**Installer**

**for**

**Ignition-based Applications**

**Document Version: 1.0**

**Dec 23, 2015**

Table of Contents

Table of Contents

1. Change History 2

2. Introduction 3

2.1 Design Goals 3

3. Use Cases 4

3.1 End-user Installation 4

3.2 Installer Creation 5

3.3 Installer Update 5

3.4 Master Builder Update 6

4. Typical Workflow 7

4.1 Master Builder 7

4.2 Application Resources 7

4.3 Application-Specific Builder 8

4.4 Installer Module/Project 8

4.5 Application Installer 9

5. Install Panels 10

5.1 Overview 10

# Change History

| Version | Date | Author | Changes |
| --- | --- | --- | --- |
| 1.0 | Dec 23, 2015 | C. Coughlin (ILS) | Initial concept, use cases |

# Introduction

Ignition™ is an execution and development platform from Inductive Automation designed to support a wide-range of industrial applications. While a basic Ignition-based application may require only a project file, a more comprehensive application may require a number of different components for a complete installation. These components that may involve a variety of file types and installation procedures. The variety may be confusing to the end user. Absence of one or more of these components may yield an incomplete or inconsistent installation and result in subtle (or not so subtle) errors.

Beyond the initial clean installation, updates of previously installed versions may be just as complex and error-prone.

This document describes an ILS-Automation product that handles application releases in the familiar paradigm of an installer. Using the ILS Application Installer, the end user is presented with a wizard-style sequence of screens that handle installation of the various components that make up the target application. These components may be, among other things: full or partial projects, global projects, UDT definitions, icons, internal and/or external python packages, SQL update scripts, and Java-based modules.

The release bundle or installer for a particular delivery is packaged into a single file, an Ignition module file. Embedded within the module are all resources required for the target application plus an Ignition project to install it. This project is available only when the install module is loaded into the Gateway. The project supports an end-user wizard-style interface to accomplish the installation.

## Design Goals

There are 3 overarching goals that determine the design of this product:

1. Directed installation. For the end-user, the installation process shall be completely scripted. Even if automated installation actions are not possible (but clearly preferred), the user should be guided through each step of the process.
2. Single file. The installation package shall be a single file. Everything required for a complete installation of the target project shall be contained in that bundle.
3. No outside dependencies. The installer shall operate within a plain Ignition installation. There shall be no dependencies on tags, databases or other configuration of the Ignition gateway.

# Use Cases

This section describes the major use cases supported by the Application Installer product. These cases involve three actors:

* END-USER – This is the customer of the final product. This user accepts the installation package and installs the application that it contains on an Ignition gateway.
* INSTALLER-DEVELOPER – This user prepares the installation package. He or she is typically a member of the team that developed the application to be installed.
* ILS – ILS Automation is the developer of the master builder. The master builder facilitates construction of an installer for a particular application.

Two main resources are considered:

* application-installation-package – This package is an Ignition module that contains all artifacts necessary for installation of the target application. It contains the Ignition project that is the installer.
* master-builder-package – This application bundle is also an Ignition module. It is a general-purpose starting point for building installers. It contains a project that guides installer-developers through the process of generating an installer for some target product.

## End-user Installation

In this use case, the ultimate user of an application installs it at a production site.

Setup: Ignition 7.8 is installed.

1. The END-USER receives an application-installation-package from an INSTALLER-DEVELOPER.
2. The END-USER loads this package into an Ignition gateway. As soon as the package (module) runs, an application-installer project appears.
3. The END-USER follows the steps in the installer project to install or update the application that is included in it.
4. Once these steps are complete, the END-USER unloads the application-installation-package from the gateway.

Result: The target application is installed at the end-user’s site. The installer package and the project that it contains are completely removed from the Gateway.

## Installer Creation

In this use case, an application developer installs the master builder and creates an application-specific installer.

Setup: Ignition 7.8 is installed.

1. The INSTALLER-DEVELOPER receives the master-builder-package from ILS.
2. The INSTALLER-DEVELOPER loads this package into an Ignition gateway. As soon as the package (module) runs, a master builder project appears.
3. The INSTALLER-DEVELOPER follows the steps in the master builder project required to create an installer for the target application. Some custom development of Vision screens is required to personalize the resulting installer.
4. As a final step, the INSTALLER-DEVELOPER creates an application-installation-package that is specific to the target product. This is an Ignition module and must be signed at the Inductive Automation website before deliver to the END-USER.
5. Once these steps are complete, the INSTALLER-DEVELOPER unloads the master-builder-package from the gateway.

Result: The INSTALLER-DEVELOPER has created a custom application-installation-package specific to the target application. All evidence of the master-builder-package has been removed from the Gateway.

## Installer Update

In this scenario, the installer developer has already created an installation package. Now the product has changed and it is time to build an installer for the update.

Setup: Ignition 7.8 is installed. A completed application-installation-package has been loaded into the gateway. This installation package corresponds to a now-obsolete version of the product.

1. The INSTALLER-DEVELOPER loads a master-builder-package from ILS.
2. The INSTALLER-DEVELOPER loads master-builder package into the same gateway holding the application installer project. Once the load is complete, a master builder project appears.
3. As before, the INSTALLER-DEVELOPER follows the steps outlined in the master builder project. The first of these steps inquires whether this is to be a new installation or update. Unlike previously, INSTALLER-DEVELOPER specifies the project that is to be updated. The builder application uses this project for its initial settings. The install screens from prior application installer become the default screens in the builder.
4. The INSTALLER-DEVELOPER continues following the steps in the master builder project.
5. As a final step, the INSTALLER-DEVELOPER creates an application-installation-package that is specific to the new version of the target product.
6. Once these steps are complete, the INSTALLER-DEVELOPER unloads the master-builder-package from the gateway.

Result: The INSTALLER-DEVELOPER has created a custom application-installation-package that updates an existing installation of the target application. All evidence of the master-builder-package has been completely removed from the gateway.

## Master Builder Update

In this scenario, ILS Automation releases an update of the master builder. The installer developer wishes to use this version of the master builder to create an installer for a product update.

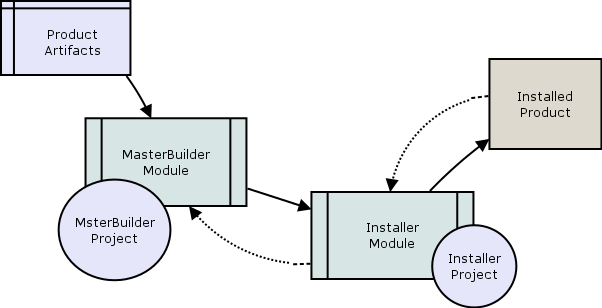
Setup: Ignition 7.8 is installed. A completed, but obsolete, application-installation-package has been loaded into the gateway. This installation package corresponds to a now-obsolete version of the product.

1. NOTE: This use case proceeds exactly as the previous. There are no differences in the use case steps due to the fact that the master installer may itself have been updated.

Result: The INSTALLER-DEVELOPER has created a custom application-installation-package specific to the target application. The package incorporates any new features available in the new master builder from ILS. All evidence of the master-builder-package has been completely removed from the gateway.

# Typical Workflow

This section describes the general order of operations involved with preparation and use of an installation bundle.



Workflow

## Master Builder

The master builder is an Ignition module supplied by ILS-Automation. It contains construction templates for each type of resource handled by the installer. In addition it supplies the general-purpose scripts needed to create the final installer project.

The master builder module contains a "Master Builder" project. This project contains standard Ignition screens that guide the installer-developer through the construction steps using a familiar “wizard” paradigm. The end result of completing the steps in the master builder is creation of an separate Ignition module which holds everything necessary to install the application – plus an installer project designed for the end user to accomplish this.

## Application Resources

By application resources we mean the various artifacts that make up the product to be delivered. These artifacts must exist in the file system of the development machine. Potential resources are:

* icons (.png files in a jar)
* UDT definitions (.xml export from Ignition)
* modules (signed .modl file)
* database create or alter scripts
* partial projects (Vision resources to be merged)

Note that the master builder module … .

## Application-Specific Builder

The engineer or developer preparing an installer for a specific product and version copies the master builder project and configures an install sequence.

The install builder is designed to produce an installer that follows the very common paradigm of a series of overlaid panels, each guiding an installation steps In the builder, the panels refer to file-system locations of the various resources, but do not load them until the ultimate installer-project is created.

The amount of work required to create a project for an update is likely to be minimal if starting from the previous builder project. The panels are likely to remain unchanged, and, if the files paths point to locations in a build system (i.e. source control repository), file path configurations are likely to be unchanged as well. Any upgrades in the master builder may simply be merged.

## Installer Module/Project

Once the product builder project is complete and configured (including a loonk to the starter jar), the engineer/developer selects a location in the file system and launches the build. On completion of the build, an Ignition module will have been created. It will contain, embedded, all the resources that comprise the product. Most importantly it will contain the install-wizard for use by the end user.

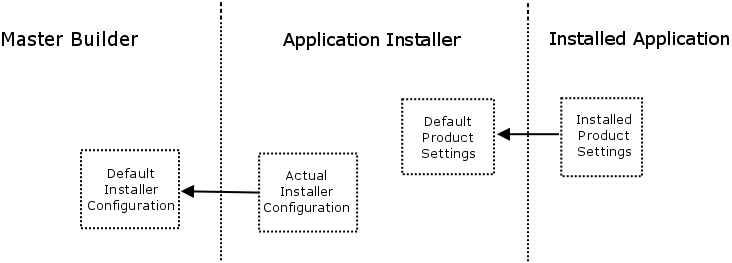
The install wizard is an Ignition project that appears when the Gateway is re-started with the Application Installer module in place. The wizard project disappears when the Application Installer module is removed.

## Application Installer

The end-user loads the release-project into the target Gateway, starts the Designer and follows the installer instructions. The ILS “Application Installation” module must have been previously installed. On completion of the install, the user’s system is configured with a new project containing the subject product and version. Any existing user-data in the previous project is preserved. The newly-created-project is placed in a disabled state to prevent it from interfering with the existing system.

# Configuration Sharing

This section describes the flow and ownership of the various elements of configuration. In general the configuration information is stored in a project resource. If a downstream project pre-exists, then its values overwrite the upstream defaults. In this way, existing settings are preserved as the new installer or application is created.



The configurations are stored as a project resource of type “installer”. The format is JSON.

# Install Panels

Each panel described below is a template in the Master Builder project. Each panel handles a specific type of application resource.

## Overview

When complete, the panels are overlain to form the familiar wizard layout familiar to many users. A sample is shown below:

Wizard