Sept 2011

Patch Assistant
3.1.0 GUI Guide

Patch Assistant v.3.1.0

Patch Creation Tool Graphical User Interface Guide

Table of Contents

Overview

Patch Assistant is a wizard-based patch creation tool that is used to prepare and deliver patches for online games. The easy-to-use tool aids publishers by streamlining the patch creation and deployment process.

Patch Assistant acts as a central hub for a variety of patching projects; handling versioning, deployment and activation from a simple interface. It creates all the files necessary to deliver content via the Distributed Delivery Network™ (DDN), enabling high-performance delivery and comprehensive reporting analytics.

Prerequisites

Currently Patch Assistant can be run in a Windows operating system. The Patch Assistant installer will check for the following requirements:

- Windows Installer 3.1
- .NET Framework 3.5 SP1

Glossary

Content: a file or set of files that the DDN can deliver.

Distributed Delivery Network (DDN): - Solid State Networks' advanced delivery system. It comprises a reliable source (CDN) for content, the Solid Tracker and optional Solid Seed servers.

Metafile: contains all of the information the DDN needs to deliver a single piece of content. When stored to the filesystem, metafiles use the .solidpkg extension.

Product: a software product that includes all the components necessary to use and maintain it (i.e. the game filesystem, pre-requisites, and a patcher).

Payload: a component of the software product that contains deployment information. The payload directory contains a deployment settings file and content for every release.

Release: a set of files that constitute a deployable version of a software product or component of a product.

Patch Manifest: an encrypted patch versioning file containing release information the patcher uses to check for updates, as well as download configuration.

Reliable Source: a generic term for any web server or content delivery network used to deliver content via HTTP.

Login Credentials

Log in with your email and the password provided by Solid State Networks. For assistance please contact support@solidstatenetworks.com.

Product Setup

The product setup screen allows you to create new products and attach existing products. To view the setup screen select the **Patch Assistant** node located atop the tree on the left side of the application.

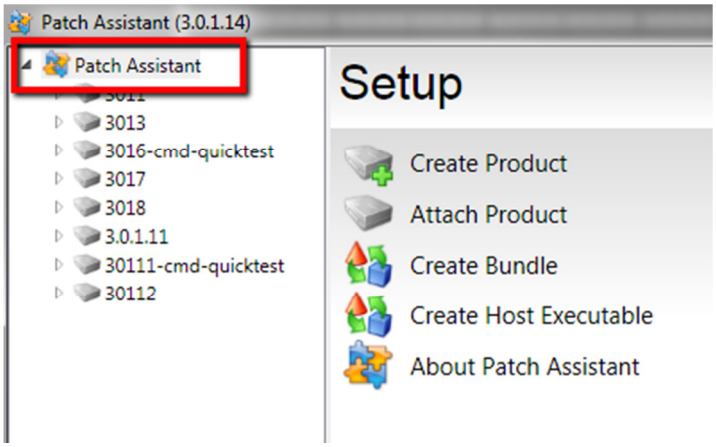

Figure 1 – Setup Screen

Create a New Product

1. From the *Setup* screen, select **Create Product.**
2. Fill out the fields on the *Create Product* screen:
    a. **Name:** product name, in most cases the name of the game or software.
    b. **Path:** local path where patch data and filesystems will be created.
3. Click **Next**. The product node will be added to the tree on left. Click **Finish**.

Attach an Existing Product

1. From the *Setup* Screen (Figure 1), select **Attach Product**.
2. Enter the local path to the product directory and click **Finish**. The product node is added to the tree.

Create Bundle

This function compiles a bundle – a file system contained in an encrypted, signed zip file used by the host (such as the patcher, downloader or launcher bundles). Use this to deliver JavaScript securely from remote sites and extend the functionality of the host.

1. From the Setup screen, select **Create Bundle.**
2. Fill out the fields on the *Create Bundle* screen:
    a. **Bundle path**: local path to the bundle file system
    b. **Output file**: local path and filename (complete with extension) where bundle will be created
3. Click **Finish**.

Create Host Executable

This function compiles the skin file system into an encrypted, signed application.

1. From the Setup screen, select **Create Host Executable.**
2. Fill out the fields on the *Create Host Executable* screen:
    a. **Resource Type:** base executable to use when building. Internet Explorer (stripped) will make the resulting .exe smaller by stripping debugging information from the host.
    b. **Bundle path**: local path to the bundle (skin) file system
    c. **Product Name**: Product name in executable
    d. **Company Name**: Company name in executable
    e. **Copyright**: Copyright information in executable
    f. **Output file:** local path and filename (complete with extension) where executable will be created
    g. **UPX Compress:** When checked, the executable is compressed using Ultimate Packer for eXecutables (UPX) compression
3. Click **Finish.**

Payload Setup

The *Product Management* screen allows you to create new payloads and attach existing payloads to products. To view the *Product Management* screen, select a product node from the tree on the left side of the application.

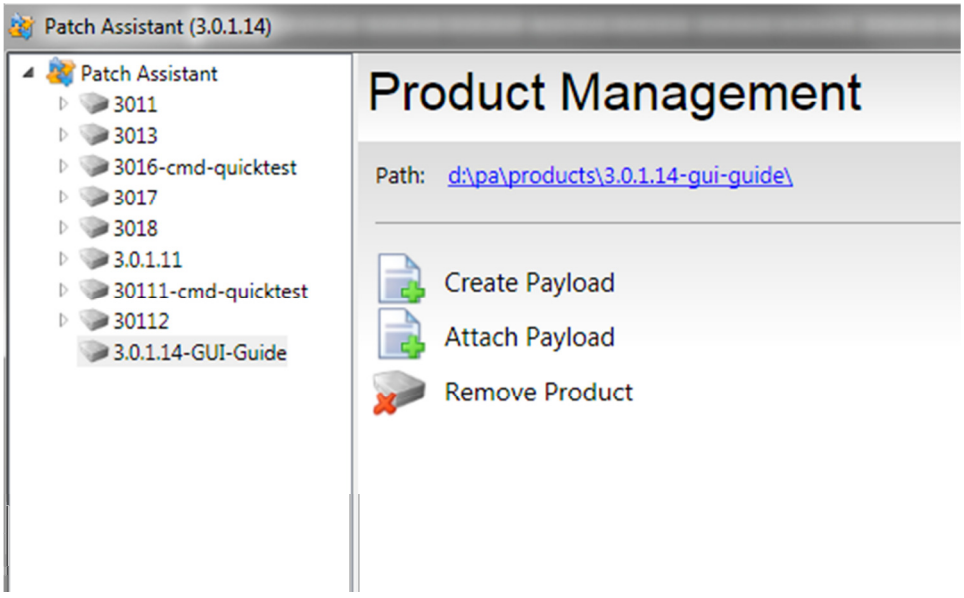

Figure 2 – Product Management

Create a New Payload

1. From the *Product Management* screen, select **Create Payload**.
2. Fill out the fields on the *Create Payload* screen:
    a. **Name:** payload name, in most cases the name of the component (pre-requisite application, launcher/patcher or game filesystem)
    b. **Target Directory:** path to where the launcher will apply the downloaded payload files. The default module is **{ModulePath}**, which denotes the active launcher directory. Other available modules (all encapsulated by curly brackets {}:
        i. ModuleFilename, TempPath, ModuleArguments, ProgramFiles, StartMenu, UserDocuments, UserDesktop, UserAppData, UserAppDataRoaming
    c. **Requires Elevation:** Indicates if the payload requires elevated (Administrator) permissions in order to be applied. For some operating systems with User Account Control, Administrator privileges are required when applying patches within the Program Files directory.
    d. **Dependency:** Full URL to a dependent payload (e.g. http://cdn.company.com/game/directx-pre-req.patchmanifest). Click on the down arrow to add the URL to the list.
3. Click **Finish.** The payload folder will be added to the Product tree.

Attach an Existing Payload

1. Select the Product node you wish to attach a payload to from the tree on the left.
2. From the *Product Management* Screen (Figure 2), select **Attach Payload.**
3. Enter the local path to an existing payload directory (must be a sub-directory of the selected Product) and click **Finish**. The payload folder will be added to the Product tree.

Edit Payload

The **Edit Payload** function allows you to change the target directory, elevation setting and dependencies. You cannot change the Project name. Select the **Edit Payload** option from the *Payload Management* screen (Figure 3 below) to update your payload options.

NOTE: in order for your updated payload options to take effect, you MUST re-create the patch manifest for that payload using the **Create Patch Manifest** function (Building and Injecting releases are not required). Editing payload options without re-creating the patch manifest WILL NOT affect existing payload manifests in test or production environments.

Release Setup

The *Payload Management* screen allows you to create new releases or build and inject all listed releases. To view a payload's management options select the payload folder from the tree below its corresponding Product.

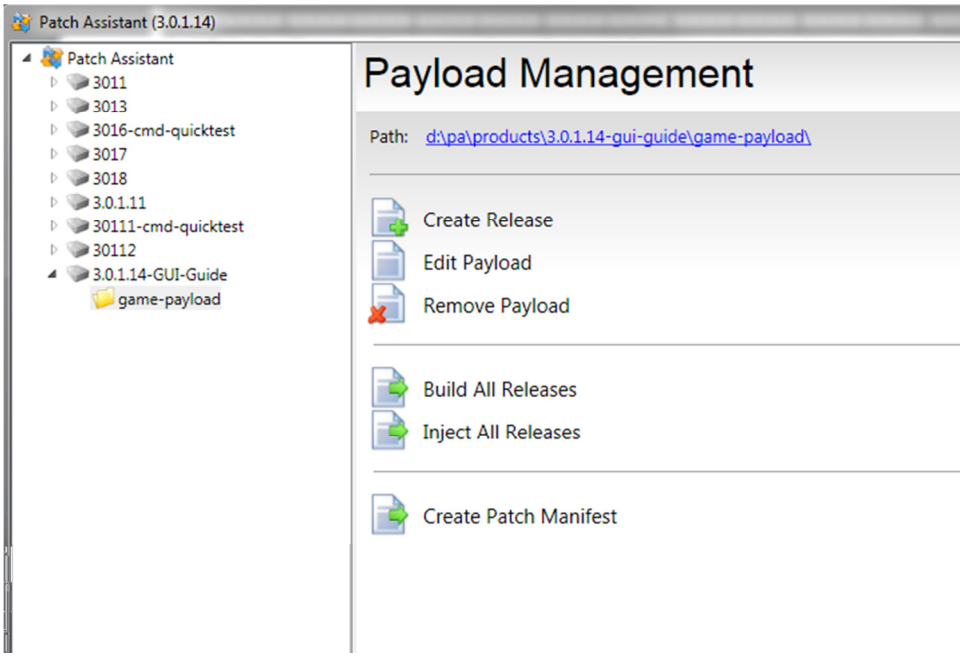

Figure 3 – Payload Management

Create a New Release

1. From the *Payload Management* screen, select **Create Release** and fill out the fields on the *Create Release* screen:

    a. **Name:** unique, friendly name for the release (usually a version number).

    b. **Source:** source directory containing the release content.

    a. **Differential Type:** denotes whether or not Patch Assistant uses the XDelta differential compression tools and algorithm when building the release files.

    b. **Encrypt:** whether or not the patch data is encrypted.

    c. **Disable differential expansion for files:** files listed here are excluded from differential updating. Separate files with a semicolon only, for example: file1.txt;file2.txt;*.ini;

    d. **Update from:** Check the boxes to all the releases that you want Patch Assistant to create an update path from. For example, if you're creating Release 5 and you check the Release 2 box, Patch Assistant will create the 2-to-5 update path.

e. **Rollback to:** Check the boxes to all the releases that you want Patch Assistant to create a rollback path to. For example, if you're creating Release 5 and you check the Release 2 box, Patch Assistant will create the 5to2 rollback path.

NOTE: The update from and rollback to section is where you manage whether your payloads update incrementally, progressively (single-step), or whether certain releases are required in a hybrid incremental/progressive patchline. See our Knowledge Base for more information.

2. Click **Next**. Patch Assistant will copy the source directory into the payload directory within the source\[release number] folder. It will also create an encrypted .version file containing a SHA1 for each file in the release that is used for patch verification and repair.

Release Management

The *Release Management* screen allows you to build, inject, and manage the extra data for your releases. To view the release management options select the release node from the tree.

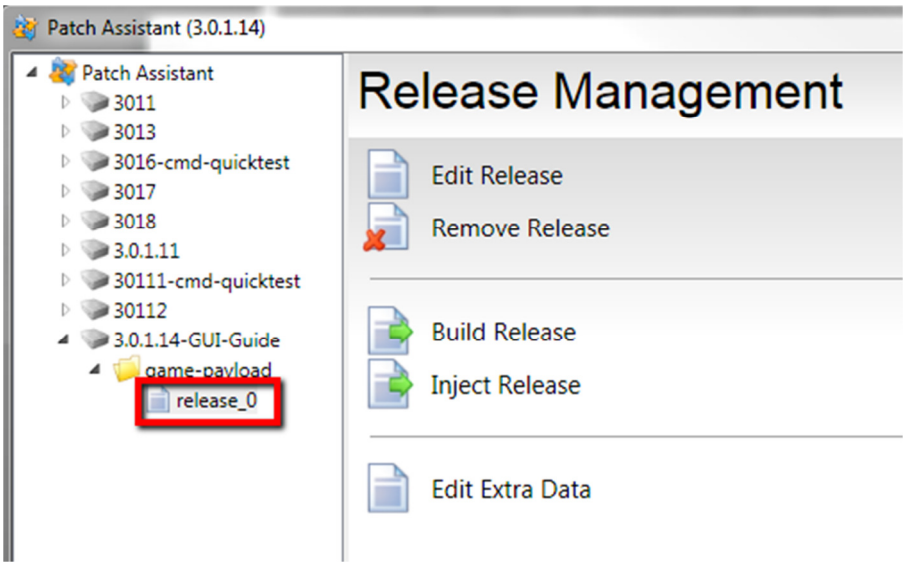

Figure 4 – Release Management

Build Release

This function makes Patch Assistant build the update and rollback paths for that particular release.

1. From the *Release Management* screen, select **Build Release**.
2. Toggle the "Remove old patch files before building" box:
    a. When checked, the existing update paths for that release will be deleted
    b. When unchecked, only new update paths to/from the release will be created. Existing update paths will be skipped.
3. Click **Next**. Patch Assistant will calculate the difference for each update path and compress the data. The results will be placed in the payload\out\ directory. This will be the directory you will want to upload or rsync to your reliable source (CDN/web server).
4. After the release is built, click **Finish**.

Inject Release

This function injects your patch content into our Distributed Delivery Network system, which enables peer-to-peer delivery (optional), performance metrics/analytics and enhanced content security.

1. From the *Release Management* screen, select **Inject Release**.
2. Fill in the fields on the *Inject Release* screen:
    a. **Group:** Typically you will have access to a single injection group. If you have multiple Injection Groups, select the one for this release from the dropdown box.
    b. **Reliable Source:** HTTP delivery URL for your content (patch files), such as http://cdn.yourcompany.com/. **This URL must end with a trailing forward slash "/"**. It can be unique to each payload and can be at any directory level on the delivery server (i.e. http://cdn.company.com/game/beta/1/)
    c. **Download Configuration:** URL to a solid download configuration file. The configuration file includes piece order settings, number of concurrent HTTP connections, and other download settings.
3. Click **Next**. Patch Assistant will inject the release into our ADMIN and Analytics system. It will also output an encrypted metafile (.solidpkg) to the payload\out\ directory. This will be the directory you will want to upload or rsync to your reliable source (CDN/web server).

Edit Extra Data

This function adds custom key/value pairs to an update path.

1. From the Release Management screen, select **Edit Extra Data**.
2. Fill out the fields in the *Edit Extra Data* screen:
    a. **From Release:** release currently existing on end-users' machines
    b. **To Release:** release after patch is applied
    c. **Key:** extra data key
    d. **Value:** extra data value
3. Click on the down arrow to add that Extra Data to the specified update path.
4. Click **Finish**. Patch Assistant will add the Extra Data to the Patch Manifest when it is created for the active payload.

Create Patch Manifest

This function sets the active (latest) release for each project. It also denotes any upcoming (optional to download) releases.

1. From the *Payload Management* screen, select **Create Patch Manifest.**
2. Fill out the fields in the *Create Patch Manifest* screen:
    a. **Name:** unique name for the Patch Manifest (.patchmanifest file)
    b. **Create Maintenance Manifest checkbox:** When checked, the launcher will display a maintenance message to end-users.
    c. **Required Release:** required version to launch the application.
    d. **Upcoming Release:** this release can be optionally downloaded, but it is not applied until it is made the required release. You can specify multiple upcoming releases available for download.
3. Click **Next.** Patch Assistant outputs a .patchmanifest file to the payload\out directory. This will be the directory you will want to upload or rsync to your reliable source (CDN/web server).

TEST ENVIRONMENT NOTE: Patch Manifests can be uploaded to any environment for testing purposes, even if the release content itself has been uploaded to the production environment. See our best practices documentation for more detailed information.

Contact Information

Solid State Networks understands that each game has unique challenges and demands, which is why we offer flexible and customized solutions to suit those specific needs. Our dedicated support team is available to answer questions and assist with implementations when necessary.

For Support Please Visit: http://support.solidstatenetworks.com