Mobile Informer Installation Guide

v3.8, April 2025

Version 6.3.0

Table of Contents

Preface

The Mobile Informer platform is a mobility environment built specifically to support mobile applications that integrate directly with Maximo. The platform enables rapid development of lightweight, extensible applications that allow for direct communication with Maximo via web services and comply with open standards.

Deployed within Maximo, the Mobile Informer server platform requires no middleware software. Mobile Informer manages the heavy-lifting and complexity of mobilizing Maximo, letting you design custom user interfaces specifically to support each business process, while taking advantage of the wide range of native features available in each mobile environment.

This guide walks through the process of installing Informer into an existing Maximo system.

Document Changelog

Document Version 3.8

- Updated for Informer 6.3.

Document Version 3.7

- Update Maximo Manager Version

Document Version 3.6.1

- Updated supported MAS Manager Version.

Document Version 3.6

- Added instructions for MAS8 installation.

Document Version 3.5

- Updated for Informer release 6.0.

Document Version 3.4

- Updated latest informer release.

Document Version 3.3

- Updated latest informer release.

Document Version 3.2

- Remove IRM installation requirement

Document Version 3.1

- Updated compatibility section
- Minor clarifications

Document Version 3.0

- Structural, cosmetic, and editing improvements

Document Version 2.11

- Remove extraneous log snippet at the end of the "Online Installation Phase" page

Document Version 2.10

- Update references to 5.8 changelog and binaries not expressly mentioned previously, despite 5.8 features being documented in versions 2.8 and 2.9

Document Version 2.9

- Added documentation for use of PostgreSQL for external informer queues

Document Version 2.8

- Added an LDAP authentication guide
- Updated and expanded the "Cluster Configuration" guide

- Documentation of new Queue processor configuration properties (added in 5.8.0), replacing a collection of old properties
- Clarified wording on the "Leftover PMCOMSR Service" Pre-Installation check

Document Version 2.7

- Remove 7.5 from the "Supported Platforms" list

Document Version 2.6

- Add a "Pre-Installation Check" to check the health and functionality of Maximo's "Admin Mode" toggling
- Separated out the universal "Pre-Installation Checks" from the upgrade-specific ones. Each relevant upgrade version now has its own appendix with all relevant checks.
- Move the "Pre-Installation Checks" to the main "Installation Guide" section to emphasize the fact that it is not elective, and to reduce the chance that it gets skipped

Document Version 2.5

- Update document references for Informer 5.7
- Add section in "Optional components" to cover the use of an external Informer Queue schema, added in 5.7.0
- Rework of monitoring and confirmation instructions in the "Online Installation Phase" documentation
- Added an additional "Pre-Check" for ISCATALOGDEVICE duplicates, when upgrading from 5.5.0 or earlier
- Provide information about IP address blocks used with APNS or GCM(collectively known as the optional "Push" feature)
- Replaced defunct IBM link describing "webservices.unify.faults"

Document Version 2.4

- Include references to the latest Informer and IRM versions (Informer 5.6.0 and IRM 2.1.1)
- Update Installation monitoring step to recommend query against `ISPACKAGE`
- Updated appendix to include NOTIFYCLASSES in Service list and state new init order

Document Version 2.3

- Include references to the latest Informer and IRM versions (Informer 5.5.2 and IRM 2.1.0)
- Slightly simplify a few steps, in recognition of simplifications provided by the above version upgrades
- Removed the details about the security mapping config in the Informer installation appendix, because the installer no longer creates it, nor is such an exception recommended any longer.

Document Version 2.2

- Cited and recommended versions were increased to IRM 2.0.15 and Informer 5.5.0-hf14
- Pre-Installation Checks now include a proactive check for an Informer upgrade issue

surrounding required fields

- Pre-Installation Checks now include a proactive check for a Maximo 7.1 to 7.5 upgrade issue involving the PMCOMSR Service
- SQL statement to disable the ISNOTIFYCLASSES is no longer necessary, and was removed from the guide
- Manual creation of the IRM logger is no longer necessary, and was removed from the guide
- "Live Installation Phase" now includes a check for the explicit"Application of ISINFORMER's manifest has finished" message, instead of waiting for a period of inactivity
- Post-Installation instructions include updated recommendations regarding the `informer.catalog.xml.store` property, which is now enabled by default in new installations

Document Version 2.1

- Removed the recommendation to uncomment the NOTIFYINFO security exception block. This is no longer necessary nor appropriate with modern versions of the Informer Clients.
- Added an explicit step for terminating the installer JVM and starting up the full system once online installation is complete.
- Fix SQL errors in pre-installation steps (apparently caused by autocorrect/spellcheck)

Document Version 2.0

- Restructured document to separate instruction/steps from data/reference.
- Combined IRM and Informer builds, to make installation and upgrade faster.
- Added information about the creation and use of a dedicated IRM log.
- Clarified criteria for knowing when installation is complete(successfully or otherwise).
- Corrected some outdated details and removed some now-obsolete elements no longer relevant in 5.5.

Document Version 1.0

Initial version.

Release Notes

These release notes are stated at a high level, covering only the most significant and noteworthy features.

More detailed, technical release notes can be found on the product page at https://support.interlocsolutions.com/projects/informer-server/wiki/Changelog

Changes in Informer 6.3, since 6.2

- Allow import of license as a file
- Display limited license info in dialog
- Provide more efficient Event Filter
- Provide MaximoVersion info via simplified command property
- Report each user are member of which profiles
- Updating Notification and Catalog Sequences
- Extend GetNotificationMetaData
- Bug fixes.

Changes in Informer 6.2, since 6.1

- Many Bug Fixes
- Providing ability to remove multiple users at once from profile
- Showing queue processors error list
- Allow replacement / update of an existing resource on a profile
- Display Maximo data type in notification object and catalog attributes lists in Informer Developer
- Improve robustness of User Refresh processing
- Retain users in an "inactive" state when permissions are invalidated
- Increase description length of ISFILEFRAGMENT

Changes in Informer 6.1, since 6.0

- Many Bug Fixes.
- Profile Self-Test.
- Improved support of command jars and LiveUtil jars.
- Support multiple LDAP servers for authentication.
- Azure AD (OIDC) login support.

Changes in Informer 6.0, since 5.9

- Better support for MAS8.
- Switch installation process to use updatedb.
- Introduce new queue processors
- Device-based licensing

Changes in Informer 5.9, since 5.8

Maximo Application Suite 8 support

Installation

- Deprecate IRM dependency

Push Notification

- New Push Relay
- Utilize FCM HTTP v1 APIs

User Authentication

- Be able to use Maximo HTTP level user info for Informer session

Changes in Informer 5.8, since 5.7

Catalog Partitioning Strategies

- Catalog Partitions are now a first-class feature, with no performance tradeoffs
- Practical limits on Catalog size are now the practical limits on partition size. Across all partitions in sum, total Catalog size is now practically unlimited.

Improved Queue Processor Configuration

- Informer Queue processor configuration has become simultaneously simpler and more configurable
- Distribution through the cluster is now done via instance-specific Maximo Properties by default

Changes in Informer 5.7, since 5.6

Informer Queue Table Externalization

- Informer Queue activity can now optionally be isolated to a separate Database
- See External Queue Database Configuration

Enhanced Reconciliation UI

- When reconciling Catalogs, the user is prompted for more info as to what kind of changes are to be audited for
- Using the non-default options can improve the performance of the reconciliation process

Changes in Informer 5.6, since 5.5.0

Catalog Partitioning feature

- Mutually-exclusive subsets for a Catalog can be tracked an synced with minimal setup, automatic maintenance, and low overhead.

New Catalog Queue

- New queue has better performance and scaling characteristics
- New priority system helps long-running background work step aside for more timely changes

Changes in Informer 5.5.0, since 5.4

New Catalog API

- Greatly improved performance
- Stronger guarantees that the server will deliver all updates
- Better handles interruption and resumption
- Sufficiently-new Informer Clients on the app-side automatically recognize and use this new Catalog synchronization API
- A legacy mode continues to serve apps using older Informer Clients, without the benefits

Changes in Informer 5.4, since 5.3

New API on the NOTIFYPUB service allowing public and anonymous access to designated commands and resources.

Performance improvements and bug fixes.

Changes in Informer 5.3, since 5.2

Maximo 7.6 support.

New multi-platform installer

Efficiency enhancements

Bugfixes

Changes in Informer 5.2, since 5.0

Catalog change evaluation architecture changed to use a queue instead of performing initial evaluation on background thread . This allows catalog evaluation processing to be isolated to one or more selected JVMs and prevents potential delays in processing if a large number of transactions are performed on a single JVM.

Full catalog data set now tracked . This allows for the forced reevaluation of all catalog records for active devices . The catalog can now be rebuilt in the case that data is loaded directly to the database or if the where clause includes date based parameters.

Added SQLite database generation for download to the device for preloaded catalogs. This allows supporting clients to download the database directly and skip the XML for the initial catalog load.

Added support for resources to be delivered to supporting clients . This allows images, documents, and other application resources to be provided to the application directly.

Changed catalog listeners to be explicitly defined, if upgrading from a previous release the current configuration will be automatically upgraded.

Added monitoring and tracking to allow for better record lifecycle tracking and queue performance monitoring.

Added version installation tracking in the ISNOTIFYVERSION table.

Bug fixes and enhancements.

Supported Platforms

This page describes the supported platforms for Informer.

Platforms

- Maximo 7.6
- Maximo Manage 8

Operating System

All Operating Systems supported by Maximo

Database

- Oracle
- SqlServer
- DB2

Authentication

- Maximo Auth
- LDAP
- Custom

Configuration

- Standalone
- Clustered

Middleware

- WebSphere Application Server 7.0
- WebSphere Application Server 8.5
- WebSphere Application Server 9.0
- WebSphere Liberty Server 22
- WebSphere Liberty Server 23

Client Platforms

- Android 26.0+ (Oreo, API Level 26)
  - Informer Android Client 3.0.0 or later is recommended to utilize all new Server features.
- iOS 11.0+
  - Informer iOS Client 6.0.0 or later is recommended to utilize all new Server features.
- Windows 10 or later
  - Informer Windows Client 2.1.5 or later is recommended to utilize all new Server features.

Installation

Obtaining Binaries

Installation binaries can be found on the downloads page for Informer at https://support.interlocsolutions.com/projects/informer-server/files

Installer bundles can also be downloaded directly:

- http://files.mas.interloc.cloud/files/informer-bundle-linux-6.3.0.tar.gz
- http://files.mas.interloc.cloud/files/informer-bundle-windows-6.3.0.zip

Maximo 7.6

Installers are available for both Windows and Linux/AIX. Windows installers have a `.exe` file extension, and Linux/AIX installers have a `.run` extension.

Informer Installers are available for Windows and Linux/AIX. Your choice of installer should be determined by the OS of the system containing your SMP folder, not by the OS on which WebSphere is running (if these are different systems).

The instructions and screenshots from here on will presume Windows by default, but should remain functionally applicable for all supported systems.

You will need:

- An installer for Informer
- An installer for the Informer Push Relay (if using push messaging)

MAS Manage

Installer are available as Maximo Customization Archive

Pre-Installation Checks

Pending Configuration Changes

The installation procedure applies any pending database configuration changes. Therefore, please make sure that all pending database configuration changes have been applied prior to beginning the Release Manager or Informer installation process.

DBSTORAGEPARTITION

Verify that the entries in `ALNDOMAIN` with domain id `DBSTORAGEPARTITION` are correct for the Maximo environment, and that there are no extra entries. The Informer installation process will use these entries to identify the Maximo database schema.

---

mxe.int.webappurl

Ensure that the Maximo property `mxe.int.webappurl` is a valid URL to access the Maximo application.

---

mxe.int.containerdeploy

For Maximo 7.5.0.x, the Informer installer only supports automatic deployment to the Product Web Service Container. If you want the automatic web service deployment to occur the `mxe.int.containerdeploy` property must be set to 0 (zero).

---

informer.condexp.group

The Informer installation assumes that a security group named EVERYONE exists for use in some required conditional expressions. If your system does not include this security group, add a new system property named informer.condexp.group and specify the global group that all users should be part of.

Industry Solutions and Modules

If the target environment has the Oil & Gas Industry Solution or the Health, Safety Environment Module installed it is necessary to check that each org-level object contains an associated `ORGID` attribute and each site-level object contains an associated `SITEID` attribute. Without these attributes, Maximo database configuration will fail and therefore the Informer installation will not complete.

`ORGID` attributes will need to be added for all objects returned from the following query:

```
select * from maximo.maxobject o
where SITEORGTYPE in ('ORG','SYSTEMORG','SYSTEMORGSITE','ORGSITE')
and o.objectname not in (select a.objectname from maximo.maxattribute a where
attributename = 'ORGID' and a.objectname = o.objectname);
```

`SITEID` attributes will need to be added for all objects returned from the following query:

```
select * from maximo.maxobject o
where SITEORGTYPE in ('SYSTEMORGSITE','ORGSITE')
and o.objectname not in (select a.objectname from maximo.maxattribute a where
attributename = 'SITEID' and a.objectname = o.objectname);
```

Leftover PMCOMSR Service

The PMCOMSR Maximo Service has not existed since Maximo 7.1. The official Maximo 7.5 upgrade guide from IBM includes instructions to disable this Service during upgrade.

When this Service remains disabled, but not removed, the upgrade will be successful and database migration will be possible. As long as it exists, however, even when disabled, it will cause Maximo to fail validation whenever attempting to deploy any new "Standard Web Service" in Maximo. Informer defines such services as part of its installation process, so this Maximo issue must be fixed prior to Informer installation.

If this Service entry still exists in your Maximo 7.5+ instance, remove it entirely using the following SQL:

```
DELETE FROM maxservice WHERE servicename = 'PMCOMSR';
```

Additional Checks During Upgrades

Not all upgrade paths can be anticipated or automatically corrected for, especially when old data/state is the product of bugs since fixed.

If upgrading, please also conduct the appropriate checks and precautions linked below, to help ensure a smooth upgrade.

Choose the most applicable document to match your current version:

- Appendix: Additional Pre-Installation Checks when Upgrading from Informer 5.5
- Appendix: Additional Pre-Installation Checks when Upgrading from Informer 5.3

Setup Phase for Maximo 7.6

Obtain Installation Binaries

At the time of this publication, the latest version of Informer was 6.0.0. This document presumes that the version being installed matches this version or later.

Binaries can be downloaded from our support site at https://support.interlocsolutions.com/projects/informer-server/files or may have been provided to you directly by an Interloc contact.

Informer and Push Relay are delivered in a single package to ensure that you always have compatible versions as a set.

Application Server (JVM) Configuration

Disable SOAP Fault Unification

```
-Dwebservices.unify.faults=false
```

"SOAP fault unification" is a feature which is default-on in WebSphere 8, and is available but default-off in some patch versions of WebSphere 7. This is intended to increase security by stripping error information, but unfortunately also gets in the way of Informer's internal functions which rely on this information. This feature MUST be disabled for Informer Clients (i.e. apps) to work properly. Apply the above parameter to any and all JVMs which could serve Informer requests.

More information: https://www.ibm.com/support/knowledgecenter/SSLKT6_7.6.1/com.ibm.mif.doc/wsregistry/t_config_ws_errors.html

Cluster Configuration

Informer was designed from the beginning to work well within a clustered Maximo environment. This allows Informer to scale with your Maximo installation without the need to manage additional software or systems. In a system of scale, however, a greater level of control is often desirable. Information on how this can be accomplished can be found in the Recommended Infrastructure section of this guide.

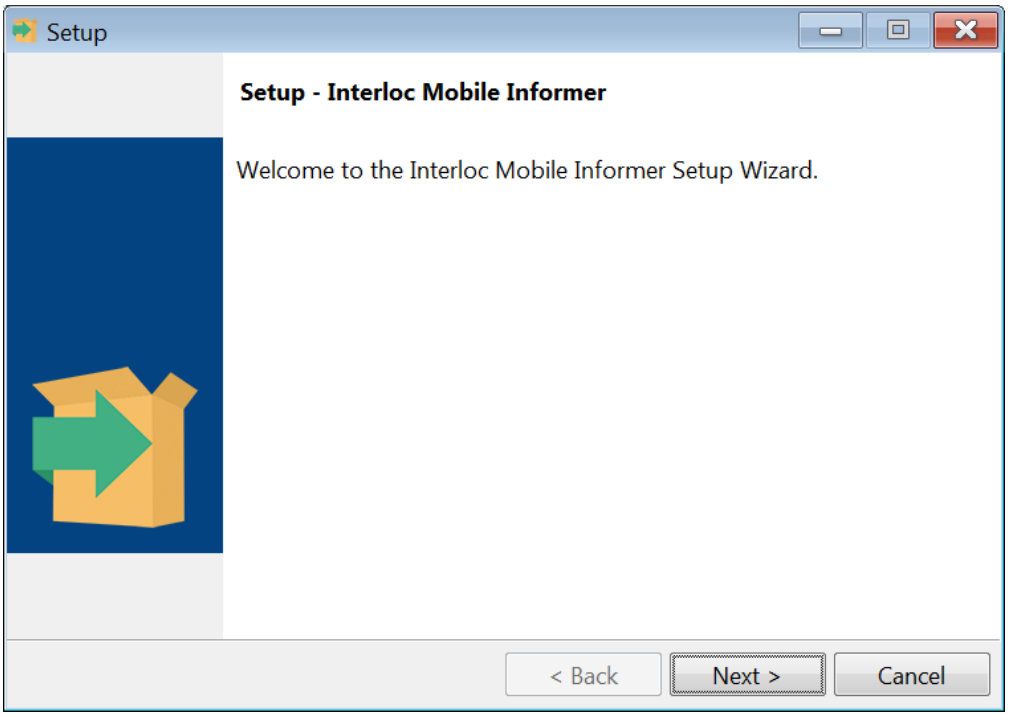
NOTE	The installation and configuration process is the same for clustered and non-clustered environments, so thorough consideration/optimization of your cluster configuration can be deferred until after installation, if you wish.

Informer Installation

Performing the following steps will install the Informer binaries and web.xml configurations into your SMP folder. The Interloc Mobile Informer Platform will install or upgrade itself in the selected Maximo environment once the new EAR starts up.

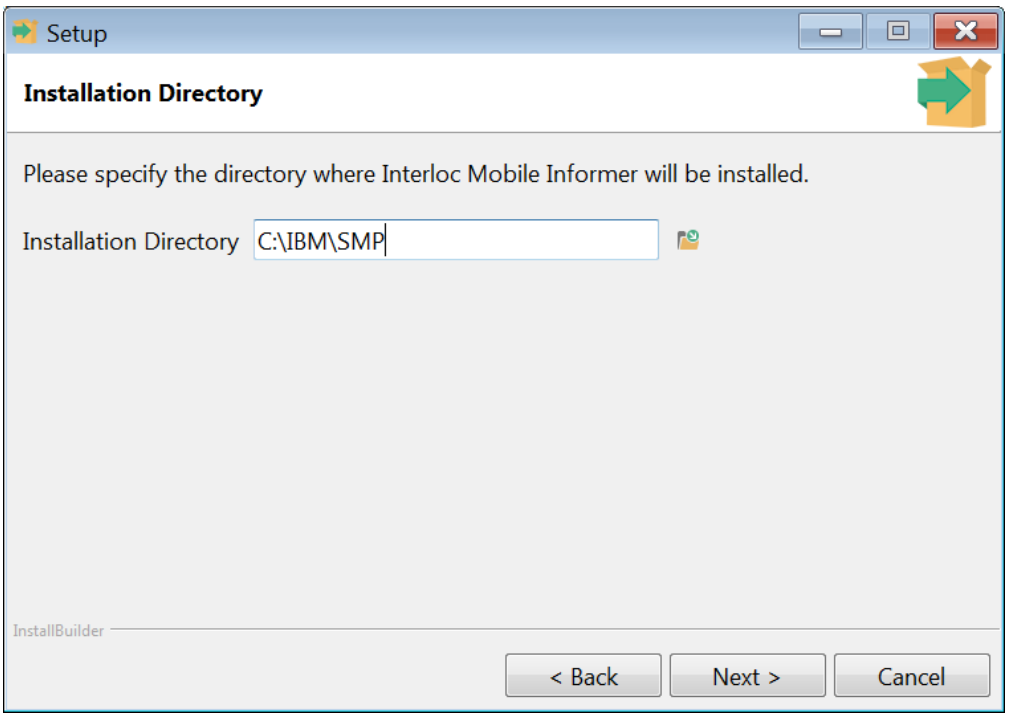
Installation

Execute the `informerserver-*.exe` executable to start the installation, displaying the installation welcome screen.

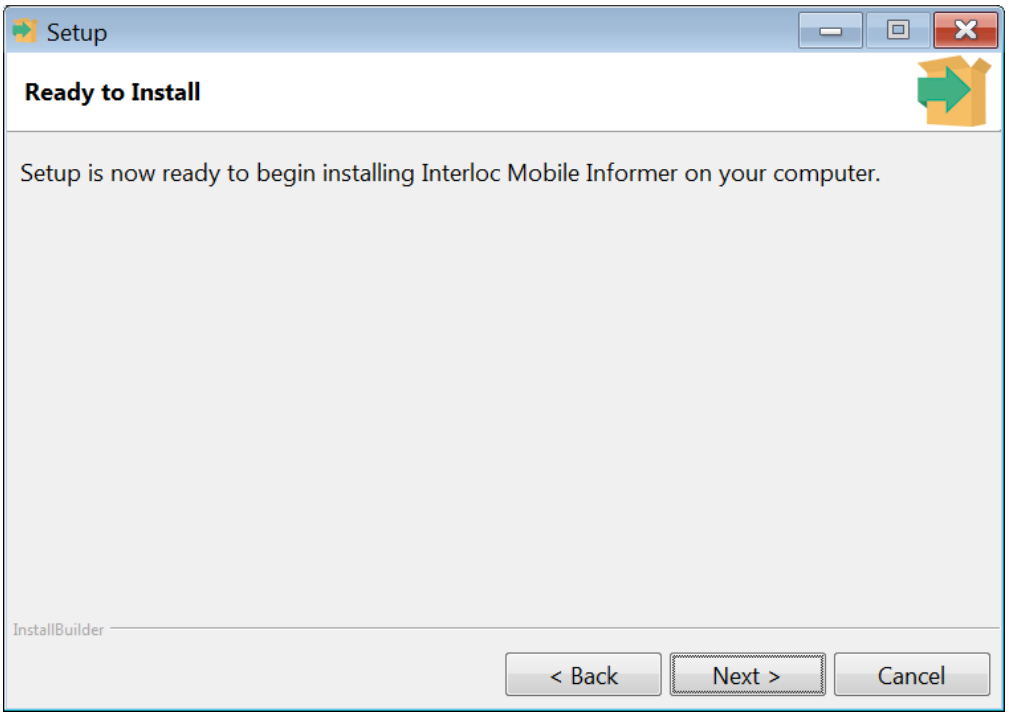


Click the Next button to continue with the installation.

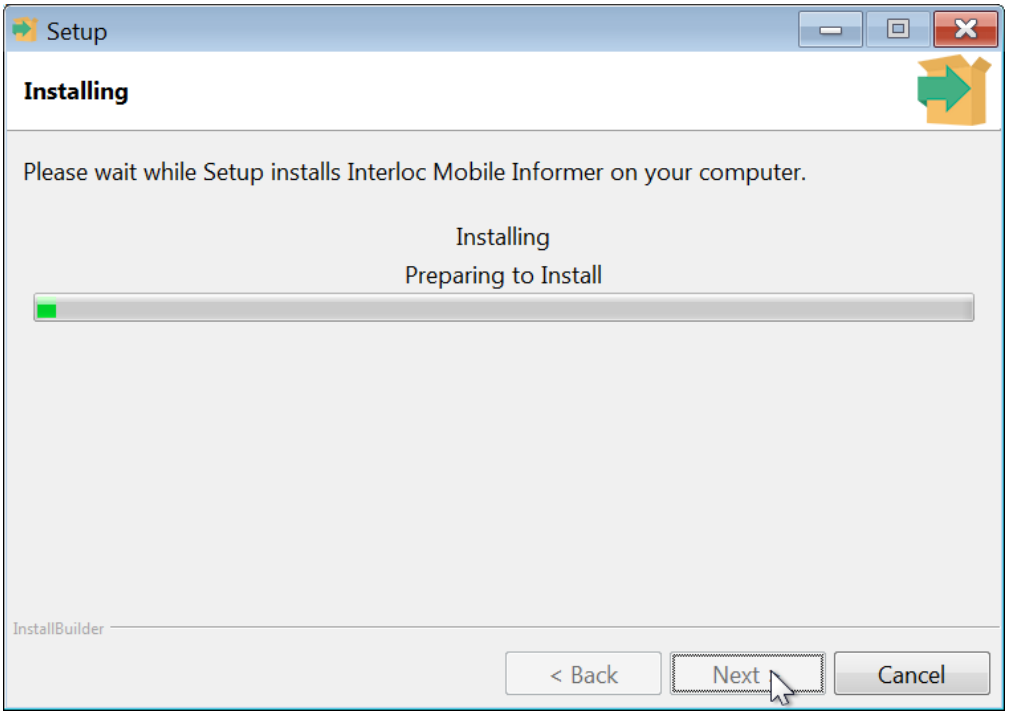
The license agreement is displayed for review and acceptance . Click the"I Agree" button to accept the license and then click Next to continue with the installation.



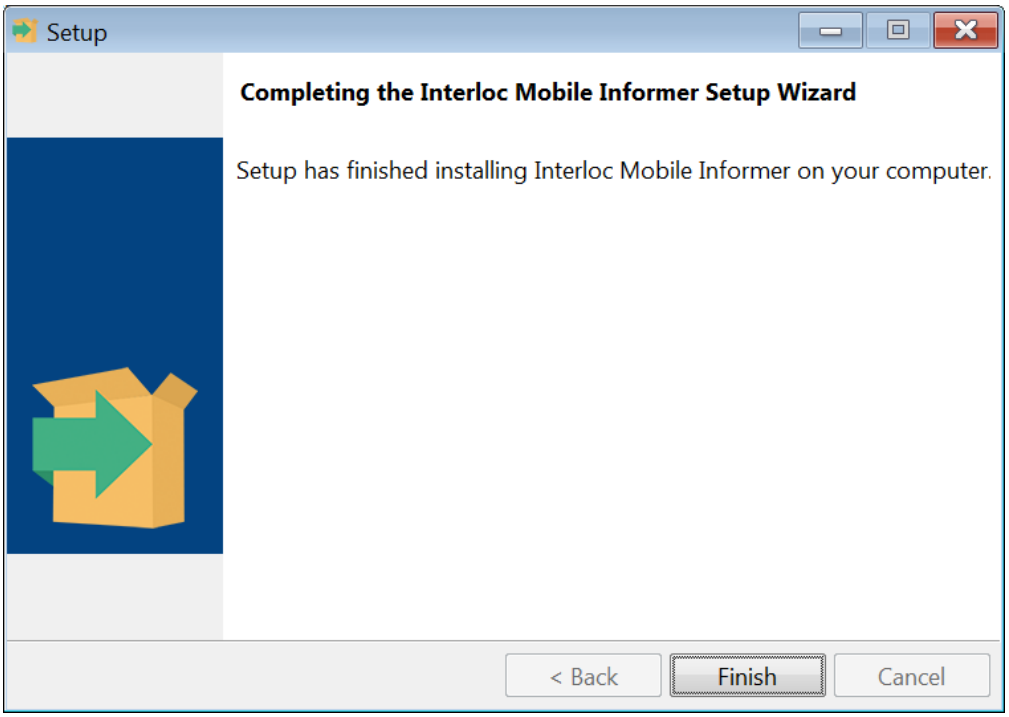
Select the target Maximo installation's parent SMP folder . For example, if Maximo has been installed to `C:\IBM\SMP\maximo` the destination folder should be `C:\IBM\SMP` . The installer validates that you have selected a valid Maximo installation folder before allowing the installation process to proceed.



The installation is now ready to run . Click the Next button to commence the installation.



The installer extracts the installation files to the target environment.



When the installation is complete, the confirmation dialog is displayed . Click the Finish button to complete the installation.

Execute informer-install.bat

Once the installer has completed, a script must be run to modify local XML.

WARNING

Do you have alternative properties files?

By default, the informer-install.bat uses the database connection information found in the `maximo.properties` file located at the following path.

```
../../applications/maximo/properties/maximo.properties
```

This default can be overridden by using the `p` flag or `--proppath` switch, followed by the path to the desired properties file.

Navigate to the `<SMP>\maximo\tools\maximo` subfolder of your IBM product installation folder. Enter `informer-install.bat`, and press `Enter` to execute the command.

NOTE

Details on what this script does can be found in Appendix: Informer Installer Details

Run updatedb.bat

Once `informer-install.bat` has completed, the necessary database changes must be applied. Still within the `<SMP>\maximo\tools\maximo` directory, enter `updatedb.bat` and press `Enter` to execute the command.

Rebuild EAR(s)

Rebuild and redeploy any and all `maximo.ear` files, as you would for any other system configuration change.

Bring everything Online

Start all JVMs in the Maximo system.

The next step is post-installation configuration.

Setup Phase for MAS Manage

Adding Customization Archive

- Obtain the Customization Archive zip file from https://support.interlocsolutions.com/projects/informer-server/files.
- Upload the Customization Archive zip file to a hosting place.
  - Alternatively, this URL http://files.mas.interloc.cloud/files/isinformer-mas-6.3.0.zip can be used if no customization hosting exists.
  - For MAS Manage versions earlier than 8.7, use http://files.mas.interloc.cloud/files/isinformer-mas8pre87-6.3.0.zip

Post-Installation Phase

Grant Security Access

The Informer platform is configured and administered via the Informer Developer and Informer Administration applications respectively within Maximo.

When the application is initially installed, no security groups have access to the application. An administrator needs to log in and grant access to the groups that should have access to the Informer application.

Once access has been granted, the Informer Developer and Informer Administration applications can be found on the Administration menu.

Review Cron Tasks

Informer introduces its own Cron Tasks, for automated maintenance of the Informer system. Some of them are automatically configured with default instances, and some are not.

Please review the Appendix: Informer Cron Tasks section of this guide. The purpose of each Cron is described, as are recommendations and requirements for instance configuration.

For the Crons configured with default instances (e.g. CatalogPreload), review their schedules to determine whether this is compatible with your system load needs.

For the Crons without default instances (e.g. SessionPurge), consider whether this Cron is desirable, and what parameters might be.

Review System Properties

An exhaustive list of Informer properties is included in the Appendix: Maximo Properties section of this guide.

Most are highly conditional, applicable only in specific unusual cases. Some, far less so.

The following is a list of the most commonly relevant properties, which you could consider before considering installation complete.

Take note of which of these values are live-refreshable in Maximo; Most are.

Storing Catalog XML

If Informer has just be installed in this system for the first time, you can skip this section.

Recent versions of Informer can serve Catalog data directly from stored XML. This is enabled by default in 5.5.1, but existing installs may have an existing default to disable it.

This was formerly not enabled by default because the change would invalidate existing data and

synchronization state on devices. As of 5.5.1 this is no longer the case; The change is now seamless, and invisible to devices and users. If upgrading from a previous version of informer, consider enabling this property.

Enabling this behavior does come with an increased use of DB storage space, however the tradeoff of storage cost is almost universally worthwhile because this setting greatly improves synchronization performance, reduces load on primary MBO tables, makes synchronization more reliable in general, and makes the system easier to debug.

Recommendation

1. Set the value of `informer.catalog.xml.store` to `1`
2. "Live Refresh" this property
3. Reconcile all Catalogs on active Profiles (Do not worry about inactive Profiles; These will be upgraded automatically upon activation, if this property is set.)

Push Relay URL

The `informer.push.relayurl` property identifies the path to the Push Relay WAR, covered in the Informer Push Relay section of this guide. This property is how Informer knows where to reach the relay.

Device Serial Number Update

The automatic updating of serial numbers is a feature meant to reduce the number of obsolete device records when devices change IDs. This is primarily an issue on iOS, which will get a new app-specific serial number every time the app is uninstalled and reinstalled (as opposed to upgraded).

In Informer 5.5, this problem is much better served by the DeviceCleanup Cron Task

Recommendation

1. Set the value of `informer.update.serialnum.off` to `1`
2. Configure an instance of the `DeviceCleanup` Cron Task

Informer Configuration Guide

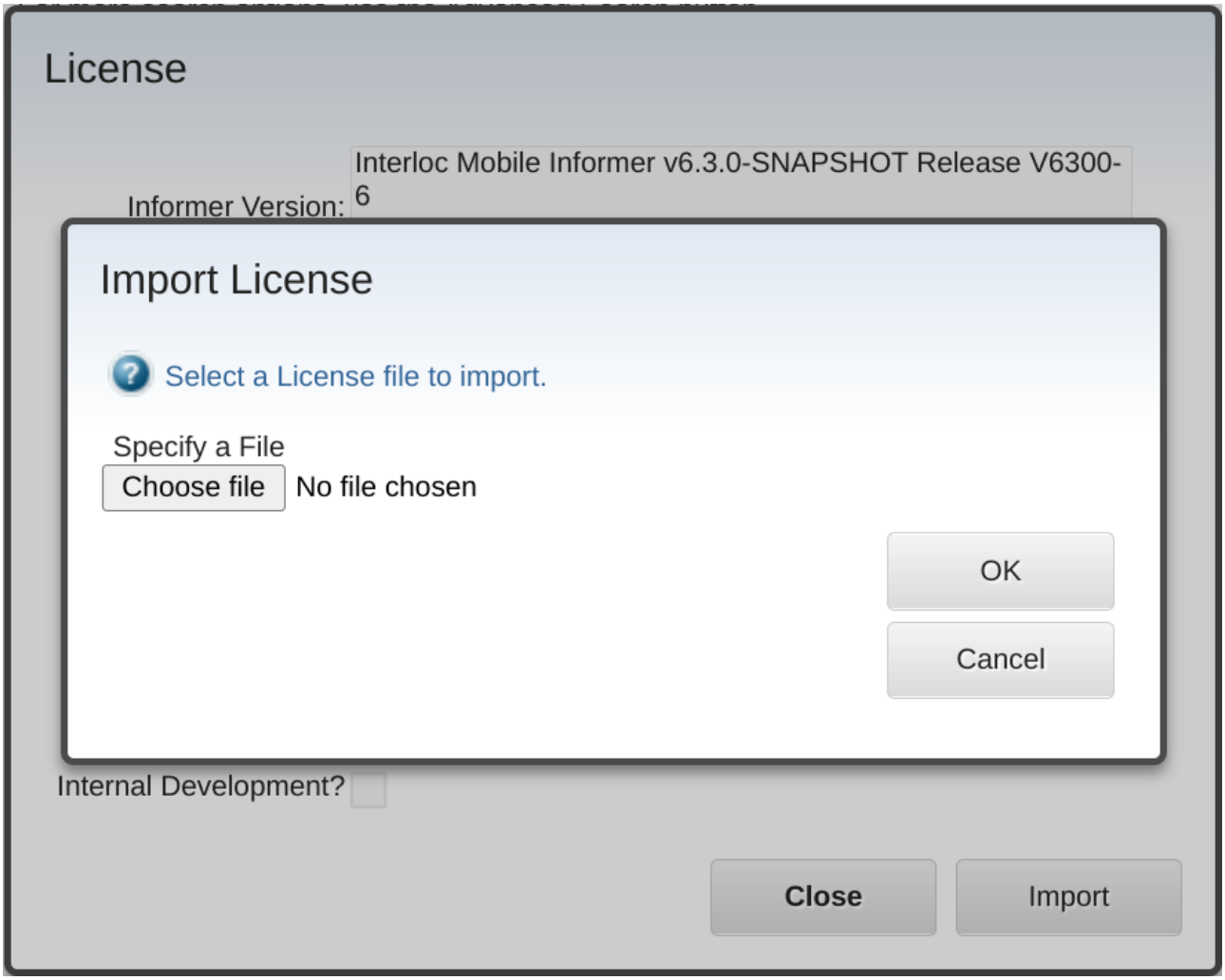
Informer License Installation

Once Informer has been installed a license must be provided before the application will be functional.

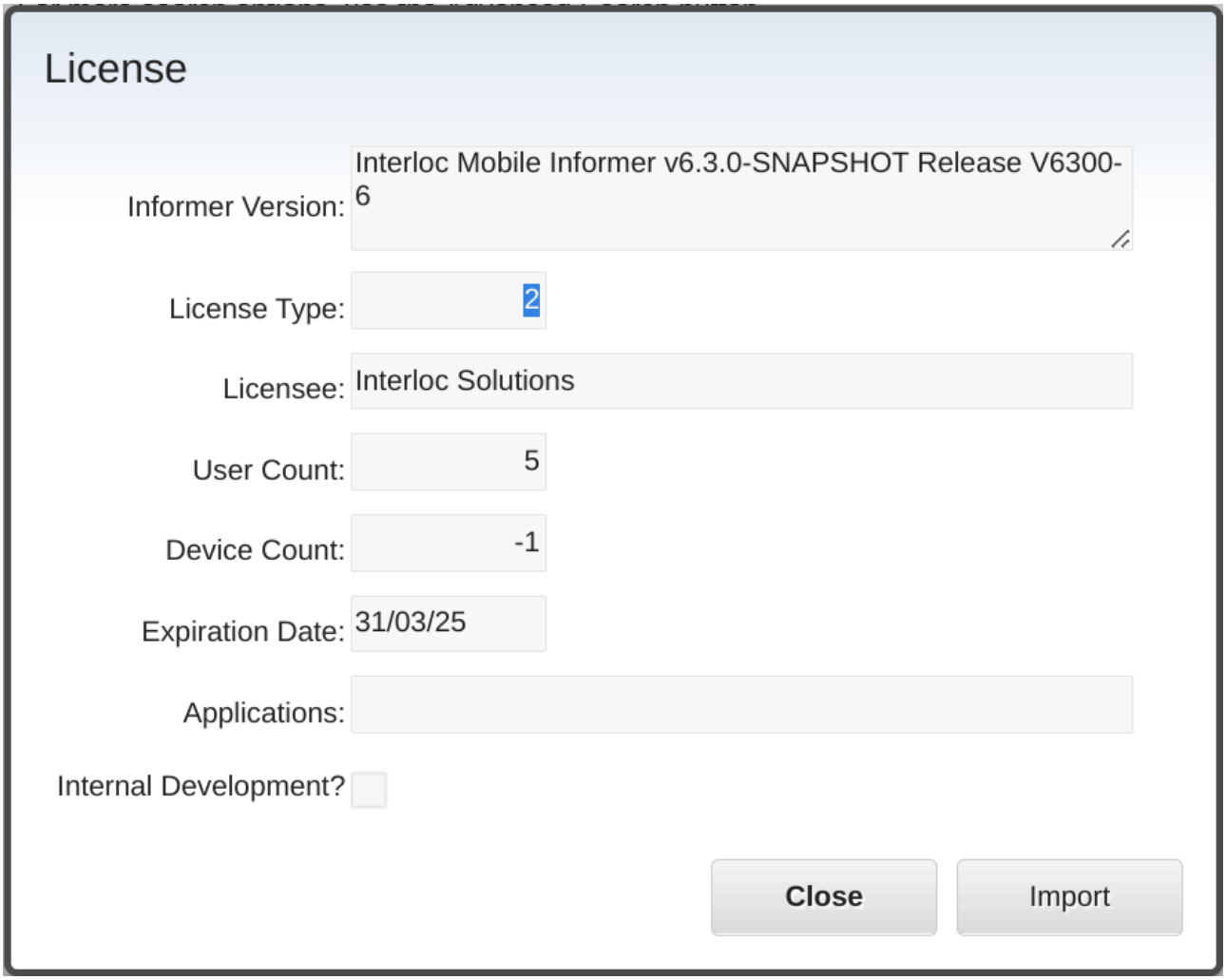
Log into Maximo and navigate to the Informer application using the "Go To" menu:

```
Go To > Administration > Informer > Informer Developer
```

1. While on the List page, select License from the Select Action Menu
2. Click on the Import button.



3. Select the license file to import.
4. Click OK. The dialog box will close and the license details will be displayed.



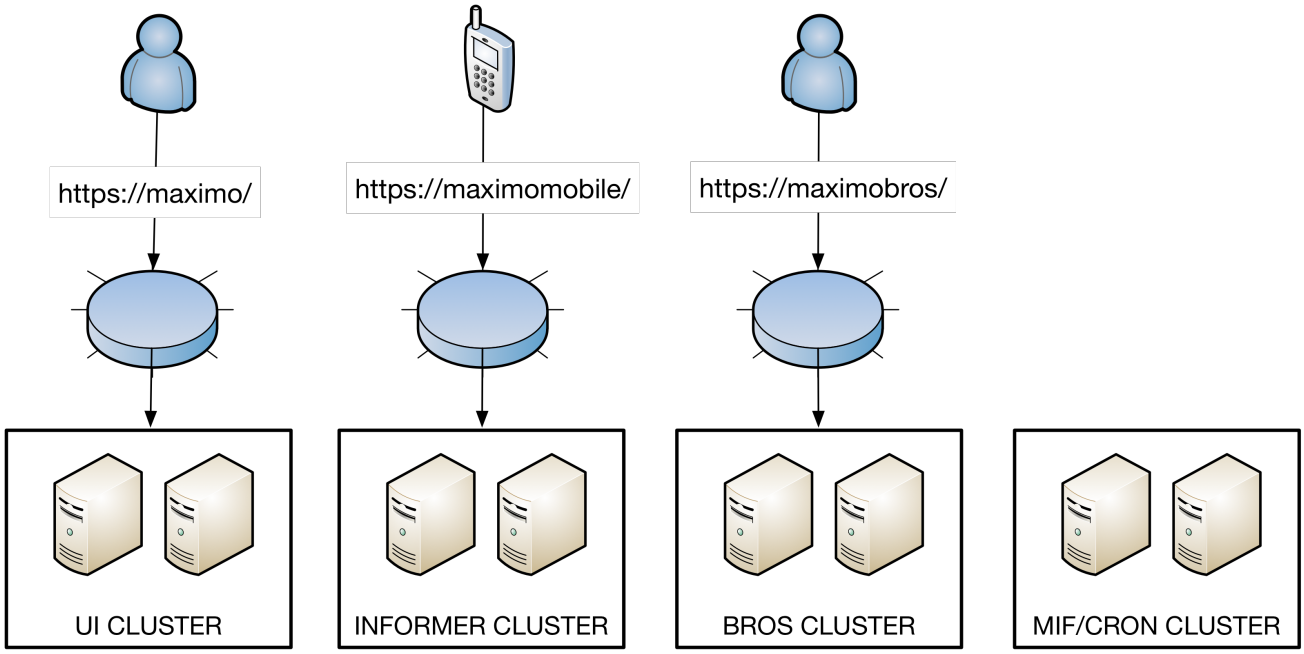
Should additional licenses be purchased a new license will be provided. Importing the new license file through this dialog box will update the available licenses within minutes.

Cluster Configuration

Recommended Infrastructure

Based on IBM recommended best practices it is recommended to create separate JVM clusters for UI, MIF, CRON, and BROS (Reports). It is recommended when deploying Informer to define a fifth type of JVM cluster, dedicated to queue processing and requests from mobile devices. The recommended cluster configuration is based on IBM Knowledge Center – Configuring clustered systems.

Task	Purpose
Create a maximo.properties file for each cluster that you want to deploy.	You create separate properties files so that each cluster can have different settings. For a dedicated Informer EAR, disable scheduled reports with `mxe.report.birt.disablequeuemanager=0` and crontask with `mxe.crontask.donotrun=ALL`
Create copies of the ejb-jar.xml file for each cluster that you want to deploy. If your deployment includes WebSphere Application Server, you also need to create and edit copies of the ibm-ejb-jar-bnd.xmi file.	The `ejb-jar.xml` file and the `ibm-ejb-jar-bnd.xmi` file are modified to configure message-driven beans for continuous queues. In an Informer EAR, the MEA MDB sections should be commented out.
Create copies of the `buildmaximoear.cmd` file for each cluster that you want to deploy.	The `buildmaximoear.cmd` files are used to create the individual EAR files for each cluster.
Build the EAR files.	The EAR files for each cluster are built based on the settings in the individual properties files, `ejb-jar.xml` files, and the `ibm-ejb-jar-bnd.xmi`.
Create the clusters.	The clusters are created by creating JVMs that are members of the cluster.
Deploy the EAR files for the clusters.	You deploy the EAR files on the application server so that each cluster supports its dedicated functions.



NOTE	Informer uses neither RMI nor JMS. When creating an Informer cluster, Informer JVMs should not run JMS queues (unlike other MIF JVMs).

Queue Processors

Informer was designed from the beginning to work well within a clustered Maximo environment. This allows Informer to scale with your Maximo installation without the need to manage additional software or systems. However, having control over how Informer operates in the cluster by limiting the number of processing threads on a given node may be highly desirable. Informer will put additional processing load on both database and application servers. Every Informer deployment will be unique and it is recommended to perform load testing prior to deploying in production.

There are three primary types of queue processors: "Notification","Catalog", and "Push".

The number of each queue processor thread can be configured with these System/Instance Properties:

Property Name	Description	Default Value
informer.queue.pool.notification	Specifies the level of concurrency for Notification queue processing. A setting of "0" disables the Notification queue processors on the JVM.	3
informer.queue.pool.catalog	Specifies the level of concurrency for Catalog queue processing. A setting of "0" disables the Catalog queue processors on the JVM.	2
informer.queue.pool.push	Specifies the number of "Push" queue processing threads to run on a specific JVM node. If this is set to "0" then no push queue threads will run. NOTE: Any given node can have a maximum of "1" push queue processor.	1

The default global properties specify each type of processor will run on any given JVM. This ensures

that single-JVM systems perform all required functions, and each added JVM will automatically spin up its own processors to share the workload. In practice, however, deployments of scale will often want to increase the level of concurrency (and therefore, the system resources) granted to the Informer processors. Not all JVMs in a Maximo environment are equivalent, however, so Maximo's instance-specific property support can be used to create uneven distribution of work.

NOTE

Upgrading?

- If you are upgrading from Informer 5.9.3 or earlier, the installation will convert the old-style System Properties to these new styles. This applies to both global and instance-specific property configurations.
- It was once common to use JVM arguments to configure these pools, rather than Maximo System Properties. Stored in WebSphere (as opposed to the Maximo DB), these properties cannot be read by the installer JVM and therefore cannot be automatically translated for other members of the cluster. After upgrade, please review your JVM arguments, create equivalents in global and/or instance-specific System Properties, and remove these legacy arguments. Henceforth, all such configuration will occur in Maximo, not WebSphere.

Notification Pool Configuration

The `informer.queue.pool.notification` property is specified as single numeric value that represents the total number of processors.

When scaling up, the most effective number of processors depends on your Profile, users, and environment.

Catalog Pool Configuration

The `informer.queue.pool.catalog` property is specified as a single numeric value that represents the total number of processors.

Push Pool Configuration

The `informer.queue.pool.push` property is specified as a single numeric value. If using a Push configuration, set this value to 1 on at least one JVM. If you are certain you will not use Push, you can disable this globally by setting a value of 0.

Example 1. Dedicated Single JVM

Scenario: A clustered system which has a single dedicated Informer JVM, called INF1. Push configurations are used by one or more Profiles in the system.

Property	Instance-Specific?	Value
informer.queue.pool.notification		0
informer.queue.pool.catalog		0
informer.queue.pool.push		0
informer.queue.pool.notification	INF1	16
informer.queue.pool.catalog	INF1	33
informer.queue.pool.push	INF1	1

Example 2. Dedicated Cluster

Scenario: A clustered system which has two dedicated Informer JVMs, called INF1 and INF2. Push configurations are used by one or more Profiles in the system.

Property	Instance-Specific?	Value
informer.queue.pool.notification		0
informer.queue.pool.catalog		0
informer.queue.pool.push		0
informer.queue.pool.notification	INF1	15
informer.queue.pool.catalog	INF1	33
informer.queue.pool.push	INF1	1
informer.queue.pool.notification	INF2	15
informer.queue.pool.catalog	INF2	33
informer.queue.pool.push	INF2	0

Example 3. Reuse of Non-UI JVMs

Scenario: A clustered system which has no dedicated Informer JVMs. The UI JVMs should be protected, but other JVMs in the system have a little bandwidth to spare. No Profiles in the system use Push.

Property	Instance-Specific?	Value
informer.queue.pool.notification		0
informer.queue.pool.catalog		0
informer.queue.pool.push		0
informer.queue.pool.notification	MIF1	6
informer.queue.pool.catalog	MIF1	11
informer.queue.pool.notification	MIF2	6
informer.queue.pool.catalog	MIF2	11
informer.queue.pool.notification	CRON1	12
informer.queue.pool.catalog	CRON1	21

Example 4. Equal Distribution

Scenario: A clustered system which has no dedicated Informer JVMs, but just wants to share a light load equally across all JVMs and see how that goes. Push configurations are used by one or more Profiles in the system.

NOTE This is the same as default configuration, without applying any changes.

Property	Instance-Specific?	Value
informer.queue.pool.notification		3
informer.queue.pool.catalog		2
informer.queue.pool.push		1

Example 5. Minimal Maximo System

Scenario: A single-JVM system which needs to accelerate Informer processing. Push configurations are used by one or more Profiles in the system.

Property	Instance-Specific?	Value
informer.queue.pool.notification		21
informer.queue.pool.catalog		22
informer.queue.pool.push		1

LDAP Configuration

The Informer Server responds to the app via Maximo web services. One of the most important services is `login` where Informer authenticates and authorizes the user to make an appropriate response for that request.

The Authenticator class Informer uses to conduct authentication is configurable/replaceable if there is a need, but the default ones provided with the product cover most cases. By default, Informer is configured to authenticate against the Maximo user database. If Maximo is delegating authentication to LDAP, Informer must be told how to do the same. All relevant settings are specified using Maximo System Properties, all of which can be live-refreshed for near-immediate effect in a live system.

Mandatory Properties

The configured Informer Authenticator should be switched from the default "MaximoAuthenticator" to the "LDAPAuthenticator". All other properties mentioned in this document are configuration parameters for this new authenticator.

```
informer.authenticator =
"com.interlocsolutions.maximo.notify.security.LDAPAuthenticator"
```

The default LDAP Context implementation is provided by WebSphere. We recommend changing this to one provided by the JDK, instead.

```
informer.security.ldap.context.factory = "com.sun.jndi.ldap.LdapCtxFactory"
```

The value of `informer.security.ldap.authtype` should always be `simple`; SASL is not supported in this implementation.

The `informer.security.ldap.url` property identifies the path to your LDAP server, including protocol and port number.

Authentication Strategies

When authenticating via LDAP via the `simple` strategy, a (DN, password) pair is used. Maximo login uses a simple "loginid", however, not a DN, so it is typically necessary to run an LDAP search to find the user's DN.

Method 1: Authenticated User Lookup

In this setup, the authenticator will bind first as a service account, and use that connection to find the user's DN.

To provide credentials for the service account (lookup user), the `informer.security.ldap.lookup.user` and `informer.security.ldap.lookup.password` properties are

used.

The base DN for the search can be specified using the `informer.security.ldap.search.path` property.

Within the search context (with base DN as the root), some search criteria must be used to find the appropriate record in the directory. Use the `informer.security.ldap.search.pattern` property to specify the LDAP search filter, using `{0}` as a placeholder for whatever loginid is being authenticated.

LDAP supports multiple kinds of search scopes which can be declared, defining how deep the search traversal should go. The default is `SUBTREE`, but `OBJECT` and `ONELEVEL` can also be specified.

For example, one such configuration might look like this:

```
informer.security.ldap.context.factory = "com.sun.jndi.ldap.LdapCtxFactory"
informer.security.ldap.searchbase = ""
informer.security.ldap.authtype = "simple"
informer.security.ldap.url = "ldaps://ldap.example.org:636"
informer.security.ldap.lookup.user = "cn=myserviceacc,ou=Users,dc=example,dc=org"
informer.security.ldap.lookup.password = "agoodpassword"
informer.security.ldap.search.path = "ou=Users,dc=example,dc=org"
informer.security.ldap.scope = "SUBTREE"
informer.security.ldap.search.pattern = "uid={0}"
informer.security.ldap.user.pattern = ""
```

For more info on subjects like "search base DN", "search filter", or"search scope", documentation can be found here: https://ldap.com/the-ldap-search-operation/

Method 2: Anonymous User Lookup

Some LDAP servers can be configured to allow unauthenticated searches of the directory. The LDAPAuthenticator does not currently support unauthenticated search. You can try to use the Method 3 defined below, if applicable, or else define a service account for use by Informer (even if this is not strictly necessary) for use in Method 1.

Method 3: User DN Pattern Shortcut

In the case that the loginid appears as part of the user DN, and all users to authenticate are siblings at the same level in the hierarchy, a DN pattern can be specified to skip the user lookup search entirely by transforming a loginid directly into a user DN.

The symbol `{0}` is once again used a placeholder for whatever loginid is being authenticated.

```
informer.security.ldap.context.factory = "com.sun.jndi.ldap.LdapCtxFactory"
informer.security.ldap.searchbase = ""
informer.security.ldap.authtype = "simple"
informer.security.ldap.url = "ldaps://ldap.example.org:636"
informer.security.ldap.lookup.user = ""
informer.security.ldap.lookup.password = ""
informer.security.ldap.search.path = ""
informer.security.ldap.scope = ""
informer.security.ldap.search.pattern = ""
informer.security.ldap.user.pattern = "uid={0},ou=Users,dc=example,dc=org"
```

Optional Tools and Components

These elements are not part of the directed installation guide, but are worth knowing about and considering when planning or reviewing an Informer installation.

Informer Push Relay

Does This Apply To Me?

The Push Relay is required only when using a "push" configuration for data update notification.

Use of a push configuration is generally recommended for responsiveness, but it does require communication with Apple or Google servers and this is not practical for all customers and all contexts.

By default, all Informer apps can handle a configuration with or without a push configuration, but a push configuration can serve as an accelerator to improve time-to-delivery of updates.

Apple and Google each provide native push services, APNS and FCM respectively, and Informer can be configured to work with these services. However, these services rely heavily on digital certificates that are incompatible with the IBM Java SDK runtime which the WebSphere version that Maximo deploys to uses. To allow these services to be used by Informer a small relay services is utilized, which runs with a 1.17-compatible Oracle Java SDK runtime.

System Requirements

Push Relay installers are available for Windows and Linux, but can also be deployed manually to any compatible Servlet container that supports JAX-RS . The WAR file for deploying to a non-Windows environment can be obtained by running the installer then copying the push.war file found in the `<push_home>\webapps\push.war` in Windows, or `/opt/pushrelay/informer/webapps/push.war` in Linux or AIX.

Obtaining Binaries

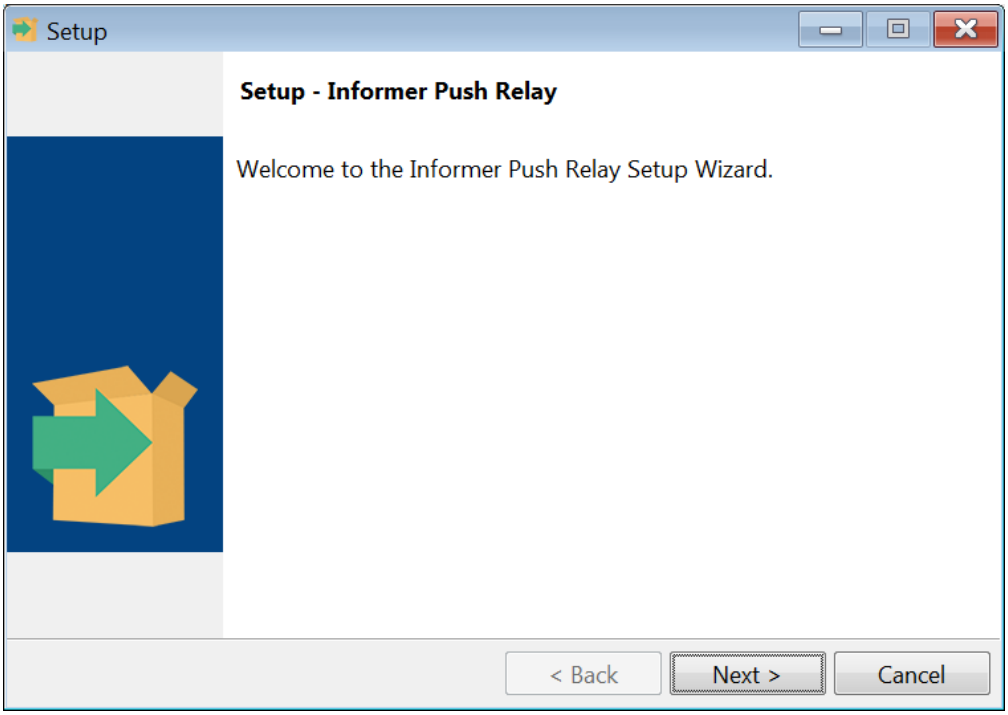
The Informer Push Relay is distributed as part of the bundled Informer installation downloads. Install (or upgrade to) the version of the Push Relay which comes alongside the primary Informer installer. Informer installation bundles can be found on the downloads page for Informer: https://support.interlocsolutions.com/projects/informer-server/files

Installers are available for both Windows and Linux/AIX. Windows installers have a `.exe` file extension, and Linux/AIX installers have a `.run` extension. Choose the version which matches your target system. The instructions and screenshots from here on will presume Windows by default, but should remain functionally applicable for all supported systems.

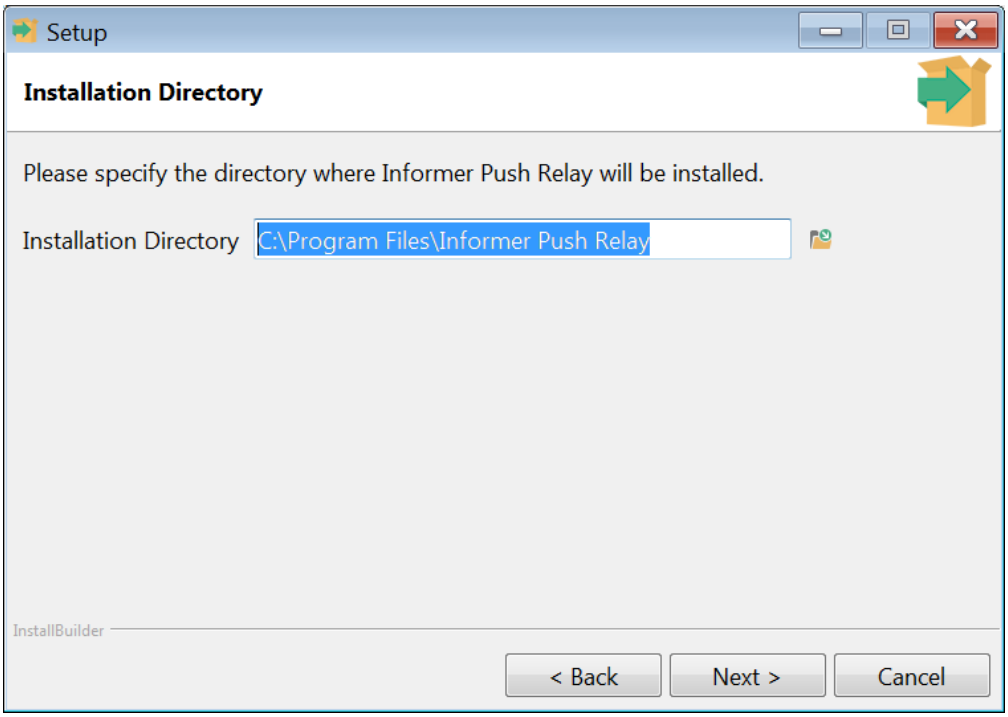
WARNING	If there is an older Push Relay already installed (5.6.1 or earlier), the previous one needs to be uninstalled first.

Installation

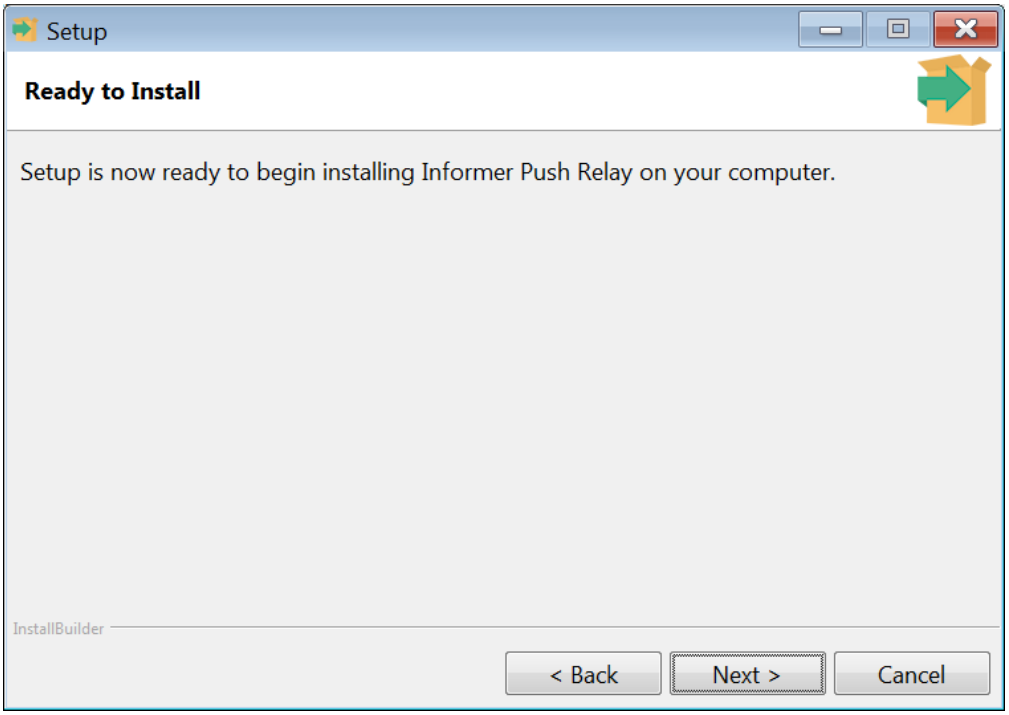
Execute the `push-relay-*.exe` executable to start the installation.



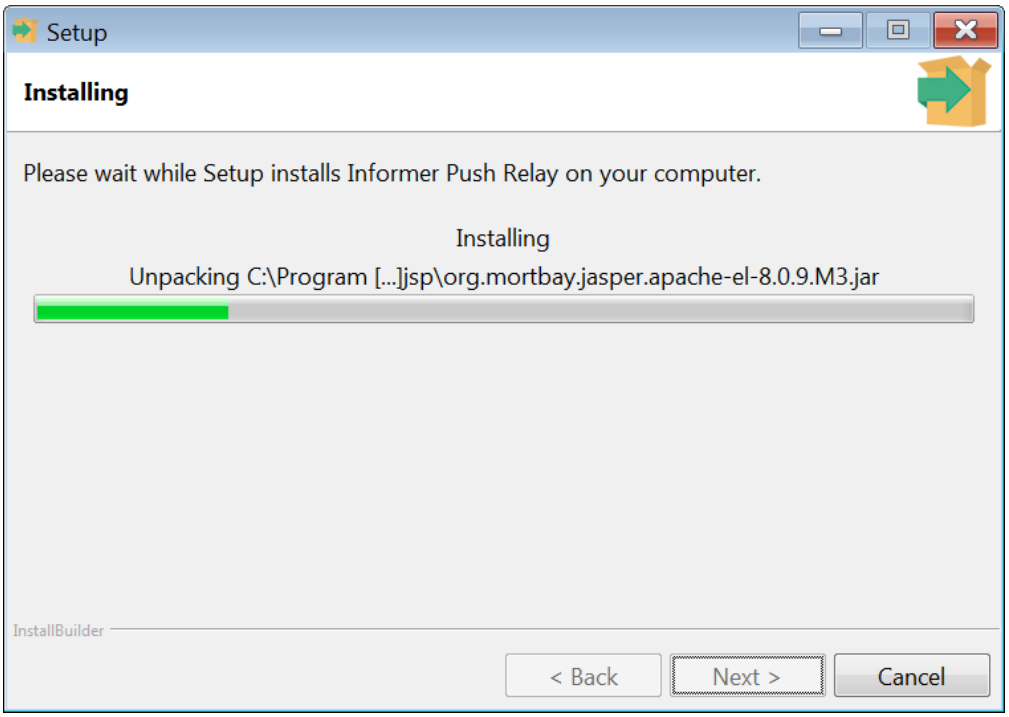
Click the Next button on the welcome screen to continue with the installation.

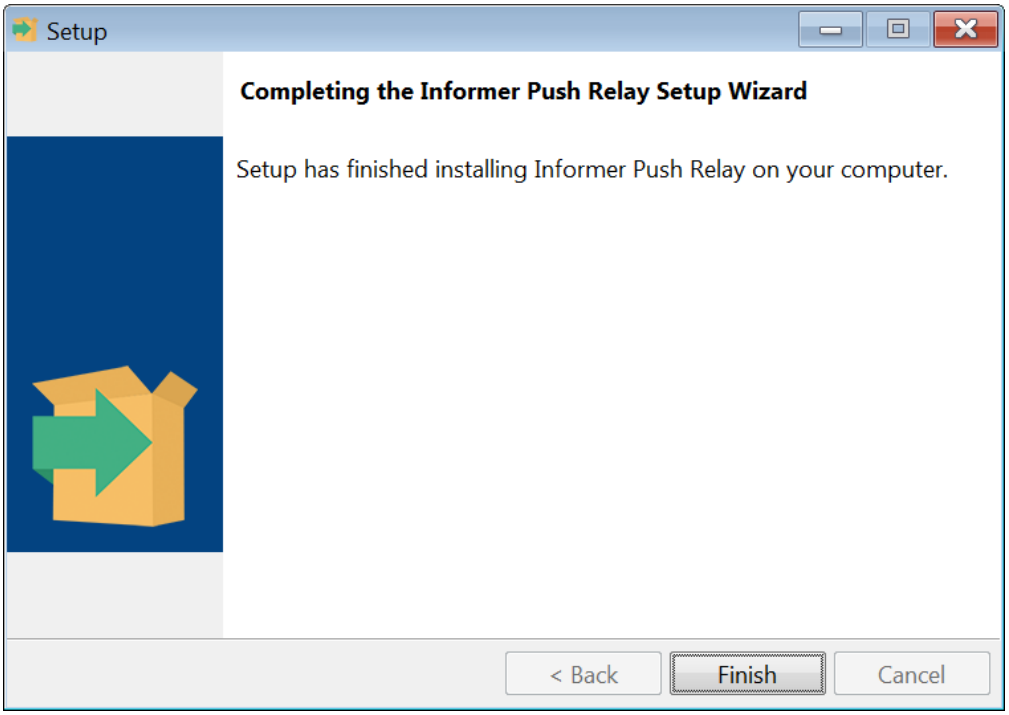


Select an installation folder, or leave the default and click the Next button.



Click the Next button to commence the installation.





Click the Finish button to complete the installation.

Verification

To verify that the push relay was installed correctly, open your browser and navigate to `http://<hostname>:8080/push/verify` where `<hostname>` is the name of the server that the relay service was installed on.

If everything was successfully installed, a page should be displayed stating

```
The Interloc Mobile Informer Push Relay is running version x.y.z.
```

Configuration

Set Maximo System Property

Set the value of `informer.push.relayurl` to `http://<hostname>:8080/push` where `<hostname>` is the name of the server that the relay service was installed on.

Apply Firewall Settings

The Informer push notifications rely on the native push frameworks provided by Apple and Google to work efficiently and effectively . This however, does require Maximo to communicate with servers operated by Apple and Google so that push notifications can be sent, and for devices operating on a Wi-Fi network, received.

Configuring for APNS (Apple/iOS)

When using Apple Push Notification Services (APNS) TCP communication must be available

outbound on port 2195 and must be available inbound on port 5223.

The IP addresses used are less clearly defined. At the time of this writing, Apple had this to say:

> The APNs servers use load balancing, so your devices don't always connect to the same public IP address for notifications. It's best to let your device access these ports on the entire 17.0.0.0/8 address block, which is assigned to Apple.
>
> https://support.apple.com/en-us/HT203609

Configuring for FCM (Google/Android)

When using Google Firebase Messaging (FCM) TCP communication must be available on ports 5228, 5229, and 5230 . Google states that most communication will occur on port 5228, but there are times when ports 5229 and 5230 are used.

The IP addresses used are less clearly defined. At the time of this writing, Google had this to say:

> For outgoing connections, FCM does not provide specific IPs because our IP range changes too frequently and your firewall rules could get out of date impacting your users' experience. Ideally, you will whitelist ports 5228-5230 with no IP restrictions. However, if you must have an IP restriction, you should whitelist all the IP addresses in the IPv4 and IPv6 blocks listed in Google's ASN of 15169. This is a large list and you should plan to update your rules monthly. Problems caused by firewall IP restrictions are often intermittent and difficult to diagnose.
>
> https://firebase.google.com/docs/cloud-messaging/concept-options#ports_and_your_firewall

Informer Server Check Utility

The Informer Server Check is a utility that executes some basic checks against the Informer installation.

Informer Server Check is installed as part of the Informer Tools package, which can be installed by unzipping `informer-tools-5.2.0.zip`

The Tools can be found alongside the Informer installation binaries: link:https://support.interlocsolutions.com/projects/informer-server/files

Informer Server Check can be run via the `servercheck.bat` file in the extracted directory.

You can run the Informer Server Check from any machine with a Java 1.6 or later runtime that has connectivity to the Informer Maximo environment.

To run the Informer Server Check utility, execute the following at a command prompt:

```
> servercheck.bat
```

Configuring the Tool

Configuring via Properties File

`servercheck.properties` is a self-documented file with the list of parameters you can use, with reasonable defaults and comments. Modify this file to fit your environment, save the file, and run the `.bat` or `.sh` script again.

```
####################
## Server host info
####################
# Set ssl=true if your Maximo URL starts with https://
host=localhost
ssl=false
port=80
meapath=/meaweb/
#
####################
## Maximo/Informer Credentials
####################
# These can be omitted to skip the login step
# If the password is omitted, but the user specified, the user will be prompted in the
terminal
# These are used to log into Informer, but will also be used to resolve 401
challenges.
user=wilson
password=wilson
####################
## Informer Profile
####################
# The Profile name determines which Informer Profile to log into
# This will only be used if a user is specified; Otherwise, no login attempt is made
profile=INFORMERTESTING
####################
## Maxauth Credentials
####################
# The maxauth parameters are used if you use "native security" on the MIF
# http://www-01.ibm.com/support/docview.wss?uid=swg21575076
# This is not particularly common. Do not mistake this for AD/LDAP security
# MAXAUTH authentication is only enabled if maxauth=true
maxauth=false
maxauthuser=miller
maxauthpassword=miller
####################
## Verbosity
####################
# The verbosity flag increases the amount of data written to the terminal.
# This information is can be useful in the case of an error, but is very "busy" and
more difficult to interpret.
# Please set this flag and rerun your test, if seeking support.
verbose=false
```

Command Line Interface (CLI)

Everything can be accomplished using only the properties file, but the command line flags can be more useful for those most comfortable in CLIs.

Any flag specified via the CLI will supersede any default set in the properties file.

To use the CLI exclusively, delete or rename `servercheck.properties`

The Informer Server Check takes the following parameters:

```
Usage: servercheck.bat (-h|--host) <host> [(-n|--port) <port>] [(-u|--user) <user>]
[(-p|--password) <password>] [(-f|--profile) <profile>] (-m|--meapath) <meapath> [-
l|--ssl] [-a|--acceptallsslcerts] [--maxauth] [--maxauthuser <maxauthuser>] [--
maxauthpassword <maxauthpassword>] [-v|--verbose]

  (-h|--host) <host>
        The maximo host name.

  [(-n|--port) <port>]
        The maximo port number.

  [(-u|--user) <user>]
        The informer user name.

  [(-p|--password) <password>]
        The informer user password.

  [(-f|--profile) <profile>]
        The informer profile.

  (-m|--meapath) <meapath>
        The mea web path. (default: /meaweb)

  [-l|--ssl]
        Connect using SSL

  [-a|--acceptallsslcerts]
        Accept all SSL certificates (Ignore SSL validation errors)

  [--maxauth]
        Use MAXAUTH authentication. This is used when 'mxe.appServerSecurity' is
        not used on the server. When this flag set, a special HTTP header is
        added.

  [--maxauthuser <maxauthuser>]
        The maxauth user name.

  [--maxauthpassword <maxauthpassword>]
        The maxauth password.

  [-v|--verbose]
        Verbose mode
```

Basic Version Check

Execute the utility specifying only the –host parameter. This will connect to the Informer service and display the Informer version number.

```
> servercheck.bat -h 192.168.253.10
Parsing arguments
 Using URL: http:// 192.168.253.10:80/meaweb/services/NOTIFY
Performing Version Check
 Server Version: 5.3.0
Checks Complete
```

Login and Profile Access Check

This check requires a profile to be created and a user given access to that profile.

For example, to verify a profile named INFORMERTESTING with a user name of `wilson` and a password of `wilson`:

```
> servercheck.bat -h localhost -f INFORMERTESTING -u wilson -p wilson
Using java executable in PATH
Parsing arguments
meapath: /meaweb/
 Using URL: http://localhost:80/meaweb/services/NOTIFY
Performing Version Check
Response Code was 200
 Server Version: 5.5.0-hf11
Performing Login Check
Response Code was 200
 Login Result: true
 Session Id: 42d597b5-3db2-4242-a3a4-3366ea981a11
Performing Registered Profiles Check
Response Code was 200
 Registered Profiles:
  INFORMERTESTING
Logging Out
Response Code was 200
Checks Complete
```

External Queue Database Configuration

By default, Informer depends on persistent job queues to route, schedule, and distribute system events and background work. These queues are important for day-to-day operation, but their high rate of change can sometimes burden backup strategies.

As of Informer 5.7.0, it is possible to divert Informer queues and associated tables to a separate database or instance, removing their activity from the main Maximo database.

Why?

Maximo data is generally long-lived and slow-moving. The Maximo data must always be recoverable, and backups are a necessary part of administration. Changes need to be captured and should be replayed into pre-production environments.

Informer queue data, by contrast, is generally short-lived and fast-moving. The data can be regenerated if lost, so backups and data migrations are unnecessary.

These Informer queue tables (and associated tables) are defined as MBOs, and these queue MBOs are used as the default queue persistence mechanism. This reduces the administrative overhead and simplifies the infrastructure considerations for many, and is a reasonable default. The downside of this default is that Informer queue tables are generally backed up and logged with the rest of the Maximo data, and this can become a burden for some customers.

By moving these queue tables to a different database instance, that instance can be selected, tuned, configured, and/or isolated to better suit their high-volatility use case (for example, by disabling backups).

Choosing a Database

Any of the following databases can be used for a standalone Informer queue schema:

- Oracle
- Microsoft SqlServer
- IBM DB2
- MySQL / MariaDB
- PostgreSQL

The database type you choose does not need to match the database type you are using for Maximo.

MySQL / MariaDB

If you want to separate the workload from your existing infrastructure completely, or generally to use a more simplistic (if less familiar) option, MySQL can be used. It is a well-known and well-supported free solution for use on a standalone server.

On Ubuntu 18.04, for example, getting a simple MySQL instance takes little more than

```
sudo apt install mysql-server mysql-workbench
```

And the Informer DB can be created within minutes using the "MySQL Workbench" app which that command installed.

General administration of MySQL is outside the scope of Informer documentation and support, but for those comfortable with the idea of adding a new database type to their infrastructure, MySQL can be a good option.

See Appendix: Using an external MySQL Database for Informer Queues

PostgreSQL

PostgreSQL is now also supported, with similar considerations as with MySQL.

See Appendix: Using an external PostgreSQL Database for Informer Queues

Oracle, SQLServer, DB2

If you want to leverage your existing infrastructure and in-house expertise, or for the sake of familiarity and consistency, you may choose to use the same DB type you are using for Maximo. All three choices of DBMS supported by core Maximo are also supported by Informer.

See Appendix: Using an external Oracle Database for Informer Queues

See Appendix: Using an external SqlServer Database for Informer Queues

See Appendix: Using an external DB2 Database for Informer Queues

Appendix

The remaining content is provided for reference purposes, and is not part of the directed installation guide.

Appendix: Maximo Properties

System Properties

Declared Properties

The following Maximo System Properties will be added as part of the first start-up. Default values will be provided, but should be reviewed based on the environment.

Property Name	Description	Default Value	Comment
informer.apns.sandbox	Should the system use the Sandbox APNS push system.	0	If the Apple Push Notification Service (APNS) is being used for iOS devices, and this is a development environment, this should be set to true. This will route the push notifications through Apple's sandbox (development) servers instead of the production servers. If the value of this property is something other than true, or the property does not exist the production APNS servers will be used.
informer.attacheddocument.provider	The fully qualified class name of the document provider.		
informer.catalog.queue.JobDispatcher	Customizable implementation of Catalog job dispatcher	com.interlocsolutions.maximo.notify.queue.db.JobDispatcherDBImpl	The Job Dispatcher represents a policy on the "where", "when", and "who" of Catalog evaluation.
informer.catalog.queue.JobFactory	Customizable implementation of Catalog job factory	com.interlocsolutions.maximo.notify.queue.db.JobFactoryDBImpl	The Job Factory represents a set of definitions of the "what" and "how" to evaluate Catalog evaluation, independent of "where", "when", and"who".
informer.catalog.queue.commitbatchsize	The number of commits to make in one go when expanding aggregate Catalog jobs.	500	
informer.catalog.queue.softcap	The number of records to aim for in the catalog refresh queue.	-1	When set to -1, no cap is enforced. See also: informer.catalog.queue.softcap.wait

Property Name	Description	Default Value	Comment
informer.catalog.queue.softcap.wait	The duration in milliseconds to wait between checks of the catalog queue size.	18000	Only relevant when informer.catalog.queue.softcap is enabled (>0).
informer.catalog.queue.ordering (5.5.1+)	Ordering of Catalog jobs. Valid options include `priority', `fifo', and `none'.	priority	`priority' is recommended for system health, but `none' can sometimes be faster if that is your sole concern
informer.catalog.xml.store	Flag to enable caching of Catalog data at evaluation time, for use during delivery	0	
informer.compression.threshold	The message size in bytes before a message should be compressed.	4096	If this is not set or is not present then the default value of 4096 is used.
informer.createasset	Create an asset for each device that registers with Informer.	0	If this value is set then `informer.createasset.loc` and `informer.createasset.site` should also be provided at a minimum so that asset can be created, assets can only be created in one site and location and can then be moved later by an administrator.
informer.createasset.item	Rotating item number to create new Informer device assets with.		This is optional and only required if the asset should be created as a rotating item.
informer.createasset.loc	The location where new Informer device assets will be created.		
informer.createasset.site	The site that the asset will be created.		
informer.createasset.status	The status that new Informer device assets will be created with.		
informer.createsrtype	The SR type that the service request should be created as.	SR	This defines the type of service request that will be created, this may be an Incident or other server type.
informer.deploy.commtemplate	Communication template to use when sending application links.		If not set then the default template"INFORMER" will be created and used.

Property Name	Description	Default Value	Comment
informer.doctype.autocreate	Auto create document types.	1	If this is set then document types for incoming attached documents will be created if the document type does not exist.
informer.eventfilter	Customizable implementation of FilterFactory to apply to Informer-related data MBO events	com.interlocsolutions.maximo.notify.filter.BaseFilterFactory2	Change it to `com.interlocsolutions.maximo.notify.filter.AttributeFilterFactory` to make use of the advanced filtering features.
informer.false.value	The value that will be returned for false boolean values.	0	
informer.getattacheddocument.validator	GetAttachedDocument Validator classname for Informer sessions		
informer.ignoreLoginCase	Informer will ignore case on login credentials.		This setting only affects Informer's case insensitivity. The actual authentication mechanism (e.g. LDAP) must also be case-insensitive for this setting to have an effect.
informer.lang.whitelist.csv (5.5.1+)	A comma-separated list of language codes Informer should support. If blank, all are supported.		If some language are used in the broader Maximo system, but are not used in Informer applications, Informer efficiency can be improved by specifying only the ones in use, allowing Informer to skip the others.
informer.lbs.populate	Populate the LBS Location table.	0	
informer.login.recovery.commtemplate	The email login recovery communication template.		
informer.max.QueueBatchSize	The maximum number of tasks a queue processor will claim at time before processing.	100	This is an upper bound on a random range. Batching reduces contention among processing threads when using the Database-backed queues (ISNOTIFYREFRESHQUEUE and ISCATALOGREFRESHQUEUE)
informer.notify.xml.includecommanddata	Flag to include Command data in Notification XML	0	This is an uncommonly-used feature. Enabling it has some performance tradeoffs.

Property Name	Description	Default Value	Comment
informer.onerror.createsr	Create a new service request when an Informer client error occurs.	0	This will automatically create an SR every time an error occurs. This should be used in conjunction with the `informer.onerror.wfprocess` property to ensure that errors are identified as the error will no longer be displayed in the Informer application.
informer.onerror.wfprocess	The workflow process name to launch after creating the SR.		If this is blank or if the workflow process named does not exist then the SR will be created but it will not be routed anywhere.
informer.pregen.maxthreads	The maximum number of threads to run when pregenerating a catalog.	2	
informer.public.getattacheddocument.validator	GetAttachedDocument Validator classname for public Informer sessions		
informer.public.group	The group that public registered users will be placed in.	EVERYONE	
informer.public.sendfile.allowedFileExtensions	Specified class defined allowed file extensions for public/anonymous sessions	jpg,png,gif,mp4,mov,m4v,pdf,doc,docx,xls,xlsx,ppt,pptx,xml,txt,csv	
informer.public.sendfile.validator	The classname of the validator to use for files sent via public/anonymous Informer sessions	com.interlocsolutions.maximo.notify.security.datavalidator.InformerAllowedExtensionsValidator	
informer.public.site	The default insert site for new public users.		
informer.push.relayurl	The relay server for push messages.	http://localhost:8080/push	The Push Relay installer installs a Windows Service with a Jetty server running on port 8080.

Property Name	Description	Default Value	Comment
informer.query.owners	A comma delimited list of users whose private queries may be used.		
informer.queue.logError	Flag to force logging of all queue errors.	0	
informer.queue.maxerror	The maximum number of retries for a failed queued item.	5	The refresh queue processors handle most errors and retry automatically. If an error is encountered that is unexpected, this dictates the maximum number of times it will retry before marking the record in error.
informer.queue.pool.catalog	Allows a different value to be set for catalog queue processors.	2	See Cluster Configuration
informer.queue.pool.notification	Allows a different value to be set for the notification queue processors.	3	See Cluster Configuration
informer.queue.pool.push	Specifies the number of push queue processing threads to run on a specific JVM node.	1	See Cluster Configuration
informer.queueMonitor.refreshInterval	The interval at which the queue monitors refresh in seconds.	10	Raise this value reduces database hits, at the cost of UI responsiveness in the Monitor dialog of the Informer Develop and Informer Administration Maximo Applications.
informer.report.defaultprinter	The default printer to print reports		
informer.security.ldap.servers	The number of ldap servers to check for authentication.	1	If the value is greater than 1, then the "ldap" portion of the `informer.security.ldap` properties will include the server number. For example, with 2 servers: `informer.security.ldap.lookup.user`, `informer.security.ldap2.lookup.user`.
informer.security.ldap.authtype	The type of authentication to use.	simple	Used by the Informer's provided LDAPAuthenticator implementation (not enabled by default)

Property Name	Description	Default Value	Comment
informer.security.ldap.context.factory	The fully qualified connection factory class name.		Used by the Informer's provided LDAPAuthenticator implementation (not enabled by default)
informer.security.ldap.lookup.password	Password for the lookup user.		Used by the Informer's provided LDAPAuthenticator implementation (not enabled by default)
informer.security.ldap.lookup.user	The lookup user DN.		Used by the Informer's provided LDAPAuthenticator implementation (not enabled by default)
informer.security.ldap.scope	The scope of the search: SUBTREE, OBJECT, ONELEVEL		Used by the Informer's provided LDAPAuthenticator implementation (not enabled by default)
informer.security.ldap.search.path	The LDAP search path.		Used by the Informer's provided LDAPAuthenticator implementation (not enabled by default)
informer.security.ldap.search.pattern	The user search pattern, such as `uid={0}`		Used by the Informer's provided LDAPAuthenticator implementation (not enabled by default)
informer.security.ldap.url	The full URL, including protocol and port for the LDAP server.		Used by the Informer's provided LDAPAuthenticator implementation (not enabled by default)
informer.security.ldap.user.pattern	A pattern for performing a simple username replace bind.		Used by the Informer's provided LDAPAuthenticator implementation (not enabled by default)
informer.sercurity.ldap.searchbase	The base path for performing searches.		Used by the Informer's provided LDAPAuthenticator implementation (not enabled by default)
informer.session.timeout	The timeout value in minutes for Informer sessions	10	Defaults to 60,000 milliseconds or 10 minutes. If this property is not set or not present the 10 minute default is still used.
informer.stats	Indicates if Informer is tracking statistics.	false	Global toggle for Informer statistics.

Property Name	Description	Default Value	Comment
informer.stats.catalog	Indicates if Informer is tracking statistics for catalogs.	true	Toggle for Catalog statistics. Only takes effect when informer.stats is true.
informer.stats.notification	Indicates if Informer is tracking statistics for notifications.	true	Toggle for Notification statistics. Only takes effect when informer.stats is true.
informer.stats.push	Indicates if Informer is tracking statistics for push notifications.	true	Toggle for Push statistics. Only takes effect when informer.stats is true.
informer.stats.retrieval	Indicates if Informer is tracking retrieval times.	true	Toggle for data retrieval statistics (observations to delivery). Only takes effect when informer.stats is true.
informer.stopOnRMIShutdown	Flag to indicate the system should stop on RMI loss.	0	Recent versions of Informer are not dependent on RMI.
informer.system.deploy.template	Communication template for deploying applciations.	INFORMER	
informer.system.deploy.url	The full URL minus the application name for the deployment URL.		If not set then the current application URL will be used when the deployment email is sent, this allows that to be overridden to provide a public or other URL to access the application download.
informer.true.value	The value that will be returned for true boolean values.	1	The value Informer understands to represent truth. This is a reflection of Maximo's YORN abstraction. Expected values are 1 or "true".
informer.twilio.accountsid	The Twilio account SID.		
informer.twilio.authtoken	The Twilio authentication token.		
informer.twilio.fromphone	The Twilio from phone number.		
informer.twilio.login.recovery.msg	The message that will be sent from Twilio for login recovery, must contain {0} for replacement.		

Property Name	Description	Default Value	Comment
informer.twilio.verify.msg	The message that will be sent from Twilio for verification, must contain {0} for replacement.		
informer.update.serialnum.off	Turns off serial number update if true	0	The automatic updating of serial numbers is a feature meant to reduce the number of obsolete device records when devices change IDs. This is primarily an issue on iOS, which will get a new app-specific serial number every time the app is uninstalled and reinstalled (as opposed to upgraded). As of Informer 5.5, it is recommended to simultaneously set
informer.verification.commtemplate	The email verification communication template.		
informer.catalog.queue.QueueProcessor	Specifies the catalog queue processor to use.	`com.interlocsolutions.maximo.notify.queue.db.BulkByJvmCatalogJobQueueProcessor`	Possible options are: `com.interlocsolutions.maximo.notify.queue.catalog.DisabledCatalogJobQueueProcessor`, `com.interlocsolutions.maximo.notify.queue.db.BulkByThreadCatalogJobQueueProcessor`, `com.interlocsolutions.maximo.notify.queue.db.BulkByJvmCatalogJobQueueProcessor` Disabled means disabled. BulkByThread is the old behavior where many threads on each JVM attempt to claim records to process. This can lead to deadlocks. BulkByJvm is the new behavior where a single thread reads records from the queue and passes them off to separate worker threads.

Property Name	Description	Default Value	Comment
informer.notification.queue.QueueProcessor	Specifies the notification queue processor to use.	`com.interlocsolutions.maximo.notify.queue.notification.BulkByJvmNotificationJobQueueProcessor`	Possible options are: `com.interlocsolutions.maximo.notify.queue.notification.DisabledNotificationJobQueueProcessor`, `com.interlocsolutions.maximo.notify.queue.notification.BulkByThreadNotificationJobQueueProcessor`, `com.interlocsolutions.maximo.notify.queue.notification.BulkByJvmNotificationJobQueueProcessor` Disabled means disabled. BulkByThread is the old behavior where many threads on each JVM attempt to claim records to process. This can lead to deadlocks. BulkByJvm is the new behavior where a single thread reads records from the queue and passes them off to separate worker threads.
informer.push.queue.QueueProcessor	Specifies the push queue processor to use.	`com.interlocsolutions.maximo.notify.push.BulkByJvmPushJobQueueProcessor`	Possible options are: `com.interlocsolutions.maximo.notify.push.DisabledPushJobQueueProcessor`, `com.interlocsolutions.maximo.notify.push.BulkByThreadPushJobQueueProcessor`, `com.interlocsolutions.maximo.notify.push.BulkByJvmPushJobQueueProcessor` Disabled means disabled. BulkByThread is the old behavior where many threads on each JVM attempt to claim records to process. This can lead to deadlocks. BulkByJvm is the new behavior where a single thread reads records from the queue and passes them off to separate worker threads.
informer.userIdentifier	Class to map a user identifier to a Maximo user account.	`com.interlocsolutions.maximo.notify.security.DefaultUserIdentifier`	The specified class must implement `com.interlocsolutions.maximo.notify.security.UserIdentifier`. This class supposes to find Maximo's `USERID` by app provided username. The default class is looking up Maximo's `USERID` by Maximo `LOGINID`(User Name).
informer.authenticator.oidcuniqueidentifier	Unique identifier that represent the user's login id in OIDC userinfo.	`email`	In case of using `OIDCAuthenticator`, it will be used to find user identity from OIDC userinfo endpoint which needs to be match with user's login ID that app will provide.

Property Name	Description	Default Value	Comment
informer.authenticator.oidcuserinfoendpoint	Unique identifier the OIDC userinfo endpoint URL.		In case of using `OIDCAuthenticator`, it will be used if we want to Informer verify the user's token by a specific endpoint. If it is empty, then that endpoint will be identified via the configuration end point which specified by `informer.authenticator.oidcconfigurationendpoint` system property.
informer.authenticator.oidcconfigurationendpoint	Unique identifier the OIDC userinfo configuration URL.	`https://login.microsoftonline.com/common/v2.0`	In case of using `OIDCAuthenticator`, it will be used to find the OIDC configuration.
informer.dfltuser	Informer integration default login user.		It can be used to introduce a user, If the Informer app needs to provide that user credential via its requests `MAXAUTH` or `Authorization` header, to pass the Maximo integration security which has differed from the actual user that is supposed to log in to the app.
informer.commandHistory.xml.store	Flag to enable saving of command history payload XML.	0	By enabling this flag, Informer stores all command executions' payload (XML) in `ISCOMMANDHISTORY` table. (If the app specfied that command execution history needs to be saved.)

Undeclared Properties

The following properties are recognized by Informer, but are not automatically created during the first start-up. All are optional, and cover more fringe needs.

Property Name	Description
informer.authenticator	An implementation of `com.interlocsolutions.maximo.notify.security.Authenticator` to provide user credential authentication
informer.instrumentcollector	Fully-Qualified name of an alternative class to collect data from an instrumentation reading sent by a device. Must implement `com.interlocsolutions.maximo.notify.InstrumentCollector`.
informer.keepinstrumenthistory	Flag to indicate if instrumentation history should be retained. 0 or false indicate that instrumentation history should not be kept, 1 or true indicates it will be retained. The instrumentation history is only available through the database or reports and is stored in the ISDEVICEINSTRUMENTHISTORY table.
informer.smtp.host	Informer-local override of the mail.smtp.host property. Used by the PRINT_REPORT Informer Command.

Property Name	Description
informer.smtp.password	Informer-local override of the mxe.smtp.password property. Used by the PRINT_REPORT Informer Command.
informer.smtp.tlsPort	If specified, use the specified non-standard TLS port. Only applies when `informer.smtp.useTLS` is `true`. Used by the PRINT_REPORT Informer Command.
informer.smtp.useTLS	If "1" or "true", communicate with the SMTP server using TLS. Used by the PRINT_REPORT Informer Command.
informer.smtp.user	Informer-local override of the `mxe.smtp.user` property. Used by the PRINT_REPORT Informer Command.
informer.sqlite.retain	This property is intended for development debugging purposes, because the files generated will be sizable and will accumulate over time. When "true",SQLite databases created during preload will be deposited in the system's temp folder (such as /tmp on Linux systems). The file is of the form: `catalog_<CATALOGID>_<RANDOM_CHARS>.db`

HF Properties

`informer.hf.*` properties are "Hotfix" properties, which exist to bridge a short-term gap until the next full release of Informer.

HF properties, if present, are short-lived; For example, if an HF property is introduced in Informer 5.5.0-hf1, it is typically removed in 5.5.1 (when it has been rendered unnecessary).

PB Properties

`informer.pb.*` properties are "Probationary" properties, which exist to enable pre-release or experimental features and behaviors.

These are not publicly documented, and are subject to change or removal between feature releases.

Dev Properties

The following properties are not defined automatically, and are not intended for normal Production use.

These properties are for use by developers, debuggers, and/or power users. The adjustments of behavior follow a theme of eliminating safeguards, automatic behavior, and/or optimizations. These share a general goal of easing the development process, in which rapid changes are commonplace.

Property Name	Description
informer.dev.noreconcileonactivation	Intended for use during active Profile development, when the speed of deactivating and activating the profile is more important that data accuracy.

JVM Parameters

The following System properties control Informer processing threads run and if so how many will run on a given node.

Property Name	Description	Default Value
informer.queue.pool.notification	Specifies the level of concurrency for Notification queue processing. Informer has three type processors for Notification processing. The first number represents the number of"User Expansion" processors, the second represents the number of"User Evaluator" processors, and the third represents the number of "Record" processors. A value of 0 in any position disables that type of processor on the JVM. A setting of "0" (no comma separators) disables all types of Notification queue processors on the JVM.	3
informer.queue.pool.catalog	Specifies the level of concurrency for Catalog queue processing. Informer has two types of processors for Catalog processing. The first number represents the number of "Aggregate" processors, and the second one represents the number of "Record" processors. A value of 0 in any position disables that type of processor on the JVM. A setting of "0" (no commas separators) disables all types of Catalog queue processors on the JVM.	2
informer.queue.pool.push	Specifies the number of "Push" queue processing threads to run on a specific JVM node. If this is set to "0" then no push queue threads will run. **NOTE** Any given node can have a maximum of "1" push queue processor.	1

Appendix: Informer Cron Tasks

Informer uses a number of Maximo CronTasks to maintain the system health and operation. These CronTasks are added to the system on install, but most are not configured to run by default. An administrator **must** review and configure each of the following CronTasks.

CatalogCleanup

Cleans up outdated Catalog information. Informer may defer deletion of obsolete or orphaned Catalog-related records for reasons of responsiveness and daytime operational efficiency. This CronTask sweeps up at night.

This is the Catalog counterpart to NotificationCleanup.

Instance Configuration

Required for all deployments; Do not delete all instances.

A default instance is automatically defined by Informer. You may change the timing, but this Cron should run at least weekly.

Parameters

There are no parameters to configure for this CronTask.

CatalogPreload

The CatalogPreload CronTask will evaluate all available catalogs that are marked for preload and regenerate the preload pages for all catalogs that have exceeded the update threshold defined on the Catalog record.

Instance Configuration

This CronTask is essential to server performance when delivering a Catalog in whole (such as to a new or long-dormant device).

The generation of new preloads can sometimes be intensive if the Catalog is large, this work should therefore be scheduled for periods of low system activity if possible.

A default instance is automatically defined by Informer. A very conservative default of once-per-week is applied by the system, but consider making this more frequently (even nightly). Note that this CronTask will do nothing on Catalogs which do not exceed their thresholds, so scheduling a higher frequency for this CronTask does not necessarily mean more work.

Parameters

There are no parameters to configure for this CronTask.

CatalogPurge

This CronTask is deprecated, and should no longer be used. It has been superseded by the CatalogCleanup CronTask.

CatalogUpdate

This CronTask is deprecated, and should no longer be used. It has been superseded by the `ReconcileCatalog` CronTask.

CommandHistoryPurge

Informer retains a set of information on each Command executed, including a hash of the request, and the response to deliver. These records allow the server to:

- Identify duplicate requests and only execute them once (Idempotence)
- Cache responses for devices who failed to complete the Command round-trip (for example, due to a network interruption)
- Facilitate notification to the issuing a device when a Command it requested was resolved server side (via the Error Management tab in either Informer Maximo app)
- Provide historical/logging information for statistics

Although this information is very useful, over time this can become a significant number of records with diminishing value. The command history purge CronTask provides a mechanism to purge historical command information that has exceeded a specified retention period.

Instance Configuration

Optional, but generally recommended.

Without this CronTask, the Command history with grow forever. An infinite log is not of functional value to most customers, so this CronTask serves as a way to clean up records which are far too old to be useful.

Even running this monthly, with a 6-month retention period, is enough to say that the table will not grow forever.

Parameters

Name	Description	Default Value
RetentionPeriod	The command history retention period in days.	30

DeviceCleanup

This CronTask cleans up expired Device registrations.

Informer will keep track of certain information about every Device which synchronizes with it. Should a device become unused, replaced, reset, etc, then Informer will never hear from that device again, yet is still retaining information about that device indefinitely. This CronTask cleans up those obsolete records.

For devices which have not logged in for a certain number of days, all device-related Information is cleared.

One of the data points that Informer tracks for each device is which Catalogs they have synchronized, and how up-to-date the device's copy of that Catalog is. Legacy Catalog APIs (driven by Informer client version in the app) use this information to determine what to delivery to the devices. Should a device be "too far behind" in is synchronization of a large, fast-moving Catalog, then that registration will be reset, causing that particular device to leverage preloads to synchronize that particular Catalog upon next login.

The Catalog delta-counting cleanup is a more constrained cleanup for those records which out outdated for one or more Catalogs, yet have still been active within the configured timeout period (e.g. The device's user goes on vacation for 2 weeks). *This is especially important when using the legacy Catalog APIs, still used by some Informer clients at time of this writing.*

Instance Configuration

Recommended for all deployments. No default instance is defined.

Your use-case and tolerance for extraneous records dictate how permissive your parameters are.

Having some bound, at any setting, is recommended. Otherwise, the related tables can experience unbounded growth over time.

A weekly execution schedule is typically sufficient, but more frequent execution is harmless.

Parameters

Name	Description	Default Value
daysInactiveThreshold	Number of days since last login, before Informer should stop tracking this device.	30
catalogDeltaThreshold	Number of unsynchronized records a device can be behind for a given Catalog, before a reset of that particular Catalog is triggered for that device.	20000

InformerMonitorCleanup

When Informer Monitor threads run, they report the status of the processes they are monitoring to the database for display across JVMs in the system.

If a JVM goes down and does not return, this Cron ensures that stale data does not linger.

Instance Configuration

A default instance is automatically defined by Informer. You may change the timing, frequency, and parameters, but keeping an instance of this configured is always recommended.

Parameters

Name	Description	Default Value
outdatednessDuration	The outdatedness in (units) that monitor info must be before it is considered abandoned and purged.	1
outdatednessUnit	The time unit that the specified outdatednessDuration represents	HOURS

InformerStats

When Informer statistics tracking is enabled via system properties, data is collected as the system runs, but is not compiled into reports for human consumption. The InformerStats CronTask serves this purpose at regular intervals. Whenever this task is run, statistics are compiled from recent logged events, and the new report is saved as its own compiled record.

A lot of raw performance data is generated as the system runs, primarily for this CronTask to evaluate and compile. It is therefore the responsibility of this task to also clean up uncompiled records which have outlived their usefulness.

Instance Configuration

Only use this CronTask when also using the Informer "stats" feature.

This feature is **not** recommended for Production use, due to its overhead and complexity. It primarily serves as a developer's debug and analysis tool during development, issue investigation, and proof-of-concept phases.

Parameters

Name	Description	Default Value
RetentionPeriod	The duration in days that uncompiled stats will be retained.	7

NotificationCleanup

Cleans up outdated Notification information. Informer may defer deletion of obsolete or orphaned Notification-related records for reasons of responsiveness and daytime operational efficiency. This CronTask sweeps up at night.

This is the Notification counterpart to CatalogCleanup.

Instance Configuration

Required for all deployments; Do not delete all instances.

A default instance is automatically defined by Informer. You may change the timing, but this Cron should run at least weekly.

Parameters

There are no parameters to configure for this CronTask.

ReconcileCatalog

This CronTask will trigger a reconciliation of the specified Catalogs whenever it is executed.

This can be used for eventual recognition of objects which do not reliably fire MBO events, or for Catalogs with "sliding window"(date-based) selection criteria in their where-clauses.

Forces a periodic reconciliation of the catalog content with the current result set as defined by the where clause.

Instance Configuration

Applicability of this CronTask is completely dependent on the needs, design, and constraints of the Informer Profile(s) it serves.

Parameters

Name	Description	Default Value
ProfileName	The name of the profile that the listed catalogs are part of.	
CatalogNames	A comma delimited list of catalogs to reconcile for the specified profile.	

SessionExpiry

Informer creates and manages its own session tokens for devices logging in through its API. This CronTask runs frequently, cleaning up state information for expired Sessions, which no longer require it.

The activities of this CronTask do NOT affect when Sessions become invalid, instead only conducting cleanup of the Sessions which have *already* expired.

Note that this CronTask does not remove records from the more permanent, less stateful ISSESSIONHISTORY table. See SessionPurge.

Instance Configuration

Required for all deployments; Do not delete all instances.

An instance of this CronTask is created automatically by Informer.

Parameters

There are no parameters to configure for this CronTask.

SessionPurge

The Informer platform indefinitely retains session history information for each session that is created by a client device. This information is very useful when attempting to diagnose connectivity issues or analyze usage patterns. However, over time this can become a significant number of records with diminishing value. The SessionPurge purge CronTask provides a mechanism to purge historical session information that has exceeded a specified retention period.

Instance Configuration

Optional, but generally recommended.

Without this CronTask, the Session history with grow forever. An infinite log is not of functional value to most customers, so this CronTask serves as a way to clean up records which are far too old to be useful.

Even running this monthly, with a 6-month retention period, is enough to say that the table will not grow forever.

Parameters

Name	Description	Default Value
sessionRetPeriod	The session history retention period in days.	30

Appendix: Maximo Loggers

During installation, Informer creates a few new Maximo loggers which can be configured like any other. You may wish to review the created loggers and configure as appropriate for your environment and the circumstances of the moment.

Logger	Default Level	Description
informer	WARN	Informer logging which does not fall into the more specific categories
informer.command	WARN	Logs pertaining to the handling of Informer Commands (user actions via devices)
informer.nrqp	WARN	Logs pertaining to the "Notification Refresh Queue Processors" and their associated listeners and tasks
informer.crqp	WARN	Logs pertaining to the "Catalog Refresh Queue Processors" and their associated listeners and tasks

Appendix: Informer Installer Details

The following information is for informational purposes only.

informer-install.bat

The installer attempts to establish a connection to the specified database and then inserts new records in the MAXSERVICE table with the following values:

Attribute	Value
SERVICENAME	NOTIFY
DESCRIPTION	Notification Service
CLASSNAME	com.interlocsolutions.maximo.notify.NotifyService
MAXSERVICEID	Calculated to be the next in the sequence
INITORDER	1010
INTERNAL	0
ACTIVE	1

Attribute	Value
SERVICENAME	NOTIFYINFO
DESCRIPTION	Notification Info
CLASSNAME	com.interlocsolutions.maximo.notify.NotifyInfoService
MAXSERVICEID	Calculated to be the next in the sequence
INITORDER	1050
INTERNAL	0
ACTIVE	1

Attribute	Value
SERVICENAME	NOTIFYPUB
DESCRIPTION	Notification Public Service
CLASSNAME	com.interlocsolutions.maximo.notify.NotifyPublicService
MAXSERVICEID	Calculated to be the next in the sequence
INITORDER	1020
INTERNAL	0
ACTIVE	1

Attribute	Value
SERVICENAME	NOTIFYCLASSES

Attribute	Value
DESCRIPTION	Interloc Informer ClassLoader Service
CLASSNAME	com.interlocsolutions.maximo.notify.classloader.InformerClassesService
MAXSERVICEID	Calculated to be the next in the sequence
INITORDER	900
INTERNAL	0
ACTIVE	1

In addition to the database entries the installer also adds a Servlet entry to the `<maximo_home>\applications\maximo\maximouiweb\webmodule\WEB-INF\web.xml` and updates the `<maximo_home>\applications\maximo\meaweb\webmodule\WEB-INF\web.xml` with a comment that can removed to ensure that the NOTIFYINFO service is left unauthenticated in a secured environment.

The `<maximo_home>\applications\maximo\maximouiweb\webmodule\WEB-INF\web.xml` file has the following entries added:

Informer Servlets

```
<servlet>
    <description>Informer Servlet</description>
    <servlet-name>NotifyExport</servlet-name>
    <servlet-class>
com.interlocsolutions.maximo.webclient.servlet.NotifyServlet</servlet-class>
</servlet>
<servlet>
    <description>Informer Application Deployment Servlet</description>
    <servlet-name>NotifyAppDeploy</servlet-name>
    <servlet-class>
com.interlocsolutions.maximo.webclient.servlet.AppDeployServlet</servlet-class>
</servlet>
```

and

Informer Servlet Mappings

```
<servlet-mapping>
    <servlet-name>NotifyExport</servlet-name>
    <url-pattern>/notifyexport/*</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>NotifyAppDeploy</servlet-name>
    <url-pattern>/informer/apps/*</url-pattern>
</servlet-mapping>
```

Installing the Interloc Mobile Informer to the target Maximo installation folder also adds a `product.xml` file named `isinformer.xml` to the `<maximo_home>\deployment\product` folder. This file

ensures that the Java JAR files required by the installer are included in the Maximo class path.

Appendix: Using an external Oracle Database for Informer Queues

To set up an Oracle Informer external database, you need a new dedicated Oracle database instance with a user that has DDL and DML access. All data contained within this database will be temporary and can be regenerated in a disaster recovery scenario, so we recommend disabling any logging, archiving, or recovery features for this new instance.

Then, for creating the Informer data model, you can run the Informer DB setup bash file (Linux: `informer-dbsetup.sh`) or batch file (Windows: `informer-dbsetup.bat`) which can be found in the Maximo tools folder (e.g. Linux: `/opt/IBM/SMP/maximo/tools/maximo/` Windows: `C:\IBM\SMP\maximo\tools\maximo`) and an OS command terminal to execute it.

You can see the Informer DB setup script's manual by executing it with the `--help` parameter like this:
Linux: `./informer-dbsetup.sh --help`
Windows: `.\informer-dbsetup.bat --help`

For executing that script to create the Informer DB, you need at least three parameters:

- JDBC URL that should follow this syntax: https://docs.oracle.com/cd/B28359_01/java.111/b31224/urls.htm#BEIJFHHB (For thin driver)
  The simplest form would be: `jdbc:oracle:thin:@//${hostname}:${port}/${service_name}`
- DB username
- DB password

To create the Informer data model in this database, execute the `informer-dbsetup` script like so:
Linux: `./informer-dbsetup.sh ${jdbcurl} -u ${dbuser}`
Windows: `.\informer-dbsetup.bat ${jdbcurl} -u ${dbuser}`

After setup Informer DB successfully and also install Informer you need to set three Maximo properties:

- `informer.db.queue.url` → `${jdbcurl}`
- `informer.db.queue.user` → `${dbuser}`
- `informer.db.queue.pw` → `${dbpassword}`

At the end, for finalizing the installation you need to restart Maximo.

Appendix: Using an external SqlServer Database for Informer Queues

To set up an SQL Server Informer external database, you need a new dedicated SQL Server database with a user that has DDL and DML access. All data contained within this database will be temporary and can be regenerated in a disaster recovery scenario, so we recommend disabling any logging, archiving, or recovery features for this new instance.

Then, for creating the Informer data model, you can run the Informer DB setup bash file (Linux: `informer-dbsetup.sh`) or batch file (Windows: `informer-dbsetup.bat`) which can be found in the Maximo tools folder (e.g. Linux: `/opt/IBM/SMP/maximo/tools/maximo/` Windows: `C:\IBM\SMP\maximo\tools\maximo`) and an OS command terminal to execute it.

You can see the Informer DB setup script's manual by executing it with the `--help` parameter like this:
Linux: `./informer-dbsetup.sh --help`
Windows: `.\informer-dbsetup.bat --help`

For executing that script to create the Informer DB, you need at least three parameters:

- JDBC URL that should follow this syntax: https://docs.microsoft.com/en-us/sql/connect/jdbc/building-the-connection-url?view=sql-server-2017
  The simplest form would be: `jdbc:sqlserver://${hostname}:${port};databaseName=${dbname}`
- DB username
- DB password

To create the Informer data model in this database, execute the `informer-dbsetup` script like so:
Linux: `./informer-dbsetup.sh '${jdbcurl}' -u ${dbuser}`
Windows: `.\informer-dbsetup.bat '${jdbcurl}' -u ${dbuser}`

After setup Informer DB successfully and also install Informer you need to set three Maximo properties:

- `informer.db.queue.url` → `${jdbcurl}`
- `informer.db.queue.user` → `${dbuser}`
- `informer.db.queue.pw` → `${dbpassword}`

At the end, for finalizing the installation you need to restart Maximo.

Appendix: Using an external DB2 Database for Informer Queues

To set up a DB2 Informer external database, you need a new dedicated DB2 database with a user that has DDL and DML access.
All data contained within this database will be temporary and can be regenerated in a disaster recovery scenario, so we recommend disabling any logging, archiving, or recovery features for this new instance.

Then, for creating the Informer data model, you can run the Informer DB setup bash file (Linux: `informer-dbsetup.sh`) or batch file (Windows: `informer-dbsetup.bat`) which can be found in the Maximo tools folder (e.g. Linux: `/opt/IBM/SMP/maximo/tools/maximo/` Windows: `C:\IBM\SMP\maximo\tools\maximo`) and an OS command terminal to execute it.

You can see the Informer DB setup script's manual by executing it with the `--help` parameter like this:
Linux: `./informer-dbsetup.sh --help`
Windows: `.\informer-dbsetup.bat --help`

For executing that script to create the Informer DB, you need at least three parameters:

- JDBC URL that should follow this syntax: https://www.ibm.com/support/knowledgecenter/en/SSEPGG_11.1.0/com.ibm.db2.luw.apdv.java.doc/src/tpc/imjcc_r0052342.html
  The simplest form would be: `jdbc:db2://${hostname}:${port}/${dbname}`
- DB username
- DB password

To create the Informer data model in this database, execute the `informer-dbsetup` script like so:
Linux: `./informer-dbsetup.sh ${jdbcurl} -u ${dbuser}`
Windows: `.\informer-dbsetup.bat ${jdbcurl} -u ${dbuser}`

After setup Informer DB successfully and also install Informer you need to set three Maximo properties:

- `informer.db.queue.url` → `${jdbcurl}`
- `informer.db.queue.user` → `${dbuser}`
- `informer.db.queue.pw` → `${dbpassword}`

At the end, for finalizing the installation you need to restart Maximo.

Appendix: Using an external MySQL Database for Informer Queues

First of all for setup a MySQL Informer external database, you need a new dedicated MySQL database with a user that has DDL and DML access. All data contained within this database will be temporary and can be regenerated in a disaster recovery scenario, so we recommend disabling any logging, archiving, or recovery features for this new instance.

Then, for creating the Informer data model, you can run the Informer DB setup bash file (Linux: `informer-dbsetup.sh`) or batch file (Windows: `informer-dbsetup.bat`) which can be found in the Maximo tools folder (e.g. Linux: `/opt/IBM/SMP/maximo/tools/maximo/` Windows: `C:\IBM\SMP\maximo\tools\maximo`) and an OS command terminal to execute it.

You can see the Informer DB setup script's manual by executing it with the `--help` parameter like this:
Linux: `./informer-dbsetup.sh --help`
Windows: `.\informer-dbsetup.bat --help`

For executing that script to create the Informer DB, you need at least three parameters:

- JDBC URL that should follow this syntax:
  https://dev.mysql.com/doc/connector-j/5.1/en/connector-j-reference-url-format.html
  The simplest form would be: `jdbc:mysql://${hostname}:${port}/${dbname}`
- DB username
- DB password

To create the Informer data model in this database, execute the `informer-dbsetup` script like so:
Linux: `./informer-dbsetup.sh ${jdbcurl} -u ${dbuser}`
Windows: `.\informer-dbsetup.bat ${jdbcurl} -u ${dbuser}`

After setup Informer DB successfully and also install Informer you need to set three Maximo properties:

- `informer.db.queue.url` → `${jdbcurl}`
- `informer.db.queue.user` → `${dbuser}`
- `informer.db.queue.pw` → `${dbpassword}`

At the end, for finalizing the installation you need to restart Maximo.

Appendix: Using an external SQLite Database for Informer Queues

To use SQLite for the external queue database, start by deciding on an appropriate file path complete with filename. A file extension of `.sqlite3` is recommended. (e.g. `/opt/informer/informerq.sqlite3`)

Then, for creating the Informer data model, you can run the Informer DB setup bash file (Linux: `informer-dbsetup.sh`) or batch file (Windows: `informer-dbsetup.bat`) which can be found in the Maximo tools folder (e.g. Linux: `/opt/IBM/SMP/maximo/tools/maximo/` Windows: `C:\IBM\SMP\maximo\tools\maximo`) and an OS command terminal to execute it.

You can see the Informer DB setup script's manual by executing it with the `--help` parameter like this:
Linux: `./informer-dbsetup.sh --help`
Windows: `.\informer-dbsetup.bat --help`

For executing that script to create the Informer DB, you need at least one parameter:

- JDBC URL that should follow this syntax:
  https://github.com/xerial/sqlite-jdbc#how-to-specify-database-files
  The simplest form would be: `jdbc:sqlite:${filepath}`

To create the Informer data model in this database, execute the `informer-dbsetup` script like so:
Linux: `./informer-dbsetup.sh ${jdbcurl}`
Windows: `.\informer-dbsetup.bat ${jdbcurl}`

After setup Informer DB successfully and also install Informer you need to set only one Maximo property:

- `informer.db.queue.url` → `${jdbcurl}`

At the end, for finalizing the installation you need to restart Maximo.

Appendix: Using an external PostgreSQL Database for Informer Queues

To setup a PostgreSQL Informer external database you need a new dedicated PostgreSQL database with a user that has DDL and DML access.

All data contained within this database will be temporary and can be regenerated in a disaster recovery scenario, so we recommend disabling any logging, archiving, or recovery features for this new instance.

Then, to create the Informer data model, run the Informer DB setup bash file (Linux: `informer-dbsetup.sh`) or batch file (Windows: `informer-dbsetup.bat`) which can be found in the Maximo tools folder (e.g. Linux: `/opt/IBM/SMP/maximo/tools/maximo/` Windows: `C:\IBM\SMP\maximo\tools\maximo`) and an OS command terminal to execute it.

You can see the Informer DB setup script's manual by executing it with the `-help` parameter like this:
Linux: `./informer-dbsetup.sh --help`
Windows: `.\informer-dbsetup.bat --help`

To execute the script to create the Informer DB, you need at least three parameters:

- A JDBC URL following this syntax:
  https://jdbc.postgresql.org/documentation/head/connect.html
  The simplest form would be: `jdbc:postgresql://${hostname}:${port}/${dbname}`
- DB username
- DB password

To create the Informer data model in this database, execute the `informer-dbsetup` script like so:
Linux: `./informer-dbsetup.sh ${jdbcurl} -u ${dbuser}`
Windows: `.\informer-dbsetup.bat ${jdbcurl} -u ${dbuser}`

After setup Informer DB successfully and also install Informer you need to set three Maximo properties:

- `informer.db.queue.url` → `${jdbcurl}`
- `informer.db.queue.user` → `${dbuser}`
- `informer.db.queue.pw` → `${dbpassword}`

To finalize the installation, you need to restart Maximo.

Appendix: Additional Pre-Installation Checks when Upgrading from Informer 4.x

MAXMENU

When upgrading to Informer `5.x` from `4.x`, there are duplicate `MAXMENU` entries. Run the following SQL statement before upgrading:

```
DELETE maxmenu WHERE moduleapp = 'NOTIFY';
```

This will resolve the issue.

Clean up Existing Informer Profiles

If upgrading a system which has an existing Informer installation of 5.2 or earlier, please perform the following steps:

Deleting Registered Devices

Informer 5.2+ introduces a new method to track and store catalog data. Therefore, it is necessary to delete any existing device registrations before allowing devices to connect to the newly installed version of Informer. Please note that deleting device records will result in the client device **deleting all stored data, including any pending commands(user actions)** in the transaction queue. For this reason it is important to ensure that all pending commands are sent before this action is performed. To delete device records, please follow these steps:

1. Using the Informer Developer or Informer Administration applications, navigate to the Devices tab.
2. Select the check box next to the device you wish to delete. It is also possible to select more than one device or all devices on the page.
3. Click the "Delete Selected" button
4. Save

NOTE	Deleting the device record also deletes any associated push device registrations so there is no need to take any further action.

When the device attempts to connect (following device deletion) it will recognize that a reset has taken place and will wipe all stored data. Hence, the provisioning of catalogs and notifications will begin.

Deactivating Informer Profiles

To deactivate a profile, the system should be running.

1. Go To Administration > Informer Administration
2. Administration > Informer Developer can also be used
3. In the List view, open any record whose "active" checkbox is marked
4. Select the "Deactivate Profile" action
5. Repeat for each active Profile

More information on profile activation and deactivation can be found in the "Informer Administration" and "Informer Developer" application guides.

Once the profiles are deactivated, shut down the application server.

Clearing Informer Processing Queues

Before clearing the Informer queues, Informer profiles should be deactivated (above), and the application servers should be shut down.

Only then, execute the following statements to immediately and completely clear Informer processing queues.

```
TRUNCATE TABLE isnotifyqueuemonitor;
TRUNCATE TABLE isnotifyrefreshqueue;
TRUNCATE TABLE iscatalogqueuemonitor;
TRUNCATE TABLE iscatalogrefreshqueue;
TRUNCATE TABLE ispushqueuemonitor;
TRUNCATE TABLE ispushqueue;
```

The `ISPUSHQUEUE`, `ISPUSHQUEUEMONITOR`, `ISCATALOGREFRESHQUEUE` and/or `ISCATALOGQUEUEMONITOR` tables may not exist, depending on the version of Informer you are upgrading from. If an error occurs while attempting to truncate, please confirm that these tables exist. If they do not, then there is no issue.

Index Violations (When upgrading from Informer 5.5.0 or earlier)

In versions of Informer before 5.5.0, it was possible to have multiple identical Catalog registrations (for the same device registering for the same revision of the same Catalog), due to timing issues. This resulted in lookup errors, causing affected devices to get inaccurate or missing prompts for Catalog synchronization. A unique index now prevents this possibility, and is installed with Informer. In order for installation to succeed, the current data must not violate this new unique index.

This SQL will have no effect if nothing needs to be corrected. There is no need to run this again during subsequent upgrades.

Run the following SQL while the system is down, before upgrading:

```
-- Note: The below statement is consciously overcomplicated in order to be compatible
with all 3 possible DBMSes Maximo supports
DELETE FROM iscatalogdevice WHERE deviceid IN (
    SELECT iscatalogdevice.deviceid
    FROM iscatalogdevice
    JOIN (
      SELECT deviceid, (COUNT(*) OVER (PARTITION BY DEVICEKEY, CATALOGID, REVISION,
LANGCODE)) AS numcopies
    FROM iscatalogdevice
    ) subq ON iscatalogdevice.deviceid = subq.deviceid AND numcopies > 1
);
```

Index Violations

In versions of Informer before 5.4.0, it was possible to have multiple functionally-identical records in ISCATALOGDATA, which could result in lookup errors. A unique index now prevents this possibility, and is installed with Informer. In order for installation to succeed, the current data must not violate this new unique index.

This SQL will have no effect if nothing needs to be corrected. There is no need to run this again during subsequent upgrades.

Run the following SQL while the system is down, before upgrading:

```
-- Delete violators of ISCATALOGDATA_NDX6
DELETE
FROM iscatalogdata
WHERE catalogdataid IN
  (SELECT catalogdataid
  FROM iscatalogdata cd
  JOIN
    (SELECT catalogid, recordid, MAX(catalogdataid) newestDup
    FROM iscatalogdata
    GROUP BY catalogid, recordid
    HAVING COUNT(*) > 1
    ) cddup
  ON (cd.catalogid        = cddup.catalogid
  AND cd.recordid         = cddup.recordid)
  WHERE cd.catalogdataid <> cddup.newestdup
  );
```

Required Fields

In versions of Informer before 5.3.0, some fields were functionally required, but not enforced in the database configuration. Running these two queries corrects the record, preventing errors during upgrade without sacrificing other forms of validation.

```
UPDATE maxattribute SET required = 1 WHERE required = 0 AND (
 (objectname = 'ISNOTIFYATTRIBUTE' AND attributename = 'ATTRIBUTENAME')
 OR (objectname = 'ISNOTIFYGROUPOPTION' AND attributename = 'NOTIFYGROUPID')
 OR (objectname = 'ISNOTIFYGROUPOPTION' AND attributename = 'NOTIFYOPTIONID')
 OR (objectname = 'ISNOTIFYGROUPUSER' AND attributename = 'NOTIFYGROUPID')
 OR (objectname = 'ISNOTIFYGROUPUSER' AND attributename = 'NOTIFYUSERID')
 OR (objectname = 'ISNOTIFYOBJECT' AND attributename = 'OBJECTNAME')
 OR (objectname = 'ISNOTIFYOPTION' AND attributename = 'NOTIFYID')
 OR (objectname = 'ISNOTIFYOPTION' AND attributename = 'OPTIONNAME')
 OR (objectname = 'ISNOTIFYSEQ' AND attributename = 'SEQUENCE')
 OR (objectname = 'ISNOTIFYVERSION' AND attributename = 'VERSION')
);
UPDATE maxattributecfg SET required = 1 WHERE required = 0 AND (
 (objectname = 'ISNOTIFYATTRIBUTE' AND attributename = 'ATTRIBUTENAME')
 OR (objectname = 'ISNOTIFYGROUPOPTION' AND attributename = 'NOTIFYGROUPID')
 OR (objectname = 'ISNOTIFYGROUPOPTION' AND attributename = 'NOTIFYOPTIONID')
 OR (objectname = 'ISNOTIFYGROUPUSER' AND attributename = 'NOTIFYGROUPID')
 OR (objectname = 'ISNOTIFYGROUPUSER' AND attributename = 'NOTIFYUSERID')
 OR (objectname = 'ISNOTIFYOBJECT' AND attributename = 'OBJECTNAME')
 OR (objectname = 'ISNOTIFYOPTION' AND attributename = 'NOTIFYID')
 OR (objectname = 'ISNOTIFYOPTION' AND attributename = 'OPTIONNAME')
 OR (objectname = 'ISNOTIFYSEQ' AND attributename = 'SEQUENCE')
 OR (objectname = 'ISNOTIFYVERSION' AND attributename = 'VERSION')
);
```

Index Violations

In versions of Informer before 5.5.0, it was possible to have multiple identical Catalog registrations (for the same device registering for the same revision of the same Catalog), due to timing issues. This resulted in lookup errors, causing affected devices to get inaccurate or missing prompts for Catalog synchronization. A unique index now prevents this possibility, and is installed with Informer. In order for installation to succeed, the current data must not violate this new unique index.

This SQL will have no effect if nothing needs to be corrected. There is no need to run this again during subsequent upgrades.

Run the following SQL while the system is down, before upgrading:

```
-- Note: The below statement is consciously overcomplicated in order to be compatible
with all 3 possible DBMSes Maximo supports
DELETE FROM iscatalogdevice WHERE deviceid IN (
    SELECT iscatalogdevice.deviceid
    FROM iscatalogdevice
    JOIN (
      SELECT deviceid, (COUNT(*) OVER (PARTITION BY DEVICEKEY, CATALOGID, REVISION,
LANGCODE)) AS numcopies
    FROM iscatalogdevice
    ) subq ON iscatalogdevice.deviceid = subq.deviceid AND numcopies > 1
);
```

Appendix: Additional Pre-Installation Checks when Upgrading from Informer 5.2

Clean up Existing Informer Profiles

If upgrading a system which has an existing Informer installation of 5.2 or earlier, please perform the following steps:

Deleting Registered Devices

Informer 5.2+ introduces a new method to track and store catalog data. Therefore, it is necessary to delete any existing device registrations before allowing devices to connect to the newly installed version of Informer. Please note that deleting device records will result in the client device **deleting all stored data, including any pending commands(user actions)** in the transaction queue. For this reason it is important to ensure that all pending commands are sent before this action is performed. To delete device records, please follow these steps:

1. Using the Informer Developer or Informer Administration applications, navigate to the Devices tab.
2. Select the check box next to the device you wish to delete. It is also possible to select more than one device or all devices on the page.
3. Click the Delete Selected button.
4. Save

NOTE	Deleting the device record also deletes any associated push device registrations so there is no need to take any further action.

When the device attempts to connect (following device deletion) it will recognize that a reset has taken place and will wipe all stored data. Hence, the provisioning of catalogs and notifications will begin.

Deactivating Informer Profiles

To deactivate a profile, the system should be running.

1. Go To Administration > Informer Administration
2. Administration > Informer Developer can also be used
3. In the List view, open any record whose "active" checkbox is marked
4. Select the "Deactivate Profile" action.
5. Repeat for each active profile

More information on profile activation and deactivation can be found in the "Informer Administration" and "Informer Developer" application guides.

Once the profiles are deactivated, shut down the application server.

Clearing Informer Processing Queues

Before clearing the Informer queues, Informer profiles should be deactivated (above), and the application servers should be shut down.

Only then, execute the following statements to immediately and completely clear Informer processing queues.

```
TRUNCATE TABLE isnotifyqueuemonitor;
TRUNCATE TABLE isnotifyrefreshqueue;
TRUNCATE TABLE iscatalogqueuemonitor;
TRUNCATE TABLE iscatalogrefreshqueue;
TRUNCATE TABLE ispushqueuemonitor;
TRUNCATE TABLE ispushqueue;
```

The `ISPUSHQUEUE`, `ISPUSHQUEUEMONITOR`, `ISCATALOGREFRESHQUEUE` and/or `ISCATALOGQUEUEMONITOR` tables may not exist, depending on the version of Informer you are upgrading from. If an error occurs while attempting to truncate, please confirm that these tables exist. If they do not, then there is no issue.

Index Violations

In versions of Informer before 5.5.0, it was possible to have multiple identical Catalog registrations (for the same device registering for the same revision of the same Catalog), due to timing issues. This resulted in lookup errors, causing affected devices to get inaccurate or missing prompts for Catalog synchronization. A unique index now prevents this possibility, and is installed with Informer. In order for installation to succeed, the current data must not violate this new unique index.

This SQL will have no effect if nothing needs to be corrected. There is no need to run this again during subsequent upgrades.

Run the following SQL while the system is down, before upgrading:

```
-- Note: The below statement is consciously overcomplicated in order to be compatible
with all 3 possible DBMSes Maximo supports
DELETE FROM iscatalogdevice WHERE deviceid IN (
    SELECT iscatalogdevice.deviceid
    FROM iscatalogdevice
    JOIN (
      SELECT deviceid, (COUNT(*) OVER (PARTITION BY DEVICEKEY, CATALOGID, REVISION,
LANGCODE)) AS numcopies
    FROM iscatalogdevice
    ) subq ON iscatalogdevice.deviceid = subq.deviceid AND numcopies > 1
);
```

Index Violations

In versions of Informer before 5.4.0, it was possible to have multiple functionally identical records in `ISCATALOGDATA`, which could result in lookup errors. A unique index now prevents this possibility, and is installed with Informer. In order for installation to succeed, the current data must not violate this new unique index.

This SQL will have no effect if nothing needs to be corrected. There is no need to run this again during subsequent upgrades.

Run the following SQL while the system is down, before upgrading:

```
-- Delete violators of ISCATALOGDATA_NDX6
DELETE
FROM iscatalogdata
WHERE catalogdataid IN
  (SELECT catalogdataid
  FROM iscatalogdata cd
  JOIN
    (SELECT catalogid, recordid, MAX(catalogdataid) newestDup
    FROM iscatalogdata
    GROUP BY catalogid, recordid
    HAVING COUNT(*) > 1
    ) cddup
  ON (cd.catalogid        = cddup.catalogid
  AND cd.recordid         = cddup.recordid)
  WHERE cd.catalogdataid <> cddup.newestdup
  );
```

Required Fields

In versions of Informer before 5.3.0, some fields were functionally required, but not enforced in the database configuration. Running these two queries corrects the record, preventing errors during upgrade without sacrificing other forms of validation.

```
UPDATE maxattribute SET required = 1 WHERE required = 0 AND (
 (objectname = 'ISNOTIFYATTRIBUTE' AND attributename = 'ATTRIBUTENAME')
 OR (objectname = 'ISNOTIFYGROUPOPTION' AND attributename = 'NOTIFYGROUPID')
 OR (objectname = 'ISNOTIFYGROUPOPTION' AND attributename = 'NOTIFYOPTIONID')
 OR (objectname = 'ISNOTIFYGROUPUSER' AND attributename = 'NOTIFYGROUPID')
 OR (objectname = 'ISNOTIFYGROUPUSER' AND attributename = 'NOTIFYUSERID')
 OR (objectname = 'ISNOTIFYOBJECT' AND attributename = 'OBJECTNAME')
 OR (objectname = 'ISNOTIFYOPTION' AND attributename = 'NOTIFYID')
 OR (objectname = 'ISNOTIFYOPTION' AND attributename = 'OPTIONNAME')
 OR (objectname = 'ISNOTIFYSEQ' AND attributename = 'SEQUENCE')
 OR (objectname = 'ISNOTIFYVERSION' AND attributename = 'VERSION')
);
UPDATE maxattributecfg SET required = 1 WHERE required = 0 AND (
 (objectname = 'ISNOTIFYATTRIBUTE' AND attributename = 'ATTRIBUTENAME')
 OR (objectname = 'ISNOTIFYGROUPOPTION' AND attributename = 'NOTIFYGROUPID')
 OR (objectname = 'ISNOTIFYGROUPOPTION' AND attributename = 'NOTIFYOPTIONID')
 OR (objectname = 'ISNOTIFYGROUPUSER' AND attributename = 'NOTIFYGROUPID')
 OR (objectname = 'ISNOTIFYGROUPUSER' AND attributename = 'NOTIFYUSERID')
 OR (objectname = 'ISNOTIFYOBJECT' AND attributename = 'OBJECTNAME')
 OR (objectname = 'ISNOTIFYOPTION' AND attributename = 'NOTIFYID')
 OR (objectname = 'ISNOTIFYOPTION' AND attributename = 'OPTIONNAME')
 OR (objectname = 'ISNOTIFYSEQ' AND attributename = 'SEQUENCE')
 OR (objectname = 'ISNOTIFYVERSION' AND attributename = 'VERSION')
);
```

Index Violations

In versions of Informer before 5.5.0, it was possible to have multiple identical Catalog registrations (for the same device registering for the same revision of the same Catalog), due to timing issues. This resulted in lookup errors, causing affected devices to get inaccurate or missing prompts for Catalog synchronization. A unique index now prevents this possibility, and is installed with Informer. In order for installation to succeed, the current data must not violate this new unique index.

This SQL will have no effect if nothing needs to be corrected. There is no need to run this again during subsequent upgrades.

Run the following SQL while the system is down, before upgrading:

```
-- Note: The below statement is consciously overcomplicated in order to be compatible
with all 3 possible DBMSes Maximo supports
DELETE FROM iscatalogdevice WHERE deviceid IN (
    SELECT iscatalogdevice.deviceid
    FROM iscatalogdevice
    JOIN (
      SELECT deviceid, (COUNT(*) OVER (PARTITION BY DEVICEKEY, CATALOGID, REVISION,
LANGCODE)) AS numcopies
    FROM iscatalogdevice
    ) subq ON iscatalogdevice.deviceid = subq.deviceid AND numcopies > 1
);
```

Appendix: Additional Pre-Installation Checks when Upgrading from Informer 5.3

Index Violations

In versions of Informer before 5.4.0, it was possible to have multiple functionally identical records in `ISCATALOGDATA`, which could result in lookup errors. A unique index now prevents this possibility, and is installed with Informer. In order for installation to succeed, the current data must not violate this new unique index.

This SQL will have no effect if nothing needs to be corrected. There is no need to run this again during subsequent upgrades.

Run the following SQL while the system is down, before upgrading:

```
-- Delete violators of ISCATALOGDATA_NDX6
DELETE
FROM iscatalogdata
WHERE catalogdataid IN
  (SELECT catalogdataid
  FROM iscatalogdata cd
  JOIN
    (SELECT catalogid, recordid, MAX(catalogdataid) newestDup
    FROM iscatalogdata
    GROUP BY catalogid, recordid
    HAVING COUNT(*) > 1
    ) cddup
  ON (cd.catalogid        = cddup.catalogid
  AND cd.recordid         = cddup.recordid)
  WHERE cd.catalogdataid <> cddup.newestdup
  );
```

Required Fields

In versions of Informer before 5.3.0, some fields were functionally required, but not enforced in the database configuration. Running these two queries corrects the record, preventing errors during upgrade without sacrificing other forms of validation.

```
UPDATE maxattribute SET required = 1 WHERE required = 0 AND (
 (objectname = 'ISNOTIFYATTRIBUTE' AND attributename = 'ATTRIBUTENAME')
 OR (objectname = 'ISNOTIFYGROUPOPTION' AND attributename = 'NOTIFYGROUPID')
 OR (objectname = 'ISNOTIFYGROUPOPTION' AND attributename = 'NOTIFYOPTIONID')
 OR (objectname = 'ISNOTIFYGROUPUSER' AND attributename = 'NOTIFYGROUPID')
 OR (objectname = 'ISNOTIFYGROUPUSER' AND attributename = 'NOTIFYUSERID')
 OR (objectname = 'ISNOTIFYOBJECT' AND attributename = 'OBJECTNAME')
 OR (objectname = 'ISNOTIFYOPTION' AND attributename = 'NOTIFYID')
 OR (objectname = 'ISNOTIFYOPTION' AND attributename = 'OPTIONNAME')
 OR (objectname = 'ISNOTIFYSEQ' AND attributename = 'SEQUENCE')
 OR (objectname = 'ISNOTIFYVERSION' AND attributename = 'VERSION')
);
UPDATE maxattributecfg SET required = 1 WHERE required = 0 AND (
 (objectname = 'ISNOTIFYATTRIBUTE' AND attributename = 'ATTRIBUTENAME')
 OR (objectname = 'ISNOTIFYGROUPOPTION' AND attributename = 'NOTIFYGROUPID')
 OR (objectname = 'ISNOTIFYGROUPOPTION' AND attributename = 'NOTIFYOPTIONID')
 OR (objectname = 'ISNOTIFYGROUPUSER' AND attributename = 'NOTIFYGROUPID')
 OR (objectname = 'ISNOTIFYGROUPUSER' AND attributename = 'NOTIFYUSERID')
 OR (objectname = 'ISNOTIFYOBJECT' AND attributename = 'OBJECTNAME')
 OR (objectname = 'ISNOTIFYOPTION' AND attributename = 'NOTIFYID')
 OR (objectname = 'ISNOTIFYOPTION' AND attributename = 'OPTIONNAME')
 OR (objectname = 'ISNOTIFYSEQ' AND attributename = 'SEQUENCE')
 OR (objectname = 'ISNOTIFYVERSION' AND attributename = 'VERSION')
);
```

Index Violations

In versions of Informer before 5.5.0, it was possible to have multiple identical Catalog registrations (for the same device registering for the same revision of the same Catalog), due to timing issues. This resulted in lookup errors, causing affected devices to get inaccurate or missing prompts for Catalog synchronization. A unique index now prevents this possibility, and is installed with Informer. In order for installation to succeed, the current data must not violate this new unique index.

This SQL will have no effect if nothing needs to be corrected. There is no need to run this again during subsequent upgrades.

Run the following SQL while the system is down, before upgrading:

```
-- Note: The below statement is consciously overcomplicated in order to be compatible
with all 3 possible DBMSes Maximo supports
DELETE FROM iscatalogdevice WHERE deviceid IN (
    SELECT iscatalogdevice.deviceid
    FROM iscatalogdevice
    JOIN (
      SELECT deviceid, (COUNT(*) OVER (PARTITION BY DEVICEKEY, CATALOGID, REVISION,
LANGCODE)) AS numcopies
    FROM iscatalogdevice
    ) subq ON iscatalogdevice.deviceid = subq.deviceid AND numcopies > 1
);
```

Appendix: Additional Pre-Installation Checks when Upgrading from Informer 5.5

Index Violations

In versions of Informer before 5.5.0, it was possible to have multiple identical Catalog registrations (for the same device registering for the same revision of the same Catalog), due to timing issues. This resulted in lookup errors, causing affected devices to get inaccurate or missing prompts for Catalog synchronization. A unique index now prevents this possibility, and is installed with Informer. In order for installation to succeed, the current data must not violate this new unique index.

This SQL will have no effect if nothing needs to be corrected. There is no need to run this again during subsequent upgrades.

Run the following SQL while the system is down, before upgrading:

```
-- Note: The below statement is consciously overcomplicated in order to be compatible
with all 3 possible DBMSes Maximo supports
DELETE FROM iscatalogdevice WHERE deviceid IN (
    SELECT iscatalogdevice.deviceid
    FROM iscatalogdevice
    JOIN (
      SELECT deviceid, (COUNT(*) OVER (PARTITION BY DEVICEKEY, CATALOGID, REVISION,
LANGCODE)) AS numcopies
    FROM iscatalogdevice
    ) subq ON iscatalogdevice.deviceid = subq.deviceid AND numcopies > 1
);
```

Appendix: Pre-6 Queue Processors

Version 6 takes a new approach to queue processing which improves performance, reduces database resources, and avoids database deadlocks. The previous queue processors are still available and can still be used if needed. This appendix documents how to use and configure the old queue processors.

There are three primary types of queue processors: "Notification","Catalog", and "Push". Notification processors have three sub-types:"User Expansion", "User Evaluator", and "Record". Catalog processors have two sub-types: "Aggregate", and "Record". There is currently only one type of Push processor.

The number of each queue processor thread can be configured with these System/Instance Properties:

Property Name	Description	Value
informer.queue.pool.notification	Specifies the level of concurrency for Notification queue processing. Informer has three type processors for Notification processing. The first number represents the number of"User Expansion" processors, the second represents the number of"User Evaluator" processors, and the third represents the number of"Record" processors. A value of 0 in any position disables that type of processor on the JVM. A setting of "0" (no comma separators) disables all types of Notification queue processors on the JVM.	1,1,1
informer.queue.pool.catalog	Specifies the level of concurrency for Catalog queue processing. Informer has two types of processors for Catalog processing. The first number represents the number of "Aggregate" processors, and the second one represents the number of "Record" processors. A value of 0 in any position disables that type of processor on the JVM. A setting of "0" (no commas separators) disables all types of Catalog queue processors on the JVM.	1,1
informer.queue.pool.push	Specifies the number of "Push" queue processing threads to run on a specific JVM node. If this is set to "0" then no push queue threads will run. NOTE: Any given node can have a maximum of "1" push queue processor.	1

The values above specify that one of each of the six types of processors will run on any given JVM. This is the minimal supported resourcing, but ensures that single-JVM systems perform all required functions, and each added JVM will automatically spin up its own processors to share the workload. In practice, however, deployments of scale will often want to increase the level of concurrency (and therefore, the system resources) granted to the Informer processors. Not all JVMs in a Maximo environment are equivalent, however, so Maximo’s instance-specific property support can be used to create uneven distribution of work.

Notification Pool Configuration

To use the old notification queue processor, set the Maximo property `informer.notification.queue.QueueProcessor` to `com.interlocsolutions.maximo.notify.queue.notification.BulkByThreadNotificationJobQueueProcessor`.

The `informer.queue.pool.notification` property is specified as a comma-separated string of the form `"x,y,z"`, where `x` represents the number of "User Expansion" processors, `y` represents the number of "User Evaluator" processors, and `z` represents the number of "Record" processors.

The "User Expansion" pool is important, but more than 1 is not necessary. When scaling up, the most effective balance of "User Evaluation" vs "Record" processors depends on your Profile, users, and environment. Generally speaking, if you have many users with few records on average, then increasing "User Evaluation" processors will help more, and if you have fewer users with more records on average, then increasing "Record" processors will help more.

Catalog Pool Configuration

To use the old catalog queue processor, set the Maximo property `informer.catalog.queue.QueueProcessor` to `com.interlocsolutions.maximo.notify.queue.db.BulkByThreadCatalogJobQueueProcessor`.

The `informer.queue.pool.catalog` property is specified as a comma-separated string of the form `"x,y"`, where `x` represents the number of "Aggregate" processors, and `y` represents the number of "Record" processors.

"Record" processors should always be significantly more numerous than "Aggregate" processors. When scaling up the processor pool, a ratio of about 1-to-10 is a reasonable place to start.

Push Pool Configuration

To use the old push queue processor, set the Maximo property `informer.push.queue.QueueProcessor` to `com.interlocsolutions.maximo.notify.push.BulkByThreadPushJobQueueProcessor`.

The `informer.queue.pool.push` property is specified as a single numeric value. If using a Push configuration, set this value to 1 on at least one JVM. If you are certain you will not use Push, you can disable this globally by setting a value of 0.

Example 6. Dedicated Single JVM

Scenario: A clustered system which has a single dedicated Informer JVM, called INF1. Push configurations are used by one or more Profiles in the system.

Property	Instance-Specific?	Value
informer.queue.pool.notification		0
informer.queue.pool.catalog		0
informer.queue.pool.push		0
informer.queue.pool.notification	INF1	1,5,10
informer.queue.pool.catalog	INF1	3,30
informer.queue.pool.push	INF1	1

Example 7. Dedicated Cluster

Scenario: A clustered system which has two dedicated Informer JVMs, called INF1 and INF2. Push configurations are used by one or more Profiles in the system.

Property	Instance-Specific?	Value
informer.queue.pool.notification		0
informer.queue.pool.catalog		0
informer.queue.pool.push		0
informer.queue.pool.notification	INF1	1,5,10
informer.queue.pool.catalog	INF1	3,30
informer.queue.pool.push	INF1	1
informer.queue.pool.notification	INF2	0,5,10
informer.queue.pool.catalog	INF2	3,30
informer.queue.pool.push	INF2	0

Example 8. Reuse of Non-UI JVMs

Scenario: A clustered system which has no dedicated Informer JVMs. The UI JVMs should be protected, but other JVMs in the system have a little bandwidth to spare. No Profiles in the system use Push.

Property	Instance-Specific?	Value
informer.queue.pool.notification		0
informer.queue.pool.catalog		0
informer.queue.pool.push		0
informer.queue.pool.notification	MIF1	0,1,5
informer.queue.pool.catalog	MIF1	1,10
informer.queue.pool.notification	MIF2	0,1,5
informer.queue.pool.catalog	MIF2	1,10
informer.queue.pool.notification	CRON1	1,1,10
informer.queue.pool.catalog	CRON1	1,20

Example 9. Equal Distribution

Scenario: A clustered system which has no dedicated Informer JVMs, but just wants to share a light load equally across all JVMs and see how that goes. Push configurations are used by one or more Profiles in the system.

NOTE This is the same as default configuration, without applying any changes.

Property	Instance-Specific?	Value
informer.queue.pool.notification		1,1,1
informer.queue.pool.catalog		1,1
informer.queue.pool.push		1

Example 10. Minimal Maximo System

Scenario: A single-JVM system which needs to accelerate Informer processing. Push configurations are used by one or more Profiles in the system.

Property	Instance-Specific?	Value
informer.queue.pool.notification		1,10,10
informer.queue.pool.catalog		2,20
informer.queue.pool.push		1