers in Custom Initial and Demo Data

Predefined initial and demo data is provided with Cúram Business Intelligence and Analytics Reporting.

Initial data is loaded into the BI schemas. Demo data is loaded into a separate data mart schema that is intended for use in demos. A separate schema for demo data ensures that an unintentional dependency on demo data cannot be introduced.

Initial data is installed into the database when a system is first set up or when a system is upgraded.

A set of initial and demo data is provided in the BI application. Customers might also need to add their own initial and or demo data.

To avoid clashes with the initial and demo data that is included in the BI system, take the identifiers for customer initial and demo data from the defined database ranges. For example, identifiers such as primary keys.

Avoid directly modifying application files in place

Continuous delivery, Fix Pack, and iFix releases must be able to safely move, restructure, or overwrite application files. If the included application files are modified, upgrades might overwrite them without notice and the changes might not be compatible with the modifications. Reapplying the in-place changes afterward might not be possible.

Business Intelligence: Exceptions for In-Place Modifications

A list of the small number of exceptions to the in-place modifications rule for Business Intelligence development.

- `<reporting-dir>/project/properties/BIBootstrap.properties`
- `<reporting-dir>/project/properties/BIApplication.properties`
- `<reporting-dir>/.classpath`
- `<reporting-dir>/.project`
- `<reporting-dir>/components/BIBuildTools/.classpath`
- `<reporting-dir>/components/BIBuildTools/.project`

Never create dependencies on sample or demo artifacts

Never create dependencies on sample or demo artifacts. Never rely on dependencies or references to sample or demo artifacts from custom code. Sample or demo artifacts are subject to change without notice

Different product areas in Cúram take different approaches to marking artifacts as Internal, Sample, or Demo, so this information cannot give a concise statement of how to identify them. However, there are a few instances where they can be identified. These instances are artifacts whose name, code package, model package, or file path contain the words Internal, Sample, or Demo, or obvious derivatives of those words. If in doubt, contact Cúram Support.

To request a product enhancement, see the [Merative ™ Ideas Portal](#).

Business Intelligence Sample or Demo Artifacts

For Business Intelligence, the Data Dictionary contains a list of the objects that are Sample or Demo.

For example, default demo data is provided as Sample.

IBM® Cognos content is provided as Sample.

IBM®Infosphere™ Warehouse control flows are provided as Sample.

Business Intelligence: Reflecting Runtime Changes Back to Development System

If artifacts types are modified on production or test systems, always ensure that these modifications are reflected back to the development system.

Do not Create Dependencies on Internal APIs

Internal APIs can change in subsequent versions of the product without notice and should not be referenced directly in custom code.

1.6 Business Intelligence: Component Compliance Details

You must comply with these specific details for the Business Intelligence component.

Where you want to reference a class in your custom code:

- If the class is External, you are allowed to reference it.
- If the class is Internal, you are supported in referencing it in your existing code, but discouraged from doing so. Do not reference internal classes in new code.
- If the class is Access Restricted, you are not supported in referencing it where you want to customize an application class in your custom code.

Files from the `BIBuildTools` and other component folders are copied to temporary build folders during the application build process. The presence of files outside of these folders does not make them available for customization. The `BIApplication.properties` file has a property that lists the set of components that are licensable. Only folders that are specified with this property can contribute to the build.

Cúram Business Intelligence and Analytics Reporting components:

1. The components Javadoc details all customization points and External APIs.
2. The `scripts` directory of a component contains Apache Ant build scripts that must not be modified directly. Updates to any build processes can be made by creating new custom Ant scripts and by using the Ant inheritance capability.
3. The `jar` folder of this component contains previously built build tools. You must use these tools in accordance with the tool sets specified with the Cúram Business Intelligence and Analytics supported prerequisites.
4. With the `BIBuildTools` component, the `drivers` folder contains database drivers that are used to access the application database. If necessary, these drivers might be replaced with another supported database driver. For supported database versions, see the Cúram Business Intelligence and Analytics supported prerequisites.

> **Note:** If a problem arises with a driver that was not included with the product (that is, was not tested and verified for use with the application), the customer might be requested to replace the driver with a version that was tested while the specific issue is raised with the third-party vendor.

1.7 Business Intelligence: Discouraged Extension Mechanisms

Some of the mechanisms that were allowed in versions before 6.0.5 as a means of extending or customizing Business Intelligence artifacts are now discouraged. If you find that a mechanism combination that you want to use is now discouraged, see the Business Intelligence recommended extension mechanisms for current guidance and recommendations for what to do.

Entities

The addition of new data items to default entities is now discouraged.

Customers who want to add data can add new customer-specific entities and wrap custom BI maintenance operations in their own processes to maintain both tables atomically. Other application or BI objects, such as BIRT reports for ETL processes, can then be changed to point to the new entities or views.

Custom Key Performance Indicator (KPI) charts or graphs that are built with BIRT or IBM® Cognos can be built to point to the new views or entities.

Changing an entity attribute option to Not Allow Null values, or creating new attributes without a default value, are now discouraged. If you feel you have a valid need for these options on application Entity attributes, contact Cúram Support.

For information about how to make an enhancement request, see the [Merative ™ Ideas Portal](#) .

Related tasks

ETL Code Extensions

Use these guidelines when you create ETL code extensions.

Direct modification of ETL code is now discouraged

Previously you might have directly modified default ETL processes. Direct modification of the default ETL processes is now discouraged.

Create an ETL process for any custom processing. Both the physical name on disk and canonical name must be unique.

Overriding ETL Processes by copy and paste is now discouraged

Overriding ETL processes is now discouraged. Some ETL processes form part of the critical path of the BI Warehouse component. Overriding ETL processes is now potentially unsafe as you do not have full visibility of any data impacts.

Create an ETL process for any custom processing required.

Overriding ETL Processes for new data items is now discouraged

Previously you might have added an attribute to a default entity to populate data to a new data item. You might have copied and modified a default ETL process to load data into this attribute.

Specifically, overriding of ETL processing is now potentially unsafe. Customers no longer necessarily have full visibility of where such data is used.

For new data items, create a new ETL process that updates the new data item only. In this way default ETL processes continue to support the default business flow while they also supporting the fix packs and interim fixes.

If your customized code base contains any such artifact types, split out any custom processing to new ETL processes, leaving the default processing to be handled by default files. For example, if you copied and modified an existing ETL process to ensure that a new data item is loaded, then the new ETL should now only update the new data item, leaving the default processing to be run by the default ETL process.

Adding metadata for new entities or new attributes to default BI schema metadata files is now discouraged

You might have previously added IBM® Infosphere® Warehouse metadata for new entities or new attributes to the default BI schema metadata files.

This practice is now discouraged. Add any custom entities or custom attributes to new files.

DDL and Initial Data

Previously, in some cases direct modification of default artifacts was the only option available to ensure that customer customizations were built, deployed, and ran by the BI build environment.

From 6.0.5 onwards, the BI build environment now supports the inclusion of customer extensions cleanly. Create all customer extensions in new files within a custom component.

BIRT Reports

Direct modification of all BIRT content is now discouraged.

Service Packs and Emergency Patches need to be able to safely move, restructure, or overwrite default files. If you modify these files, Service Packs or Emergency Patches can overwrite them without notice with changes that might not be compatible with the modifications. Reapplying custom changes afterward might not be possible.

How to Structure Custom Code

For example, to add a custom entity to the warehouse and data mart for the Oracle database, you can use the following process.

DDL changes

> **Note:** Do not prefix the file names in steps 1-3 as they have a special meaning to the build environment. However, you must prefix the objects that are defined with these files.

1. Add any ALTER TABLE, CREATE, or DROP statements to *DropAppTables.sql* and *CreateAppTables.sql* in the *components\custom\ddl\oracle* folder. Repeat these changes for the *staging*, *warehouse*, and *datamart* folders.
2. Create any initial data in *components\custom\data_manager\initialdata*. For a list of the file names that are required, see the development directory structure.
3. Add new ETL process control entries to the files in *files\components\custom \data_manager*.

Where new data items were added to existing tables, create a custom Oracle Warehouse Builder metadata object with the custom folder. That is, add a custom entity to *staging.mod*, *central.mdo*, and *datamarts.mdo* as required.

For example, assume a data item was added to *DW_PersonHistory*. Add an object called *DW_PersonHistory_PROJECTNAME* to *central.mdo*, and write a custom ETL to update this new data item only. The default ETL process can now be overwritten by updates without affecting your changes.

Notices

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the Merative website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of Merative

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of Merative.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

Merative reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by Merative, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

MERATIVE MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Merative or its licensors may have patents or pending patent applications covering subject matter described in this document. The furnishing of this documentation does not grant you any license to these patents.

Information concerning non-Merative products was obtained from the suppliers of those products, their published announcements or other publicly available sources. Merative has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-Merative products. Questions on the capabilities of non-Merative products should be addressed to the suppliers of those products.

Any references in this information to non-Merative websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those

websites are not part of the materials for this Merative product and use of those websites is at your own risk.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

The licensed program described in this document and all licensed material available for it are provided by Merative under terms of the Merative Client Agreement.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to Merative, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. Merative, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. Merative shall not be liable for any damages arising out of your use of the sample programs.

Privacy policy

The Merative privacy policy is available at https://www.merative.com/privacy.

Trademarks

Merative ™ and the Merative ™ logo are trademarks of Merative US L.P. in the United States and other countries.

IBM®, the IBM® logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

Adobe™, the Adobe™ logo, PostScript™, and the PostScript™ logo are either registered trademarks or trademarks of Adobe™ Systems Incorporated in the United States, and/or other countries.

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft™, Windows™, and the Windows™ logo are trademarks of Microsoft™ Corporation in the United States, other countries, or both.

UNIX™ is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.