Version 1.0.1 Installation & Administration Guide

Supported Platforms

The following operating systems are supported for iDeployIt:

WINDOWS:	UNIX/Linux:
▪ **Windows 7 and higher** ▪ **Windows Server 2012 and higher**	▪ **RedHat Linux** ▪ **CentOS Linux** ▪ **openSUSE Linux** ▪ **Ubuntu Linux** ▪ **SunOS/Solaris** ▪ **HP-UX** ▪ **AIX**

5

Product Overview & Capabilities

iDeployIt is a flexible Software Deployment tool for today's software deployment scenarios. It allows for local distributions and complex remote multi-server environment deliveries. It provides business logic gates to control and configure software before, during and after deployment, using configurable triggers. iDeployIt integrates with all CM tools or build processes providing version control extraction, software building and post-delivery customization as part of a deployment.

iDeployIt provides a straightforward model for the application infrastructure and governance models. Define users who can deploy, determine products to be deployed, where deployments are installed and any configuration scripts to be run for each product. Once defined, use the *intuitive natural language interface* to deploy, no need to know passwords or any details.

Views provide a detailed accounting of all deployments in a centralized web-enabled interface. The ability to compare deployments against other deployments to look for problems, bottlenecks, or deployment details.

Many deployment products charge for the sending server and then additional costs for receivers or remote agents. However, iDeployIt is licensed per sender/shipper, there is no charge for receivers or remote agents.

With iDeployIt, you can deploy from one environment or location to multiple environments and locations while using **just one iDeployIt license.**
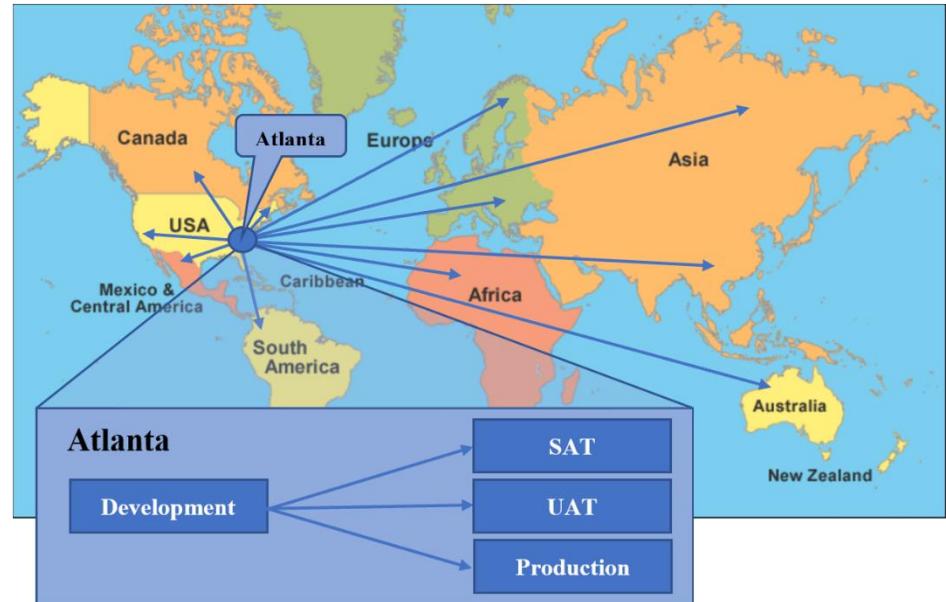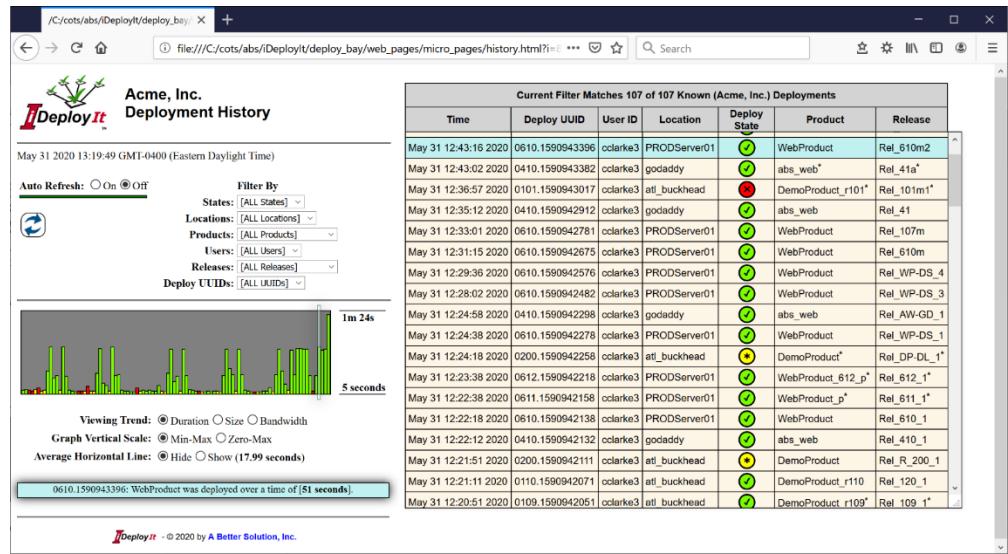
Installation Quick Start and Test Guide

This Quick Start Test Guide is designed to get the iDeployIt product installed and you through your first test deployment to confirm your installation

Installation and Test Steps

The iDeployIt product is distributed as either a Windows zip file or UNIX tar.gz file. The product footprint is under 5 Meg. After downloading the archive file extract it to your desired installation directory. The install includes a demo product to verify the installation. The steps are the same for UNIX and Windows architectures. The binaries for all architectures are included as well as both shipper and receiver software. Follow these steps to install iDeployIt and perform a local installation testing demo deploy.

1. Download the software at: https://www.abs-consulting.com/_products/download.shtml?4.

2. Create an **Installation Directory,** we will refer to this as the **{Install_home}** directory. The {Install_home} directory should NOT have embedded spaces in the path. Next, extract the contents of the downloaded *zip* or *tar.cz* file to the {Install_home} directory. Extract the contents of the archive file to this directory.

   - For Window: use the "extract all…" (or similar) from the pulldown menu of the selected zip file

   - For Unix: use one of the commands below:

     tar -xvzf iDeployIt_Install_home.tar.gz
   or
     gzip -d < iDeployIt_Install_home.tar.gz | tar xvf –
   or
      gunzip < iDeployIt_Install_home.tar.gz | tar xvf –

   After extraction, the{Install_home} directory will contain the "abs" directory and sub-directory "iDeployIt". In the **{Install_home}/abs/iDeployIt** will contain the iDeployIt product and this directory will be referred to as **{iDeployIt_home}.**

   In {iDeployIt_home} there are a few important directories and your license file (iDeployIt_lic.txt).

Reference	Location	Purpose
{Install_home}		Root directory where you extracted the zip or tar file
{iDeployIt_home}	**{Install_home}**\abs\iDeployIt	Hold iDeployIt data, executables and web pages
{iDeployIt_bin}	**{iDeployIt_home}**\bin	Holds all iDeployIt executables
{iDeployIt_etc}	**{iDeployIt_home}**\etc	Holds miscellaneous iDeployIt scripts
{iDeployIt_depot}	**{iDeployIt_home}**\deploy_bay	Holds iDeployIt configuration and data files
{iDeployIt_demo}	**{iDeployIt_home}**\demo_test	Hold iDeployIt demo deployment and data

Note: It is advisable to place the {iDeployIt_bin} directory in your *path* environment variable so it is easier to call the iDeployIt executables.

3. Navigate to the {iDeployIt_bin} directory which contains a separate sub-directory for each supported architecture. Copy the complete contents of the appropriate {iDeployIt_bin}\{arch} sub-directory to the {iDeployIt_bin} directory.

   AIX

   HPUX

   RedHat_CentOS

   sun_solaris

   SUSE

   ubuntu

   windows

   This will leave you with the binaries and scripts for your architecture in the {iDeployIt_bin} directory. After successfully testing the install, feel free to remove any unneeded architecture sub-directories.
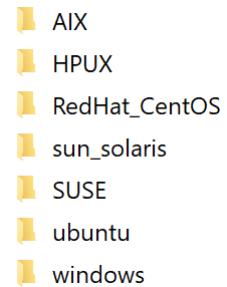
   If installing on a Unix platform, make sure the binaries have execute permissions applied. Use the ***chmod*** command to apply the permissions if needed.

   ```
   chmod 755 iDeployIt_*
   ```

   Remember {iDeployIt_home} was determined when you extracted the iDeployIt distribution in the first step. If the distribution is not where you want it, move it to a desired location now before proceeding with the iDeployIt_config program.

4. Navigate to the {iDeployIt_bin} directory from a command or terminal window. Execute the **iDeployit_config** program with the {iDeployIt_home} as a parameter.

   ```
   iDeployIt_config c:\cots\abs\iDeployIt
   ```
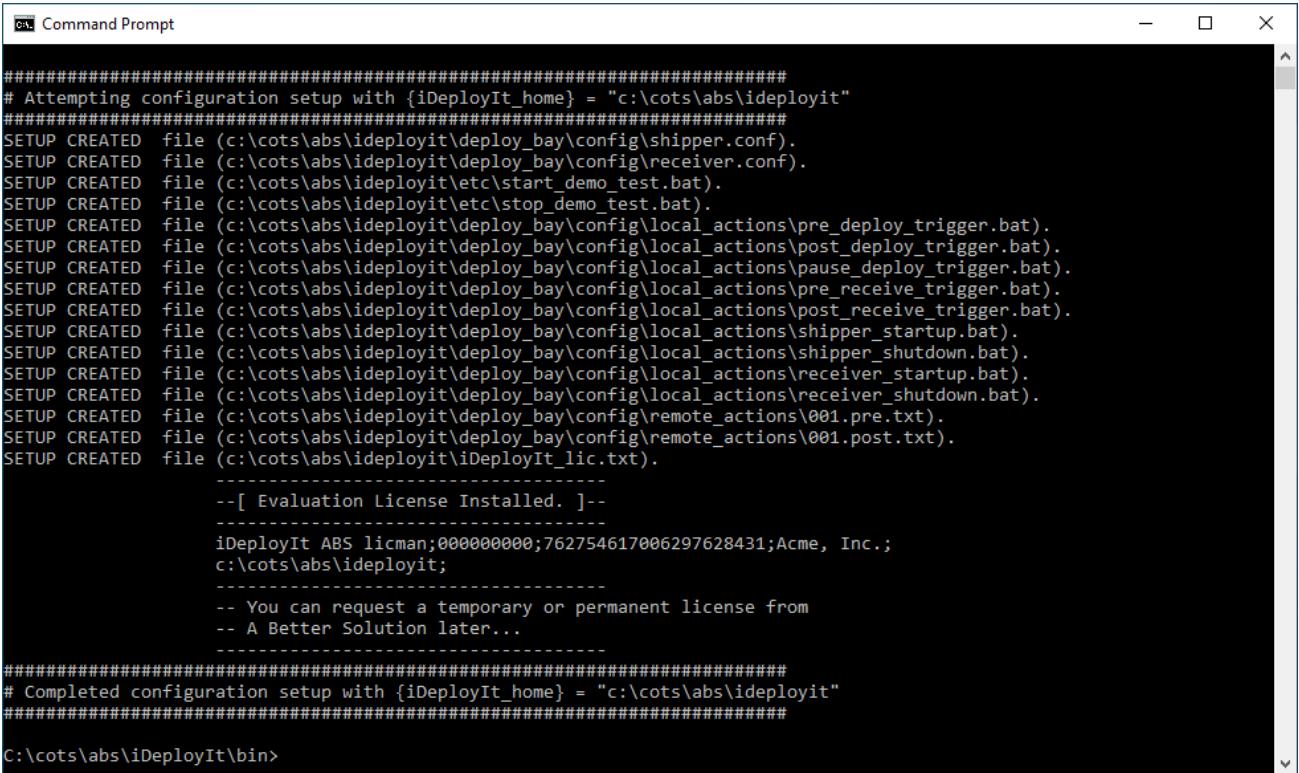
```
Command Prompt                                                  —    □    ×

#####################################################################
# Attempting configuration setup with {iDeployIt_home} = "c:\cots\abs\ideployit"
#####################################################################
SETUP CREATED  file (c:\cots\abs\ideployit\deploy_bay\config\shipper.conf).
SETUP CREATED  file (c:\cots\abs\ideployit\deploy_bay\config\receiver.conf).
SETUP CREATED  file (c:\cots\abs\ideployit\etc\start_demo_test.bat).
SETUP CREATED  file (c:\cots\abs\ideployit\etc\stop_demo_test.bat).
SETUP CREATED  file (c:\cots\abs\ideployit\deploy_bay\config\local_actions\pre_deploy_trigger.bat).
SETUP CREATED  file (c:\cots\abs\ideployit\deploy_bay\config\local_actions\post_deploy_trigger.bat).
SETUP CREATED  file (c:\cots\abs\ideployit\deploy_bay\config\local_actions\pause_deploy_trigger.bat).
SETUP CREATED  file (c:\cots\abs\ideployit\deploy_bay\config\local_actions\pre_receive_trigger.bat).
SETUP CREATED  file (c:\cots\abs\ideployit\deploy_bay\config\local_actions\post_receive_trigger.bat).
SETUP CREATED  file (c:\cots\abs\ideployit\deploy_bay\config\local_actions\shipper_startup.bat).
SETUP CREATED  file (c:\cots\abs\ideployit\deploy_bay\config\local_actions\shipper_shutdown.bat).
SETUP CREATED  file (c:\cots\abs\ideployit\deploy_bay\config\local_actions\receiver_startup.bat).
SETUP CREATED  file (c:\cots\abs\ideployit\deploy_bay\config\local_actions\receiver_shutdown.bat).
SETUP CREATED  file (c:\cots\abs\ideployit\deploy_bay\config\remote_actions\001.pre.txt).
SETUP CREATED  file (c:\cots\abs\ideployit\deploy_bay\config\remote_actions\001.post.txt).
SETUP CREATED  file (c:\cots\abs\ideployit\iDeployIt_lic.txt).
                ------------------------------------
                --[ Evaluation License Installed. ]--
                ------------------------------------
                iDeployIt ABS licman;000000000;762754617006297628431;Acme, Inc.;
                c:\cots\abs\ideployit;
                ------------------------------------
                -- You can request a temporary or permanent license from
                -- A Better Solution later...
                ------------------------------------
#####################################################################
# Completed configuration setup with {iDeployIt_home} = "c:\cots\abs\ideployit"
#####################################################################

C:\cots\abs\iDeployIt\bin>
```

   This initialized your configuration files and created example triggers and scripts that can be modified or used as examples for future changes you might make. It also installed the initial evaluation license key and installed your initial deployment key definition that will be used in your first local deployment.

8

5. Edit the shipper.conf file which is located in the **{**iDeployIt_depot**}**\config directory.

It contains the first **<u>deployment key definition</u>** to perform a deployment against.   You will learn more on deployment key definitions later, but for now focus on these two items. First, the **Source Deployment Location**. This is an example product that was include in you iDeployIt distribution.

`001>``C:\cots\abs\iDeployIt\demo_test``>``{Test_Deploy_location}``>DemoProduct>demo_location>hold`

Second, the **Destination Deployment Location**. This is where your deployment from the Source will be deployed on your machine. Edit the Destination Deployment Location so that it represents an *existing empty* directory on your local machine. For example, "c:\some_dir" is a directory I created to hold my test deployment. I modified the deployment key definition by changing its **Destination Deployment Location** like below:

`001>C:\cots\abs\iDeployIt\demo_test>``c:\some_dir``>DemoProduct>demo_location>hold`

Create the directory **c:\some_dir** (or another to suit your environment.) Save and close shipper.conf.

6. Navigate to {iDeployIt_etc} directory from a command or terminal window and execute the **start_demo_test.bat** (for Windows) or **start_demo_test.sh** (for UNIX).
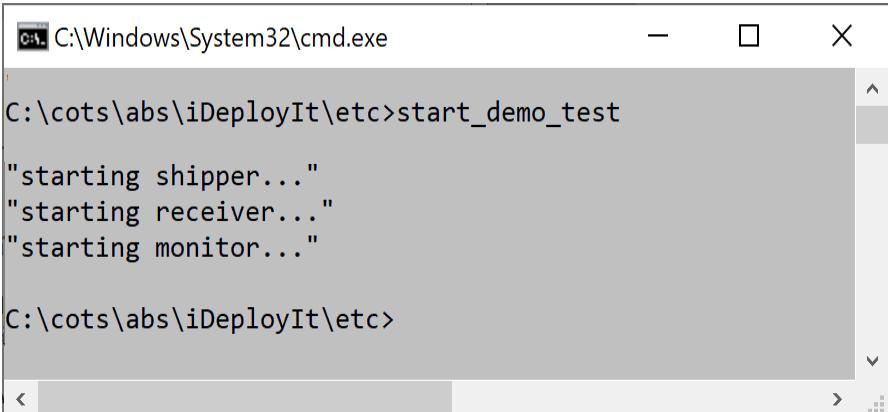
```
C:\Windows\System32\cmd.exe                    —    □    ✕

C:\cots\abs\iDeployIt\etc>start_demo_test

"starting shipper..."
"starting receiver..."
"starting monitor..."

C:\cots\abs\iDeployIt\etc>
```

This will start the **iDeployIt_shipper**, **iDeployIt_receiver** and **iDeployIt_monitor** services.

You would normally start the *shipper*, *receiver* and *monitor* as *scheduled tasks* on Windows or as *cron jobs* on UNIX, but the **start_demo_test** and **stop_demo_test** scripts are provided as a convenience for this Quick Start.

7. Perform your first deployment. Navigate to {iDeployIt_bin} directory from a command or terminal window and execute the iDeployIt_deploy command.

> **`iDeployIt_deploy {iDeployIt_home} 001 REL_DEMO_1.0`**

or

> **`iDeployIt_deploy {iDeployIt_home} DemoProduct demo_location REL_DEMO_1.0`**

```
C:\cots\abs\iDeployIt\bin>iDeployIt_deploy.exe c:\cots\abs\iDeployIt 001 REL_DEMO_1.0
Queued deployment key = 1 Product = "DemoProduct" version = "REL_DEMO_1.0" to location "demo_location"

C:\cots\abs\iDeployIt\bin>
```

This will deploy the test deploy application included with the product to the local directory you defined when modifying the "001>" deployment definition in shipper.conf.

Examine the results of the demo deployment. Notice how the page at the **Destination Deployment Location**

001>C:\cots\abs\iDeployIt\demo_test>c:\some_dir>DemoProduct>demo_location>hold

differs from page at the **Source Deployment Location**

This is because as part of the deployment we not only distributed the product, but also modified the product as part of the deployment using **Deployment Triggers.**

You will see how to use Deployment Triggers Later.
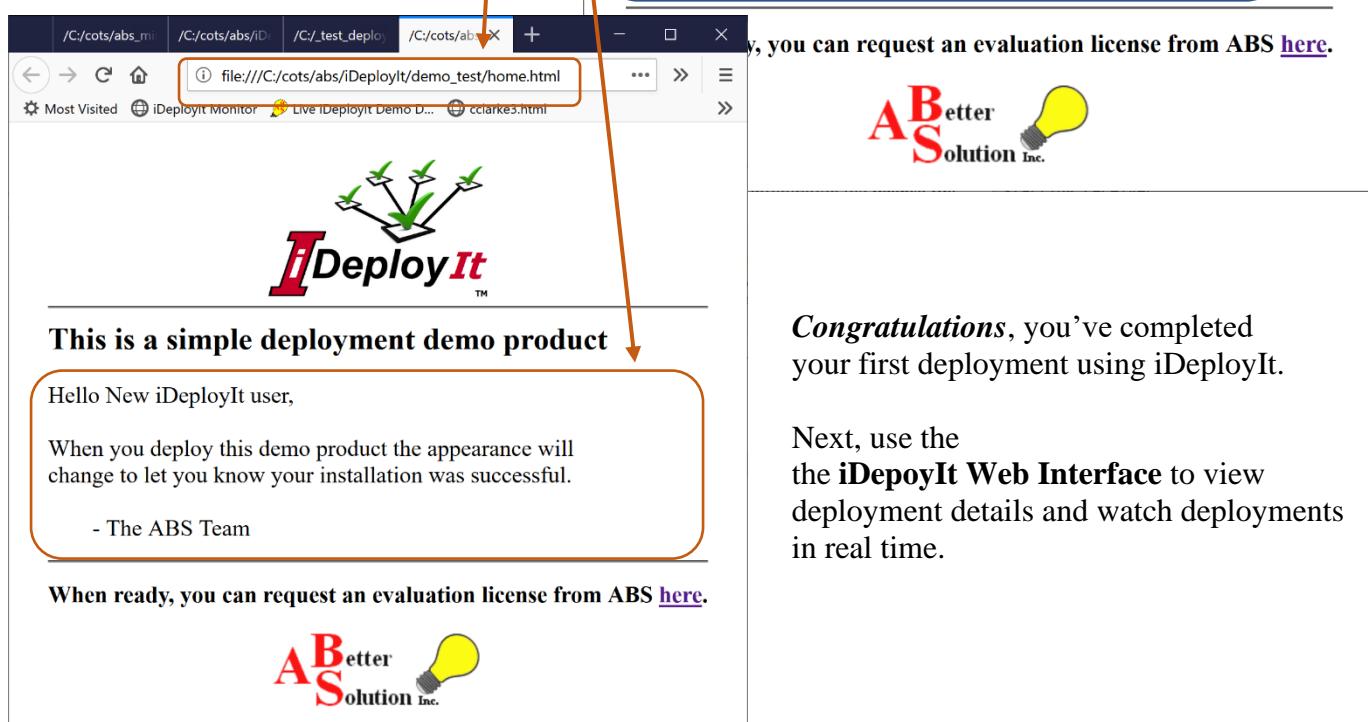
This is a simple deployment demo product

Hello cclarke3,

You released product DemoProduct (REL_DEMO_1.0)

to location demo_location
on Sat 04/11/2020 23:06:12.59.

y, you can request an evaluation license from ABS here.

This is a simple deployment demo product

Hello New iDeployIt user,

When you deploy this demo product the appearance will change to let you know your installation was successful.

- The ABS Team

When ready, you can request an evaluation license from ABS here.

*Congratulations*, you've completed your first deployment using iDeployIt.

Next, use the the **iDepoyIt Web Interface** to view deployment details and watch deployments in real time.

This is a simple local deployment that provides a small amount data and a platform to learn about the features of iDeployIt. It creates web pages that allow you also to explore the web interface.

You may notice license warning when executing a deployment or with subsequently created web pages. Ignore these for now. iDeployIt is distributed with all functions enabled with an evaluation key. Request a temporary key at any time to inhibit these warning as described in the **Evaluation License Registration** section.

Performing your first deployment also initiated the **iDepoyIt Web Interface**. Use the web interface to inspect deployments and watch currently deploying deployments in real time.

8. Open the web interface {iDeployIt_depot}\web_pages\iDeployIt.html with any html browser. You should be able to view Deployment Details for the DemoProduct Release REL_DEMO_1.0 deployment. Inspect the intuitive interface or view Web Interface details in the section entitled **Using The GUI Interface**.
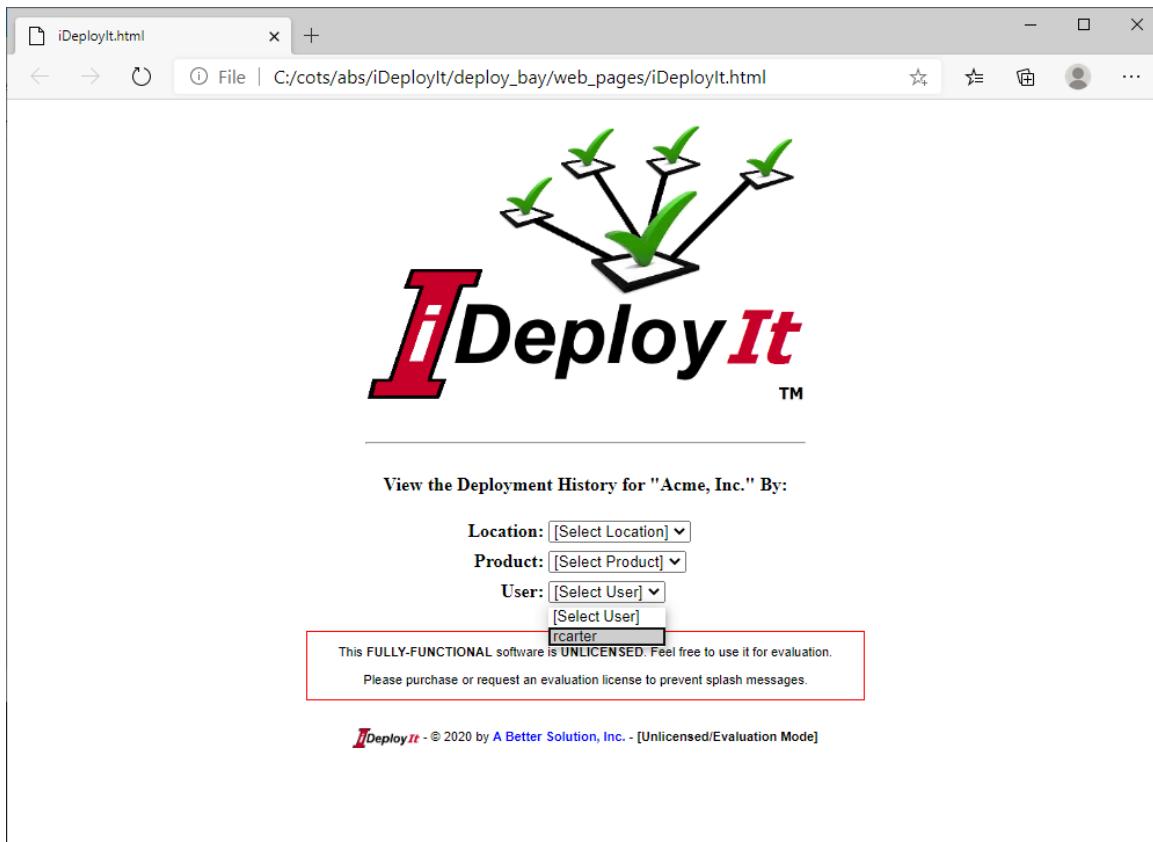
Evaluation License Registration

Use of an evaluation license will cause random *Splash Messages, banners and dialogs within the logs, command line and* the iDeployIt Web GUI. The 30 day evaluation license provides full functionality of iDeployIt.
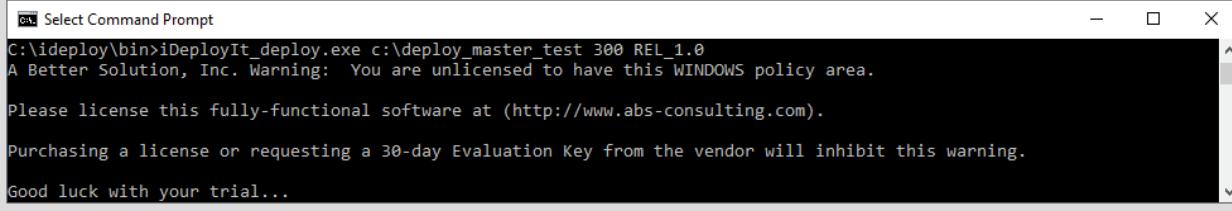


```
C:\ideploy\bin>iDeployIt_deploy.exe c:\deploy_master_test 300 REL_1.0
A Better Solution, Inc. Warning:  You are unlicensed to have this WINDOWS policy area.

Please license this fully-functional software at (http://www.abs-consulting.com).

Purchasing a license or requesting a 30-day Evaluation Key from the vendor will inhibit this warning.

Good luck with your trial...
```

Thank you for evaluating iDeployIt!

You are using an UNLICENSED, but FULLY-FUNCTIONAL iDeployIt deployment depot. Feel free to use it for evaluation.

Please contact A Better Solution, Inc. (https://abs-consulting.com)
to purchase this software or request an evaluation license that removes these splash messages..

OK
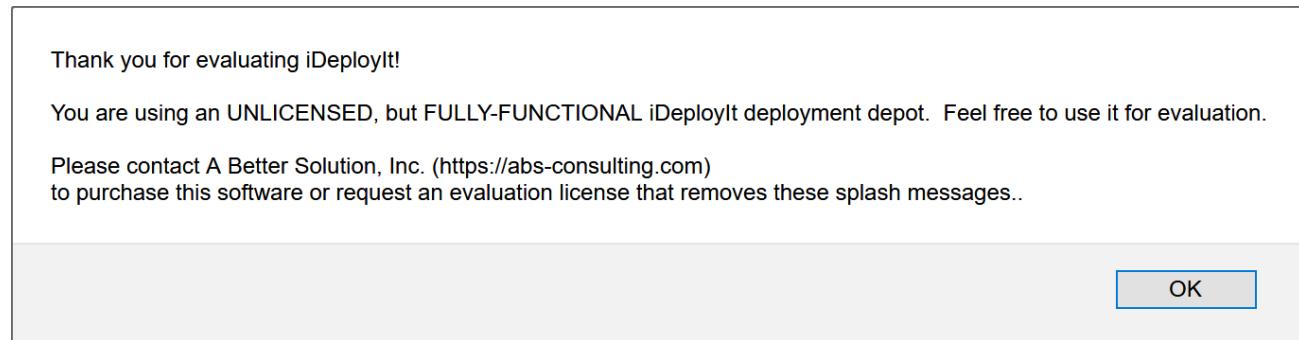
A license is required for the sending server only. See the section, Requesting License, for instructions to request a license.

License Configuration

A valid license is required for the sending server to inhibit the splash messages and banners. The evaluation license (and any subsequent temporary or permanent license) is located in the iDeployIt_lic.txt file in the {iDeployIt_home} directory.

The contents of the license file distributed with the product is below:

```
iDeployIt ABS licman;000000000;111111111;Acme, Inc.;
{iDeployIt_home};
```

You provide the three (3) **yellow highlighted** parts and ABS returns the one (1) **blue highlighted** part.

1) Edit the {iDeployIt_home} string and replace it with the actual {iDeployIt_home} directory value from your install.

2) Edit the Company Name string ("Acme, Inc." in this example) to the Company Name you wish to use. Use a string that indicates who or where you are and distinguishes your region/depot from another that might be in your topology. For example, if **Acme, Inc.** has only one region/depot (quite often the case) then "Acme, Inc." is a good name to use.

This would display "Acme, Inc." in various places on the Web interface. Choose this string carefully as it is used to generate the license.

```
iDeployIt ABS licman;000000000;111111111;Acme, Inc.;
c:\cots\abs\ideployit;
```
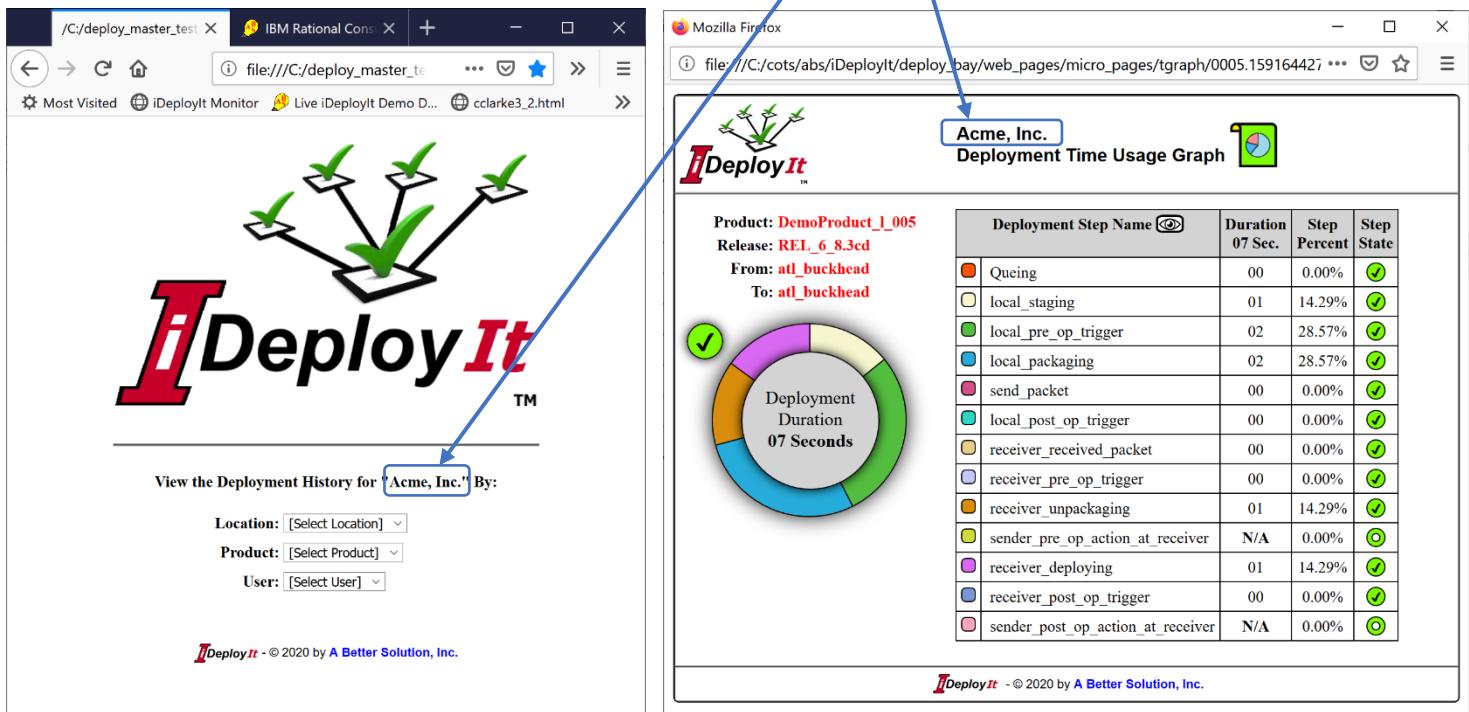

3) Generate the **Unique Depot Identifier (UDI)** and provide to ABS. The iDeplotIt product is distributed with the UDI value of "111111111". If you performed the Quick Start during installation this value was already calculated for you, placed in your license file and you are ready to request an evaluation license. Please proceed to the section Requesting License.

If you have *not* already generated your **Unique Depot Identifier (UDI)** then you must send ABS the value generated by the **iDeplotIt_deploy -hostid** command.

```
C:\cots\abs\bins>iDeployIt_deploy -hostid

When requesting a license from A Better Solution, Inc., provide the
        hostid = 76219291462886140328
```

Producing the license key below:

```
iDeployIt ABS licman;000000000; 76219291462886140328;Acme, Inc.;
c:\cots\abs\ideployit;
```

4) Navigate the to the ABS License Request website to create a license request.

https://abs-consulting.com/_products/products_ideployit.shtml?request_license

5) Take the license you received from ABS and replace the existing iDeployIt_lic.txt located in the {iDeployIt_home} directory.

6) Restart the iDeployIt services to have the license recognized in the web GUI by using the **stop_demo_test** and **start_demo_test** programs in the **{iDeployIt_etc}** directory.

License Requirements

Many deployment products charge for a sending server and then additional costs for receivers or remote agents. However, iDeployIt is licensed per sender/shipper, there is no charge for receivers or remote agents. With iDeployIt, deploy from one environment or location to multiple environments or locations with one license.

So if Acme Company has an "Atlanta" location that delivers software to multiple locations around the world and the location itself had a few environments to deliver to like *SAT*, *UAT* and *Production*, Acme Company would still only need *one* license of iDeployIt to deploy product to these different locations and environments.
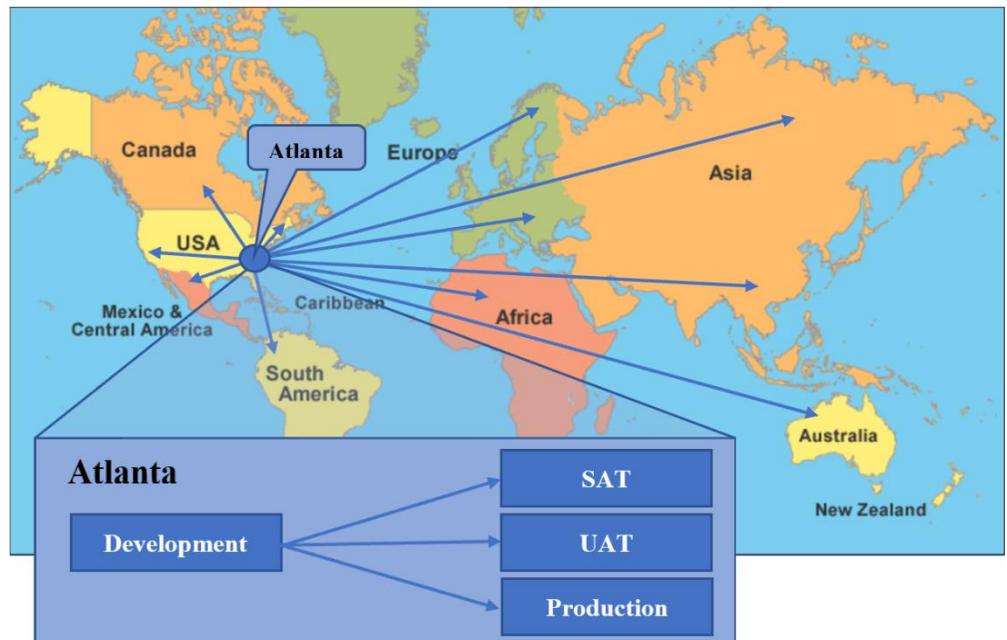
Requesting Licenses from ABS

To request a temporary or permanent license for iDeployIt, please complete the license request form located here:

https://abs-consulting.com/_products/products_ideployit.shtml?request_license

Using the GUI Interface (GUI Topology)

This section describes the **iDeployIt GUI Topology**. The GUI Topology was designed so that you are never more than 2 clicks away from the information you want to see. Windows that refer to or can display information for *multiple deployments* are referred to as ***Pages*** while Windows that display information on a *single deployment* are referred to as ***Dialogs***. Pages can also contain ***filters*** that limit which deployments are displayed.   See the Page and Dialog transition relationship below.
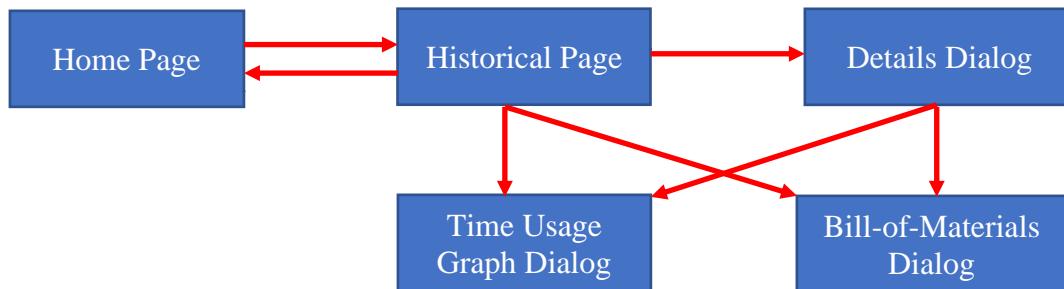

iDeployIt Home Page

This is the main page GUI of iDeployIt. From here view the history of past deployments and see the status of current deployments.
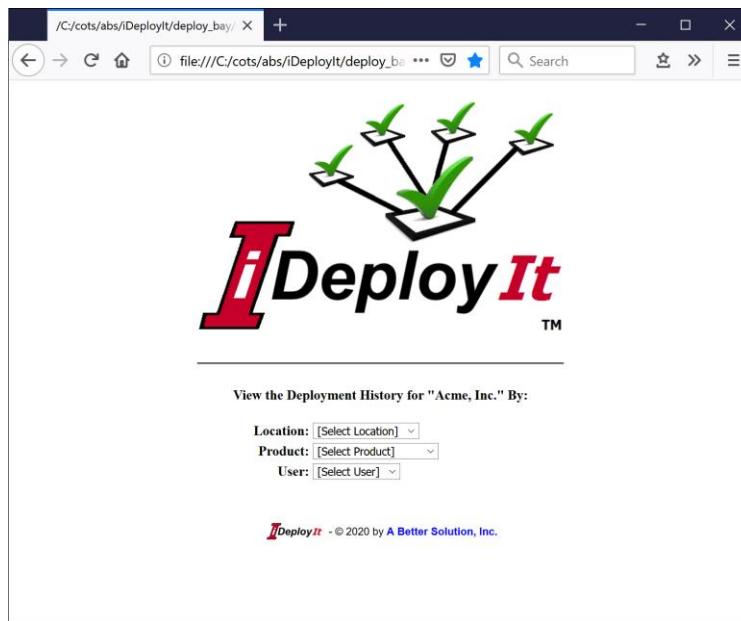
Users can view all deployments to a **location**, for a **product** or initiated by a **user** by selecting a pulldown.

Once selected, the user is shown the **Deployment Historical View** for the selected item.

Note: Pulldowns are empty until you run your first deployment.

Deployment History View Page

Select a *view* from the **iDeployIt Home Page** to display the **History View Page**. In this example, with no filters active, we see *all* the deployment history and current status of deployments for the organization.

Use the **Deployment Filter** to change what deployments are displayed in the table and graph. Filters are additive so only items that pass each filter are displayed. The filters take on the initial selection of the requested filter selected in the Main GUI.

Each of the characteristic pulldowns contain appropriate values that can be used to further filter what is displayed in the **Matching Deployments Table** as listed in the table below.

States	- **Deploying** – Deployments currently *running* or *paused*.
	- **OK** – Deployments completed without *errors* or *warnings*.
	- **Warning** – Deployments completed with *warnings*.
	- **Error** – Deployments completed with *errors*.
	- **Completed** – Deployments across all *completed states* (OK, Warning or Error).
	- **Paused** – Deployments in the *paused state*.
Locations	- A dynamic list of previous deployed to *locations*
Products	- **[CURRENT Products]** – The currently deployed Product of a given Release at a given Location.
	- A dynamic list of previous deployed to *releases*.
Users	- A dynamic list of *users* that have previously deployed products.
Releases	- **[CURRENT Releases]** – The currently deployed Release of a given Product at a given Location.
	- A dynamic list of previous deployed *releases*.
Deploy UUID	- A dynamic list of previous deployed to *UUID Groups/Deployment Key IDs*.

The **Deployments Trend Graph** show the last *100 completed* deployments that match the current filter. Use the associated radio buttons to change the display characteristics of the graph.

Select the **Viewing Trend** radio buttons to change what is displayed in the *trend graph*.

Duration – View the deployment duration trends of the last 100 matching deployments. Note: The colors of the entries (in this and all the graph variations) indicate the state of the associated deployment and do not indicate duration magnitude in any way.	
Size – View the deployment size trends of the last 100 matching deployments.	
Bandwidth – View the bandwidth trends of the last 100 matching deployments.	
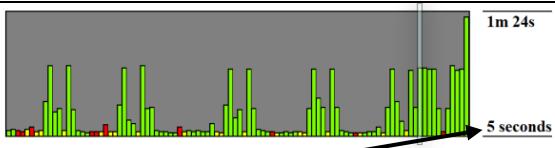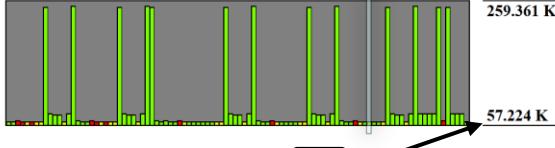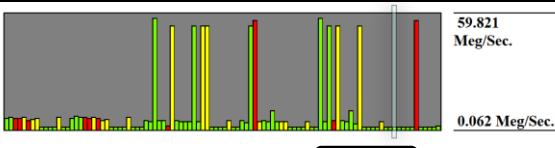
Select the **Graph Vertical Scale** radio buttons to change the scaling options for the *trend graph*.

Min-Max – Scale the graph based on the *minimum* and *maximum* values from the data.	
Zero-Max – Scale the graph based on the minimum being zero to the maximum values from the data.	
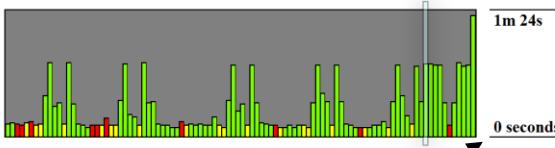
Select the **Average Horizontal Line** radio buttons to show or hide a line that depicts the average of the data displayed in the *trend graph*.

Hide – Hide the average line from the trend chart.	
Show – Show the average line in the trend chart.	

The **Graph Details Window** shows identifying details for a graph item when the mouse if rolled over an item in the *trend graph* or over an associated row in the *deployment table* which also has an entry in the trend graph in the pre-selected item color. ☐☐☐ The actual information displayed for the associated deployment is either *Duration*, *Size* or *Bandwidth* information is indicated by the current **Viewing Trend** setting.
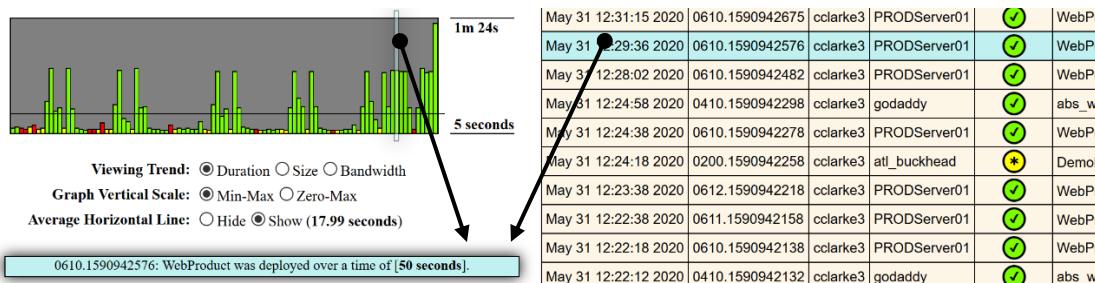
If the mouse of over a row that **does not** have an asosciated graph row because it is currently running or failed with error before completing the minimum steps it will appear in the table default row color.
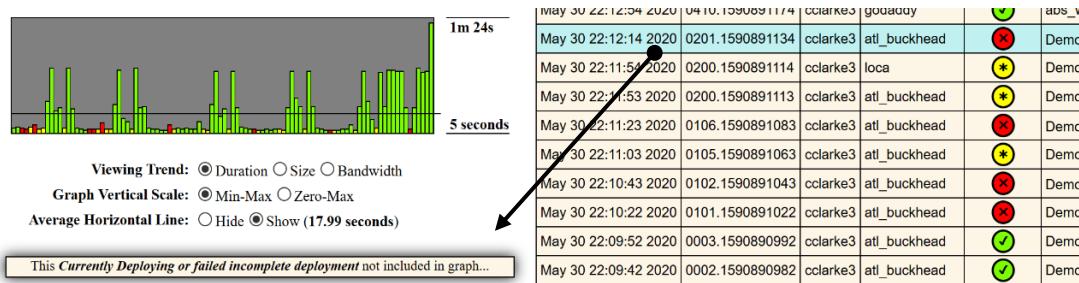
Clicking on a *graph item*, *table row* or a ***populated*** *graph details window* opens the **Deployment Detail View** for the associated deployment.

Deployment Detail View Dialog

From the **Deployment History** view, select a single deployment from the view table or the trend graph to display additional details for the selected deployment in the **Deployment Detail** view.

The upper portion of the view gives the **summary information** about the deployment.

The **Deployment State Widget** graphically depicts the thirteen (13) individual steps of the deployment, showing the state of that step for both the sender and receiver and the deployments total state.

The **progress pointer** ↑ shows the deployments current progress.

The **step pointer** points to a selected step, displaying the **step name window**, which in-turn highlights the associated entries in the **Deployment Log Table**.

Selecting the **Bill of Materials Icon** opens the **Bill of Materials Dialog** to see details about the file contents of the deployment.

Selecting the **Time Usage Graph Icon** opens the **Time Usage Graph Dialog** to see time allocation details of each of the deployment steps.

Deployment State Legend		Step State Legend	
⚪	- Unstarted	⬜	- Unstarted
✅	- Completed with success	🟢○	- Skipped or Empty
○	- Deploying progress (animated - spinning)	🟢●	- Completed with success
⏸	- Paused – waiting for user to resume	🔵	- Deploying progress (animated - spinning)
✳	- Completed with warning	🔵⏸	- Paused – waiting for user to resume or abort
❌	- Completed with error	🟡●	- Completed with warning
		🔴●	- Completed with error

Bill of Materials Dialog

View the **Bill of Materials Dialog** by selecting the Bill of Materials Icon from the **Deployment Detail View** page.

View a list of files, directories and any **Pre_deployment** or **Post_Deployment** triggers that were included in the deployment package as well as general deployment information (i.e. number of files, Product Name, Release Name, Origination Location, Staged Location, Deployment Time).

View the file and directory listing in three (3) ways:

- based on the **original source directory** (as defined in the **deployment_key** definition)
- based on the **staging directory** (as defined in the product name and release name)
- **relative** to the source or staging directory

So, given the *deployment_key*:

    710>**c:\demo_abs_web\web_dir**>/live/http/prod>WebProduct>PRODServer01>ship

and an **{iDeployIt_depot}** directory of "c:\cots\abs\iDeployIt\deploy_bay", line [05] in the listing would be displayed according to the table below:

Selected	Appearance
⦿ Source Dir	**[05] file: c:\demo_abs_web**\web_dir\images\1.jpg
⦿ Staging Dir	**[05] file:** c:\cots\abs\iDeployIt\deploy_bay_staging**WebProduct\Rel1.7b**\web_dir\images\1.jpg
⦿ Relative Dir	**[05] file:** web_dir\images\1.jpg

Time Usage Graph Dialog

View the **Time Usage Graph Dialog** by selecting the Time Usage Graph Icon from the **Deployment Detail View** page.

From here view a breakdown of performance information, duration and percent of total duration, for each step for the deployment.

Click on a **time pie slice** or an associated table row to display **Deployment Log Information** on that deployment step.

The information displayed is similar to the information displayed in the **Deployment Log Table** for the associated step in the **Deployment Detail View** for that deployment.

Hovering over a *deployment step name* highlights the *associated pie slice* and vice versa.

There are thirteen (13) **predefined deployment steps**, Steps are shown unless they are empty, skipped or not entail a significant amount of time; steps that take less then a second will not be displayed. By defaut all table rows are shown even when a pie slice is not is not displayed.

Click on the ⊙ icon to hide rows tha do not have a slice displayed. Click on the ⊘ icon to again show all rows.
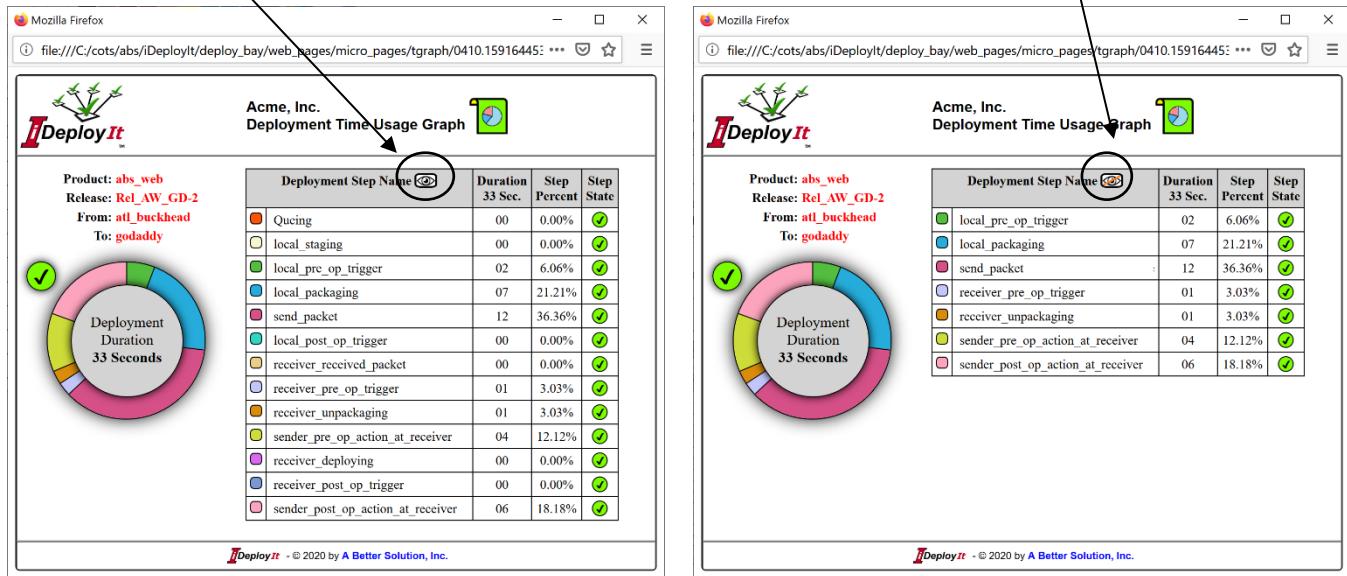
When defining the deployment to include a pause between the creation and sending of a deployment package (until a human or other process resumes the deployment) the actual time of the send step includes the "paused" time.   The **send_packet row** shows an icon ▯ to indicate that the deployment has a defined required pause.

When a paused
send_packet slice
or row is selected,
an additional **pie-chart**
shows how long it's
been paused
.

This pie-chart is a child
of the send_packet slice
indicating how much
time of the send was
actually spent waiting for
the external process to
"resume" the
deployment.
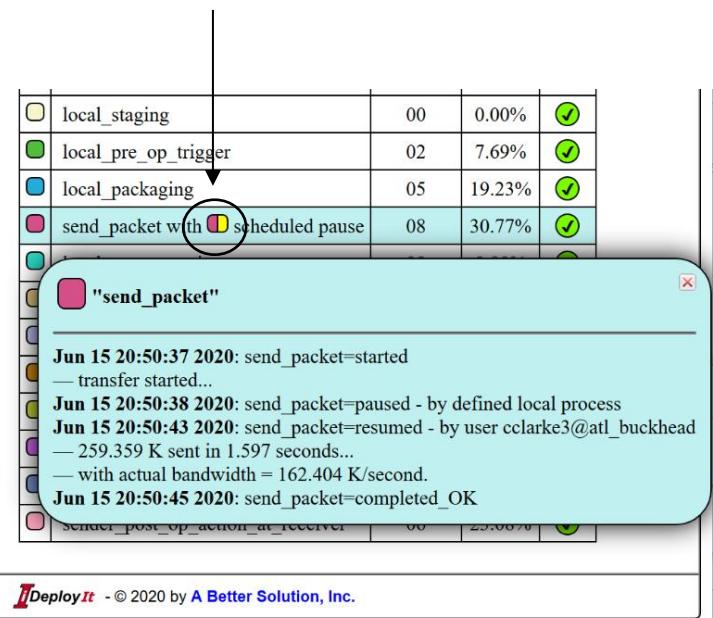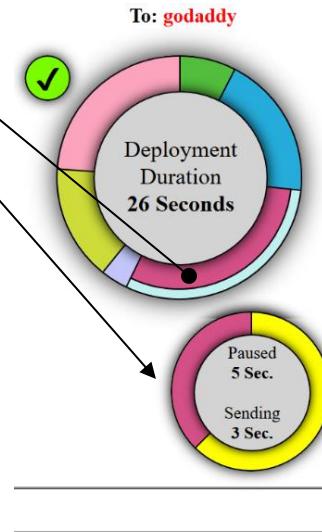
The Deployment Process

The deployment is broken into 13 distinct tasks with six (6) tasks responsible for packaging and sending a deployment to a location; seven (7) tasks to unpackage and install the deployment at its destination. Thirteen steps are predefined automatically when creating a deployment definition. Empty steps will be skipped.

Every Deployment requires the following steps at the **origination site**:
- **Queue** the Deployment
- **Stage** (or use previously staged if a re-deploy of a previously released release)
- **Package** the deployment for transfer to destination
- **Send** the Packaged Deployment to the remote location

These steps are required at the destination site:
- **Receive** the Deployment at the destination location
- **Unpackage** the Deployment as the destination
- **Placement** of the package in the final destination directory at the destination location
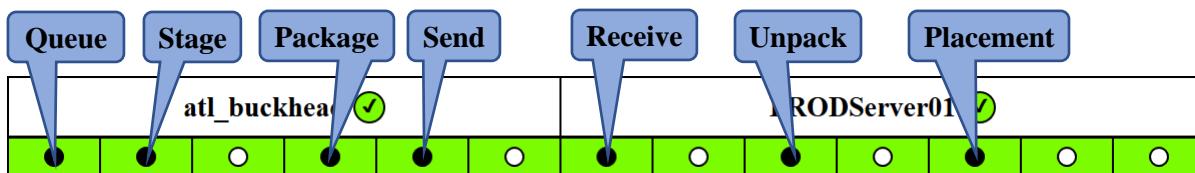
These steps are created when the deployment key definition is configured.

Add optional deployment customizations between the required steps with deployment triggers. There are two (2) sender-side triggers authored by the send, two 2) receiver-side triggers authored by the receiver and two (2) receiver-side triggers authored by the sender than can be executed if the receiver allows the execution of sender-authored triggers. Use triggers to augment or customize the deployment or to calculate and return a warning/failure code if some condition is met.
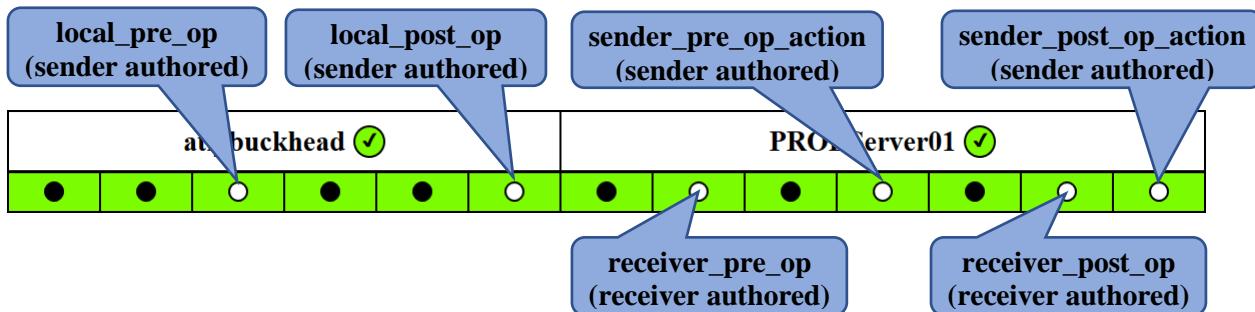
A summary of the steps is provided below:

Sending Phase		Receiving Phase	
Task	**Description**	**Task**	**Description**
Deployment Queued (required)	Deployment queued for processing by the shipper process.	**Package Received (required)**	Deployment package is received at its destination.
Local Staging (required)	Deployment is versioned and placed in the staging location.	**Receiver Pre-Deploy Check (optional) [receiver authored]**	receiver defined triggers executed before unpackaging of deployment package begins.
Pre-Send Check (optional) [sender authored]	User defined triggers are executed before the packaging task starts.	**Unpackage (required)**	Receiver process unencrypts and unpackages deployment.
Packaging (required)	Shipper processes encryption and packaging of deployment for sending.	**Sender Pre-Deploy Check (optional) [sender authored]**	Sender defined triggers executed before the deployment task begins. If permitted by the receiver.
Send Package (required)	Deployment package is sent to its destination location.	**Deploy (required)**	The deployment package is deployed to its destination location.
Post-Send Check (optional) [sender authored]	User defined triggers are executed after deployment package has been sent to the destination location.	**Receiver Post-Deploy Check (receiver authored)**	Receiver defined triggers executed at the receiver after the deploy.
		Sender Post-Deploy Check (optional) [sender authored]	Sender defined triggers executed at the receiver after the deploy if permitted by the receiver

Read more on triggers in the section entitled **Creating Deployment Triggers**.

24

Configuring the Shipper

The **shipper.conf** file is located in the **{iDeployIt_depot}**\config directory. Modify the file to reflect your environment by changing the following required configuration variables location_name and **shipper_id**.

Example:

```
location_name=atlanta_office
shipper_id=cclarke3
```

All shipper.conf **configuration parameters** are defined below.

Parameters	Purpose
location_name	Names the location for the site (what this site calls itself)
shipper_id	Names the userid that is allowed to run the shipper for this associated depot. This userid should have read access to the source location.
pre_send_trigger (optional)	Optional script or executable that is called immediately before any requested shipping packet is created. If the script exits with a zero-return code, then the packet is created and immediately shipped (unless it is destined to the _hold_out directory). If the script exits with a status greater than 100 (warning) then the packet is built and the appropriate *warning* message is written to the iDeployIt **shipper logs**. If the script exits with a return code from 1-100 (failed) status then the packet is not built and the appropriate *error* message is written to the iDeployIt **shipper logs**. If defined, this script or program must reside in the *{iDeployIt_depot}\local_triggers directory* in the associated iDeployIt depot.
post_send_trigger (optional)	Optional script or executable that is called immediately after any requested shipping packet is created and shipped (or created and placed in the _hold_out directory), even if the packet did not process successfully (providing an opportunity to clean up any data changes made in the pre_ship_trigger_script). If the script exits with a non-zero (failed) status then the appropriate *warning* message is written to the iDeployIt **shipper logs**.. If defined, this script or program must reside in the *{iDeployIt_depot}\local_triggers directory* in the associated iDeployIt depot.
pause_deploy_trigger (optional)	Optional script or executable that is called immediately before any requested shipping packet is created and shipped (or created and placed in the _hold_out directory). If the script exits with a non-zero status then the appropriate *warning* message is written to the iDeployIt **shipper logs**. If defined, this script or program must reside in the *{iDeployIt_depot}\local_triggers directory* in the associated iDeployIt depot.
startup_trigger (optional)	Optional script or executable that is called immediately after the successful startup of an iDeployIt shipper process. If the script exits with a zero return code, then the shipper process continues. If the script exits with a non-zero (failed) status then the shipper startup is considered denied by that script and the appropriate *error* message is written to the iDeployIt **shipper logs**. This script or program must reside in the *{iDeployIt_depot}\local_triggers directory* in the associated iDeployIt depot.

shutdown_trigger (optional)	Optional script or executable that is called immediately after the successful shutdown of an iDeployIt shipper process. If the script exits with a zero return code, then the shipper process continues. If the script exits with a non-zero (failed) status the appropriate *error* message is written to the iDeployIt **shipper logs**. This script or program must reside in the *{iDeployIt_depot}\local_triggers directory* in the associated iDeployIt depot.
encryption (optional)	From the values ["standard", "peer_to_peer"] with a default of "standard" if omitted. This parameter determines if **"Standard iDeployIt Encryption"** is used or if an additional **Peer_To_Peer** iDeployIt **Encryption** is used. If this value is set to "peer_to_peer" then the iDeployIt shipper process uses the encryption key stored in the shippers associated encryption key file. The encryption key file must reside in the {iDeployIt_depot}*config\keys* directory in the associated iDeployIt depot.
transport_program (optional) (defaults to "ftp")	Names the program (usually ftp or sftp) that the shipper uses to transport the packet to the receiver. This program MUST be in the PATH of the process running the "shipper". Regardless of the transport program used, packets are compressed and encrypted. To use one of the different flavors of Secure FTP (sftp) refer to the section entitled **Using Secure FTP as the transport program**.
transport_class (optional) (defaults to 0)	This provides vendor specific implementation details of SFTP or FTP programs. By default, the transport class is 0 for ftp. To use one of the different flavors of secure FTP (sftp) refer to the section entitled **Using Secure FTP as the transport program**.

Creating Deployment Key Definitions

Deployment_keys are stored in the **shipper.conf** file and are "active" when the iDeployIt "shipper" is running.

Creating a deployment_key defines:

- The content of a deployment
- Which "location" the deployment is to be sent. This location MUST exist prior to deployment.
- Where at the destination is the deployment to be placed?
- If the new deployment is immediately scheduled for shipping or marked "hold for another process."
- If the deployment has a "pause" defined to allow human (or other) interaction before a build packet is sent to its destination (i.e. request a final view before packet deployment sends the package).

Deployment_key_definitions are placed in the shipper.conf file located in the {iDeployIt_depot}/config directory and are of the form:

```
deploy_id>Source_Location>Dest_Dir>Deploy_Label>[Deploy_locations][>ShipState][>PauseState]
```

Field Name	Field Syntax
deploy_id	["000"..."9999"]
Source_Location	Source location, file or directory, where the iDeployIt_shipper retrieves deployment contents.
Dest_Dir	The physical directory at each location to place the contents received from the source_location.
Deploy_Label	An arbitrary label to name the deployment package. This typically resembles the Product Name or Component being deployed.
Deploy_Locations	One or more locations to send the deployments. Each location will require a configuration of a **location.hostmap** file if the deployments are initiated from this machine and they will be sent to remote locations using an ftp or sftp transport class.
ShipState	*Optional:* "ship" or "hold". Determines if newly created packets for this key are immediately scheduled for shipping or placed in the *_hold_*out to wait for some other process or means (i.e. sneaker net). *Note*: The _hold_out directory can be used as a "receiver directory" for the local machine or a remote machine with network access to the directory. Useful for transferring deployment packets local or to remote machines without having to use an FTP or SFTP transfer protocol.
PauseState	*Optional:* Determines if a deployment will pause before "sending the packet to the remote location" state and require human intervention to continue.

For example: Given the "shipper" location "atlanta" has these entries in its *shipper.conf* file.

Define the ***ProductTest*** deployment from source, c:\ideploy\source to destination *c:\ideploy\destination\dev* at location ***DevServer01*** and ***hold*** for manual deployment.

```
100>c:\ideploy\source>c:\ideploy\destination\dev>ProductTest>DevServer01>hold
```

Define the ***ProductTest*** deployment from source, c:\ideploy\source to destination c:\ideploy\destination\dev at locations ***QAServer01*** and ***QAServer02***

```
200>c:\ideploy\source>c:\ideploy\destination\qa>ProductTest>QAServer01 QAServer02>ship
```

Define ProductTest deployment from source, c:\ideploy\source to destination /ideploy/destination/prod at location PRODServer01 and pause deployment for approval.

```
300>c:\ideploy\source>/ideploy/destination/prod>ProductTest>PRODServer01>ship>pause
```

View additional examples in the section entitled **Deployment Definition Examples.**

iDeployIt_shipper

The iDeployIt_**shipper** program creates and ships the iDeployIt packets based on *shipper.conf* configuration settings. Only one iDeployIt_shipper process can run per iDeployIt depot. The shipper.conf file is in the {iDeployIt_depot}\config directory.   The iDeployIt_shipper program can be started manually or within a startup script or cron job (for UNIX) or startup script or scheduled 'AT' job (for Windows). This process can only be started by the userid designated by the shipper_id configuration variable defined in *shipper.conf*file.

Starting/stopping the iDeployIt Shipper

The iDeployIt_shipper executable is in the {iDeployIt_bin} directory. See the command reference, ideployit_shipper, for full command syntax.

When started the iDeployIt_shipper runs in the background (unless the –wait switch is used). The iDeployIt_shipper MUST run as the appropriate shipper_id as defined in the associated **shipper.conf** file.

Use the **–wait** option to start the shipper process in the foreground causing the calling process or program to wait for the shipper to finish.

For UNIX servers, the iDeployIt_shipper should be started from the server startup scripts or equivalent.

For Windows servers, the iDeployIt_shipper should be started as a scheduled startup task that runs when the system boots.

Starting multiple iDeployIt_shipper processes

To improve performance or to distribute administration between multiple group members , start multiple iDeployIt_shipper processes by taking advantage of the [class] parameter passed to iDeployIt_shipper. Start these processes from multiple machines to further improve performance. When using multiple receiver processes and therefore multiple shipper.conf files there are some additional rules to consider:

- Each shipper.conf file must have its own dedicated view that is writable by the associated user in its shipper_id variable. Alternately the process could run at mutually exclusive times.

- Multiple iDeployIt shipper processes running for a single iDeployIt depot can be used to further improve performance. These processes can reside on different machines with different architectures to better take advantage of available CPU and bandwidth.

Configuring the Receiver

The **receiver.conf** file is located in the {iDeployIt_depot}\config directory. Modify for your environment by changing the following.

```
location_name=atlanta_office
receiver_id=cclarke3
```

The receiver configuration parameters are defined below.

Parameters	Purpose
location_name	Names the location for the site
receiver_id	Names the userid that can run the receiver for this associated depot. This userid should have read and write access to all resources it will deploy.
pre_receive_trigger (optional)	Optional script or executable called immediately before received packet is processed. If the script exits with a return code of zero then the packet is processed. If the script exits with a status greater than 100 (warning) then the packet is processed and the appropriate *warning* message is written to the iDeployIt **receiver logs**. If the script exits with a 101 or greater (failed) status then the packet is not processed and the appropriate *error* message is written to the iDeployIt **receiver logs**. This script or program resides in the *{iDeployIt_depot}\local_triggers directory* in the associated iDeployIt depot.
post_receive_trigger (optional)	Optional script or executable called **immediately after** received packet is processed, even if the processed packet did not complete. If the script exits with a non-zero (failed) status then the appropriate *error* or *warning* message is written to the iDeployIt **receiver logs**. This script or program resides in the *{iDeployIt_depot}\local_triggers directory* in the associated iDeployIt depot.
startup_trigger (optional)	Optional script or executable called immediately after the successful startup of an iDeployIt receiver process. If the script exits with a return code of zero then the receiver process continues. If the script exits with a status greater than 100 (warning) status then the receiver startup continues and the appropriate *warning* message is written to the iDeployIt **receiver logs**. If the script exits with a 101 or higher (failed) status then the receiver startup is denied by the script and the appropriate *error* message is written to the iDeployIt **receiver logs**. This script or program resides in the *{iDeployIt_depot}\local_triggers directory* in the associated iDeployIt depot.
shutdown_trigger (optional)	Optional script or executable called immediately after the successful shutdown of an iDeployIt shipper process. If called, an appropriate *warning*, *error* or *success* message is written to the iDeployIt **receiver logs** indicating the script's return code. This script or program resides in the *{iDeployIt_depot}\local_triggers directory* in the associated iDeployIt depot.
scheduled_inactivity (optional)	Defines a time that the receiver will *NOT* process packets. During this optionally defined **scheduled inactivity duration**, packets are still received from remote locations, but processing of those packets is postponed. If defined, two (2) unique times must be provided of the form: HH:MM-HH:MM Where each HH is '00'-'23' and each MM is '00'-'59'. The first HH:MM is the "start of the inactivity time" designator and the second HH:MM is the "end of the inactivity time" designator.
encryption (optional)	From the values ["standard", "peer_to_peer"] with a default of "standard" if omitted. This parameter determines if "Standard iDeployIt Encryption" is used or if an additional **Peer_To_Peer** iDeployIt **encryption** is used. If this value is set to "peer_to_peer" then the iDeployIt shipper process uses the encryption key stored in the shippers associated encryption key file. The encryption key file resides in the *{iDeployIt_depot}\config\keys* directory in the associated iDeployIt depot.

Creating Receiver Directories

iDeployIt deployment packets are received in deployment_receiver_directories. A directory location is embedded in the encrypted **location.hostmap** . The administrator should verify the existence of the deployment_receiver_directory and that access is granted.

The deployment_receiver_directory is a directory where deployment packets maybe delivered. This directory will be placed in the receiver.conf file on the {iDeployIt_depot}\config directory. For Example, this could be the FTP root or some sub-directory contained below.

The site administrator can issue *multiple* keys for different "receiving" directories, for example, if the "receiver" wanted to receiver all packets from Atlanta in the directory c:\iDeployIt\receiver\atlanta and all packets from London in the directory c:\iDeployIt\receiver\London.

A receiver directory definition contains 1-2 parameters or parameter groups separated by the '>' character:
- A *required* physical directory parameter
- An *optional* remote actions parameter

Physical Directory Parameter

The physical directory parameter is the physical location of the actual receiver directory.   Multiple receiver directories can defined in a receiver.conf file. There is no limit or additional cost for any iDeployIt receiver topology.

Remote Actions Parameter

The remote actions parameter is optional and determines if the receiver allows the sender to execute scripts at the receiver location. The valid values and what they mean are listed in the table below:

Remote Actions Parameter (values)	Meaning
allow_remote_actions	The default if no value provided – Receiver allows the sender to execute key-specific sender-sent scripts on the receiver location. (the id running the iDeployIt_receiver process).
allow_no_remote_actions	Receiver prohibits a sender from executing key-specific sender-sent pre and post deployment triggers at the receiver.

Configure Deployment Locations (using iDeployIt_hostentry)

The **iDeployIt_hostentry** executable is located in {iDeployIt_bin}. The iDeployIt_hostentry program generates a host_key which allows authentication without providing the "shipper" a userid, password, machine name or name of a "receiving" directory.   Complete this *once* for each remote location to which you expected to deploy. It can be created from the shipper or receiver side as long as the operator knows the appropriate information need to create the desired **location.hostmap** file for the receiving location.

The resulting name of the hostmap file is {location_or_environment_name}.hostmap (e.g. atlanta.hostmap, QAT.hostmap) must reside in the **{iDeployIt_home}**\deploy_bay\config\hostmaps directory. Once properly populated, this is the communication key that iDeployIt will use to communicate with the remote location so a user will only need the location/environment name to create **deployment_keys** for that location.

Enter the name (or IP address) of the "receiver" machine that will run the iDeployIt_receiver program then enter a carriage return to accept the input.

Enter the userid that should be used to ftp the encrypted packets to the "receiver" machine then enter a carriage return to accept the input.

Enter the password for the given userid. The password is not echoed to the terminal and must be entered twice. Once entered then enter a carriage return to accept the input.

Enter the **receiver_directory** where the shipper will be allowed to send iDeployIt deployment packets. The directory can be an absolute path from the ftp root or relative to the ftp root. The

provided directory should exist and be writable by the userid. Once entered then enter a carriage return to accept the input:
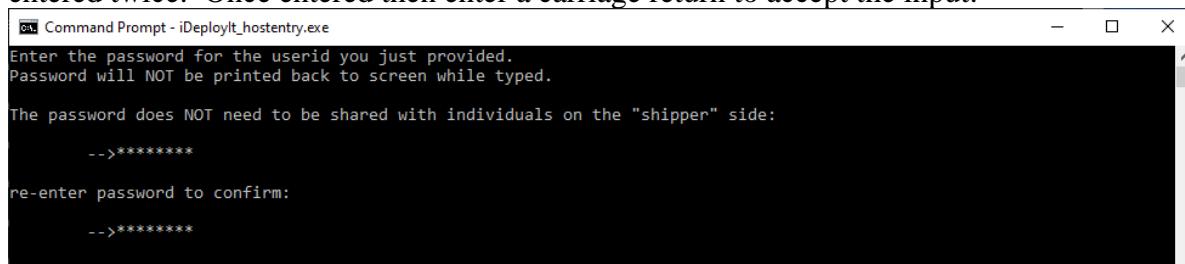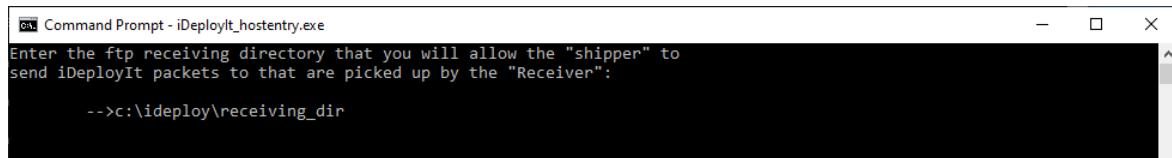
```
Command Prompt - iDeployIt_hostentry.exe                                      —    □    ✕
Enter the ftp receiving directory that you will allow the "shipper" to
send iDeployIt packets to that are picked up by the "Receiver":

       -->c:\ideploy\receiving_dir
```

View the information to confirm its validity .
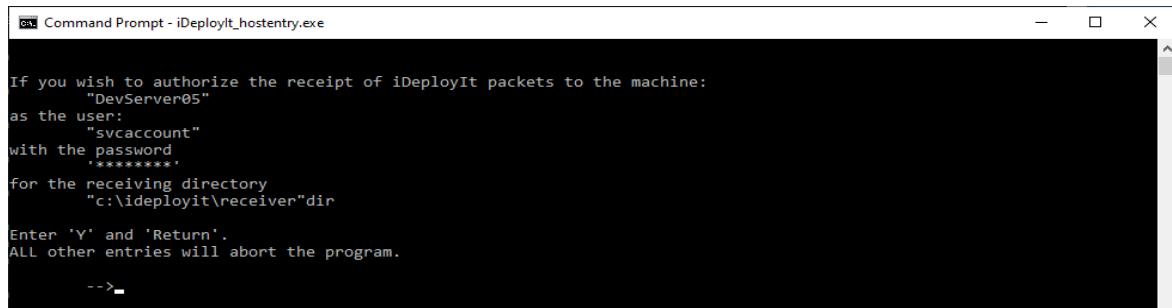
```
Command Prompt - iDeployIt_hostentry.exe                                      —    □    ✕

If you wish to authorize the receipt of iDeployIt packets to the machine:
       "DevServer05"
as the user:
       "svcaccount"
with the password
       '********'
for the receiving directory
       "c:\ideployit\receiver"dir

Enter 'Y' and 'Return'.
ALL other entries will abort the program.

       -->
```

If the information displayed is correct, then enter 'Y' to display the generated **deployment_key**.

Be sure to copy the key as needed before hitting any key to clear the screen and end the program.
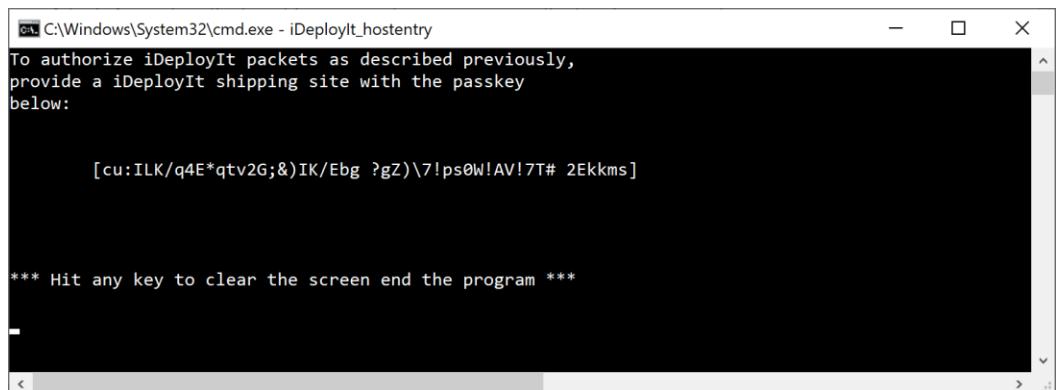
```
C:\Windows\System32\cmd.exe - iDeployIt_hostentry                             —    □    ✕
To authorize iDeployIt packets as described previously,
provide a iDeployIt shipping site with the passkey
below:


       [cu:ILK/q4E*qtv2G;&)IK/Ebg ?gZ)\7!ps0W!AV!7T# 2Ekkms]




*** Hit any key to clear the screen end the program ***
```

Using the _hold_out directory to manually transport packets

The iDeployIt receiver process is concerned with processing the packets that reside in the directory. This transparency allows for *infinite manual transport mechanisms* in addition to FTP and SFTP that are supported by automatic transport. When the "hold" option is used within a **deployment_key** the deployment packet created is not immediately sent by the iDeployIt shipper process. This packet is instead held in the _hold_out directory where it is expected to be transported by some other process. The process can be:

- ftp
- sftp
- sneaker-net (disk transfer)
- direct write to receiver
- rcp
- scp
- network transfer
- email
- or any other mechanism

This allow for deployments between machines that do not have a network connection. There are other considerations when choosing to perform a manual transport.

If transporting to a receiving directory with a "currently running" iDeployIt Receiver process it important to make sure to transport all associated CRC files (epoc.*.*.*.crc and epoc.*.*.*.crc_*) prior to their associated pack file (epoc.*.*.*.pack). This is critical because any active iDeployIt receiver process that sees the pack file will immediately process associated CRC files as it assumes them to be complete. The iDeployIt Shipper itself will not transport the pack file until all associated CRC files have been transported.

Receiver Scheduled Inactive Times

The iDeployIt Receiver can be configured to suspend packet processing during requested inactivity times. One can create a **scheduled_inactivity** definition in the receiver.conf file. During defined scheduled inactivity duration, packets are still received from remote locations, but processing of those packets is postponed until the first available time outside of the defined scheduled_inactivity window. Zero-filled military time is used.

If defined, two (2) unique times must be provided.:

```
scheduled_inactivity=HH:MM-HH:MM
```

Where each HH is '00'-'23' and each MM is '00'-'59'. The first HH:MM is the "start of the inactivity time" designator and the second HH:MM is the "end of the inactivity time" designator.

iDeployIt_receiver

The iDeployIt_receiver program processes received packets based on the receiver.conf file configuration settings. Multiple *iDeployIt_receiver* processes can run per depot. Receiver.conf files must be located in the {iDeployIt_depot}\config directory in that depot. The iDeployIt_receiver program can be started manually or within a startup script or cron job (for UNIX) or startup script or scheduled 'AT' job (for Windows). This process can only be started by the userid designated by the receiver_id configuration variable defined in the receiver.conf file.

Starting/Stopping the iDeployIt "Receiver"

The **iDeployIt_receiver** executable is located in {iDeployIt_bin} directory. See the command reference, ideployit_receiver, for command syntax.

When started the iDeployIt_receiver runs in the background (unless the –wait switch is used). The iDeployIt_receiver MUST run as the receiver_id defined in the associated receiver.conf file.

Use the –wait option to start the shipper process in the foreground causing the calling process or program to wait for the shipper to finish.

For UNIX servers, the iDeployIt_receiver should be started from the server startup scripts or equivalent.

For Windows servers, the iDeployIt_*receiver should be started as a scheduled startup task that runs when the system boots.*

Starting multiple iDeployIt_receiver processes

To improve performance or to distribute administration between multiple group members, start multiple iDeployIt_receiver processes by taking advantage of the [which] parameter passed to

33

iDeployIt_receiver. These processes can be started from different machines to further improve performance. When using multiple receiver processes and therefore multiple receiver.conf files there are some additional rules to consider:

- A single receiver.conf file can reference *multiple* receiver directories, but each receiver directory should *only be referred to by one receiver.conf* file.

- iDeployIt receiver processes running for a single iDeployIt depot can be spread over multiple receivers running from different machines to further improve performance.

Using FTP and SFTP as the transport program

In addition to the manual transport, FTP and SFTP can also be used. For automatic transport of created iDeployIt packets iDeployIt uses one (1) of the eight (8) classes of automatic transport program. To select a class, adjust the **transport_program** and **transport_class** configuration items in the appropriate shipper.conf file so that it corresponds to the program you want to use and is available on your system. The transport_program should be found in the iDeployIt_shipper processes $PATH variable or contains the full path name (no spaces allowed) of the transport program. In some cases this may involve renaming or making a copy of the transport program.

Transport Class	Example transport_program entries in a shipper.conf file	Commonly found
0	transport_program=**ftp** transport_program=c:\dir**ftp.exe** transport_program=/usr/bin/**ftp**	Most known ftp programs support this common standard.
1	transport_program=**sftp** transport_program=c:\dir**sftp.exe** transport_program=/usr/bin/**sftp**	Many known generic sftp programs support this common standard.
2	transport_program=**psftpk** transport_program=c:\dir**psftpk.exe** transport_program=/usr/bin/**/psftpk**	"Putty" sftp for Public/Private Keys and other vendor equivalents
3	transport_program=**ssftp** transport_program=c:\dir**ssftp.exe** transport_program=/usr/bin/**ssftp**	"OpenSSH" sftp and other vendor equivalents.
4	transport_program=**psftpp** transport_program=c:\dir**psftpp.exe** transport_program=/usr/bin/**psftpp**	"Putty" sftp for Password authentication and other vendor equivalents
5	transport_program=**sftp** transport_program=c:\dir**sftp.exe** transport_program=/usr/bin/**sftp**	"Linux" distributions and other vendor equivalents.
6	transport_program=**sftp** transport_program=c:\dir**sftp.exe** transport_program=/usr/bin/**sftp**	"Windows" distributions and other vendor equivalents
7	transport_program=**sftp** transport_program=c:\dir**sftp.exe** transport_program=/usr/bin/**sftp**	"BSD" sftp and other vendor equivalents.

Transport classes allow for FTP/SFTP vendor specific implementations. The default class, class 0 (standard ftp) is common and works for all standard known FTP programs. The other SFTP classes are defined to determine which Transport Program and thus which Transport Class to use in the appropriate **shipper.conf** file. The differences themselves are in the command line argument supported and the "in-session" commands supported.

Class	Command Line arguments supported	In-session Commands supported
0	ftp -in file	cd, lcd, put, get, del, quit, !, user, ascii, binary
1	sftp -B file user@machine	cd, lcd, put, get, del, quit, !
2[*]	psftpk -b file -i keyfile -be -batch user@machine	cd, lcd, put, get, del, quit, !
3	sftp -B file user@machine	cd, lcd, put, get, rm, quit, lrm
4	psftpp -b file –pw password –be –batch user@machine	cd, lcd, put, get, del, quit, !
5	sftp -f file	cd, lcd, put, get, rm, open, quit, !
6[*]	sftp -b file -i keyfile user@machine	cd, lcd, put, get, rm, quit, !
7[*]	sftp -b file user@machine	cd, lcd, put, get, rm, quit, !

Note [*] Class 2, 6 and 7 use a public/private key pair to authenticate. For Class 2 and 6, the keyfile must exists in the {iDeployIt_depot}/config/hostmaps directory and be named {location}.pk where location is the receiving destination pneumonic defined in the associated **deployment_key**. For Class 7 the key pair must be located where the sftp software natively demands.

Triggers

The scripts or programs called by triggers should not require user interaction. Scripts that cause dialogs to appear and wait for user input will cause delay of the deployment process. All undirected writes to stdout and stderr are redirected to /dev/null (UNIX) or NUL (Windows), however, redirect them to files from within the scripts (including the standard iDeployIt logs) by using the **iDeployIt_echo** or **iDeployIt_print** executables in your triggers, build and deployment scripts. The 3 types of triggers that can be created within the iDeployIt processes are:

Trigger Type	Trigger Description
Process Startup/Shutdown	Fire at startup/shutdown attempts of the *shipper*, *receiver* and *monitor* services.
Local Deployment	Locally defined **location-specific** triggers that fire when packets are processed at a location.
Remote Deployment	Sender defined *key-specific* triggers that fire when packets are processed at the destination.

Creating iDeployIt Startup/Shutdown Triggers

Startup/Shutdown Triggers Definitions. Optional Start/Shutdown Triggers Definitions can be added to automatically run scripts after the successful startup or shutdown of shipper or receiver processes. Run an in-house script to perform custom logging, send email notification, install/execute deployed files, start/stop services or even redirect packets by just adding an iDeployIt Startup or Shutdown Trigger.

There is potential to create a **startup_trigger** or **shutdown_trigger** for the *shipper* process, *receiver* process or *both*. Shipper startup or shutdown triggers are defined in **shipper.conf** files while receiver startup or shutdown triggers are defined in **receiver.conf** files.

The scripts can be uniquely named to separate the logic in unique files or be a single script where the logic is determined by using the additional iDeployIt **Start/Stop Environmental Variables** defined by the shipper ad receiver processes and available within trigger scripts.

To define a startup_trigger or shutdown_trigger modify the appropriate line in the shipper configuration or receiver configuration files.

```
(Windows example)
    startup_trigger=some_startup.bat
    shutdown_trigger=some_shutdown.bat

(UNIX example)
    startup_trigger=some_startup.sh
    shutdown_trigger=some_shutdown.sh
```

The actual script or program if defined *must* reside in the **{iDeployIt_depot}/local_actions** directory.

Trigger Definition (optional)	Defined for	Details (**Located in** shipper.conf **and/or** receiver.conf)
startup_trigger	shipper	Optional script or executable that is called immediately after the successful startup of an iDeployIt shipper process.
	receiver	Optional script or executable that is called immediately after the successful startup of an iDeployIt receiver process.
shutdown_trigger	shipper	Optional script or executable that is called immediately after the successful shutdown of an iDeployIt shipper process
	receiver	Optional script or executable that is called immediately after the successful shutdown of an iDeployIt receiver process

Creating Deployment Triggers

The iDeployIt deployment process can be further automated and augmented by the addition of Deployment Triggers Definitions. Optional Deployment Triggers Definitions can be added to automatically run scripts before or after either sending or receiving "Deployment" packets. Use in-house script to perform custom logging, send email notification, install/execute deployed files or start/stop applications services. There are two (2) main types of deployment triggers, **Locally Defined Deployment Triggers** and **Remotely Defined Deployment Triggers**.

Locally Defined Deployment Triggers are defined at the shipper and run at the shipper or are defined at the receiver and run at the receiver.

Remotely Defined Deployment Triggers are defined at the *shipper*, but are run at the *receiver* if allowed by the receiver.
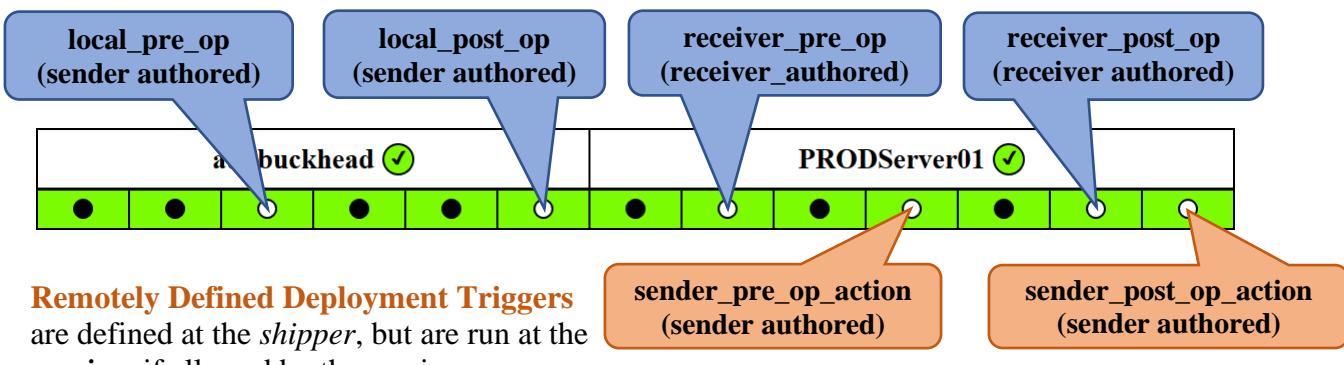
Locally Defined Deployment Triggers

Locally Defined Deployment Triggers are defined at the shipper and run at the shipper or are defined at the receiver and run at the receiver. These triggers are run between predefined deployment steps and allow for the customization of your deployment depending on actions and states defined in those steps.
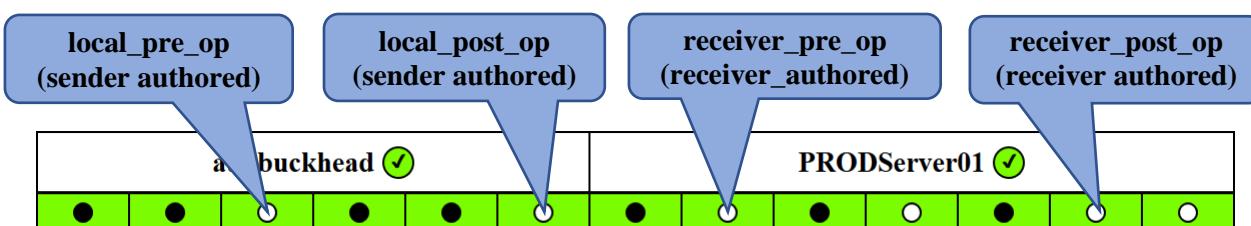
Run in-house scripts to perform custom logging, send email notification, install/execute deployed files, start/stop services, etc. by adding an iDeployIt Local Shipper or receiver Trigger.

There is potential to create optional **pre** or **post** triggers for the *shipper* process, *receiver* process or *both*. These triggers are defined in **shipper.conf** or **receiver.conf** files.

One can also create a **pause** trigger for the *shipper* processes. Pause triggers are defined in **shipper.conf** files.

> Note: A Pause Trigger is a special type of trigger that can be defined when a Deployment has a programmed pause defined within its Deployment Definition. For more on pause triggers review the section entitled **Pausing Deployments**.

The scripts can be uniquely named to separate the logic in unique files or be a single script where the logic is determined by using the additional iDeployIt **Deployment Trigger Environmental Variables** defined within the triggers execution and available within trigger scripts.

To define a local *pre* or *post* or *pause* deploy shipper trigger modify the appropriate lines in the shipper configuration file.

    (windows example)

```
pre_deploy_trigger=some_pre_deploy.bat
post_deploy_trigger=some_post_deploy.bat
pause_deploy_trigger=some_pause_deploy.bat
```

    (UNIX example)

```
pre_deploy_trigger=some_pre_deploy.sh
post_deploy_trigger=some_post_deploy.sh
pause_deploy_trigger=some_pause_deploy.sh
```

To define a local *pre* or *post* deploy receiver trigger modify the appropriate lines in the shipper configuration file.

    (windows example)

```
pre_receive_trigger=some_pre_receive.bat
post_receive_trigger=some_post_receive.bat
```

    (UNIX example)

```
pre_deploy_trigger=some_pre_deploy.sh
post_deploy_trigger=some_post_deploy.sh
```

The script or program if defined *must* reside in the **{iDeployIt_depot}/local_actions** directory.

Optional Deployment Trigger scripts are defined in the associated configuration file as described below:

| Defined in config for	Trigger Definition (optional)	Details
shipper	pre_deploy_trigger	Optional script or executable that is called immediately before any requested shipping packet is created.
	post_**deploy**_trigger	Optional script or executable that is called immediately after any requested shipping packet is created and shipped (or created and placed in the _hold_out directory).
	pause_deploy_trigger	Optional script or executable that is called immediately after the package is built, but before any requested shipping packet is sent.
receiver	pre_receive_trigger	Optional script or executable that is called immediately before any received packet is processed.
	post_receive_trigger	Optional script or executable that is called immediately after any received packet is processed, even if the processed packet did not complete (providing an opportunity to clean up any data changes made in the pre_receive_trigger_script).

Remotely Defined Deployment Triggers

Remotely Defined Deployment Triggers are defined at the shipper, but run at the receiver. The contents of the trigger scripts are packaged with the deployment at the shipper and sent to the receiver as part of the deployment to be executed at the receiver (when access is permitted by the receiver). The **deployment_key** specific triggers if defined, run between predefined deployment steps and allow for the customization of the deployment depending on actions and states defined in those steps.

atl_buckhead ✓   **PRODServer01** ✓

sender_pre_op_action
(sender authored)

sender_post_op_action
(sender authored)
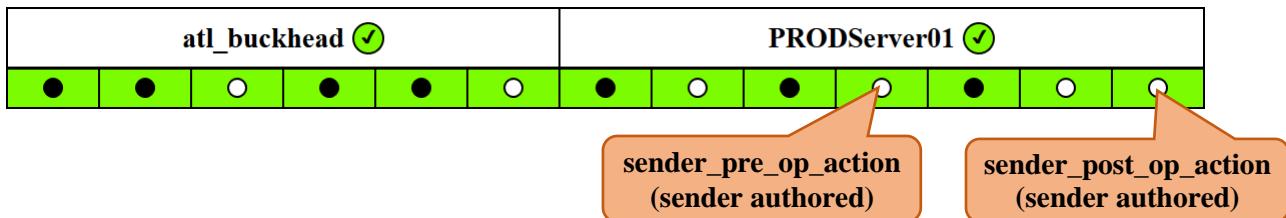
Additional **pre** and **post** triggers scripts are sent and executed at the receiver which reside in the **{iDeployIt_depot}/remote_actions** directory, these triggers are not configured in the shipper or receiver config.

The name of the file that holds the sender_**pre**_op_action code is **###.pre.txt** where ### is the zero-filled key number (i.e. "100.pre.txt" would be the pre-op trigger code to send for *deployment_key* definition 100).

The name of the file that holds the sender_**post**_op_action code is **###.post.txt** where ### is the zero-filled key number (i.e. "100.post.txt" would be the post-op trigger code to send for *deployment_key* definition 100).

📁 remote_actions
📄 003.post.txt
📄 003.pre.txt
📄 100.post.txt
📄 100.pre.txt
📄 101.pre.txt
📄 102.pre.txt
📄 103.pre.txt
📄 410.pre.txt

Any known scripting language can be placed in the contents, By default the contents are interpreted as the natural language of the *receiving destination*, so it would be interpreted as batch code if the destination is a windows machine or the prevailing shell code if the destination is a UNIX machine. If you wish it to be interpreted as some *other* language, modify the *first line* in the file as shown

      `#!{interpreter}`

and that interpreter is used to interpret the code in the rest of the file. See language examples below:

No interpreter provided, so batch code is naturally assumed for a Windows

| Perl (UNIX)	Bourne shell (UNIX)	Batch (Windows)
#!/bin/perl System ("iDeployIt_echo \"in trigger\""); sleep (1);	#!/bin/sh iDeployIt_echo "in trigger" sleep 1 exit 0	iDeployIt_echo in trigger timeout 1 > NUL exit 0
Perl (Windows)	**Korn shell (UNIX)**	**VBscript (Windows)**
#!c:\cots\perl System ("iDeployIt_echo \"in trigger\""); sleep (1); exit (0);	#!/bin/ksh iDeployIt_echo "in trigger" sleep 1 exit 0	#!cscript.exe Set WshShell = WScript.CreateObject("WScript.Shell") WshShell.Run ="iDeployIt_echo ""in trigger""", 0, true WScript.sleep 1000 WScript.Quit 0

Refer to the section entitled **Deployment Trigger Environmental Variables** to view the variables available when *Remotely Defined Deployment Triggers* are executed.

Trigger Return Codes

Triggers can change the flow of your deployment, identify warning states or a terminal error state to stop the deployment altogether.

Trigger return codes indicate the following:

- the return code of 0 is reserved for success
- return codes from 1-100 are reserved for terminal errors (those intended to stop the deployment)
- return codes from 101-256 are reserved for warnings (those intended to mark the deployment step with a warning, but continue the deployment)
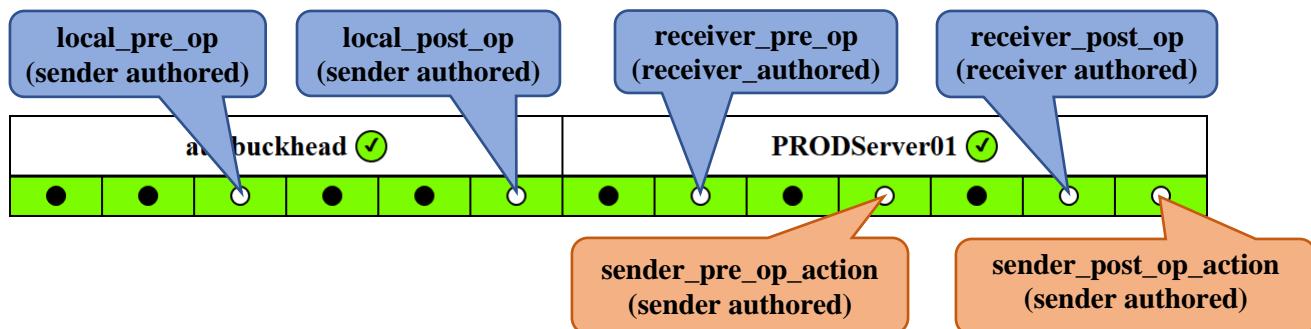
Trigger return codes = 0 are successful and the deployment will continue. As depicted below:
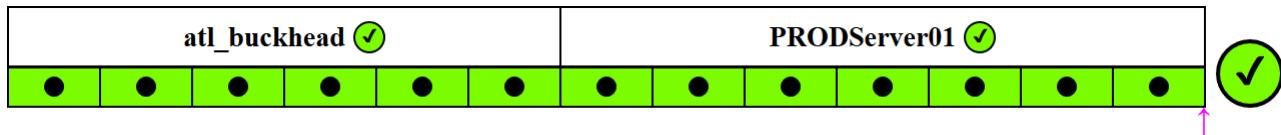
Trigger return codes between 101-256 are considered a warning, the deployment continues. but the *step*, *deployment_location* and *deployment* are marked in yellow (*warning state*). Warning details could be written to the **Deployment_Log_Table**.

Trigger return codes between 1-100 are considered a terminal error, the deployment skips further *predefined deployment steps* for that location and marks the *step*, *deployment_location* and *deployment* in red (*error state*). The deployment completed in the error state; there is one exception to this rule discussed later.
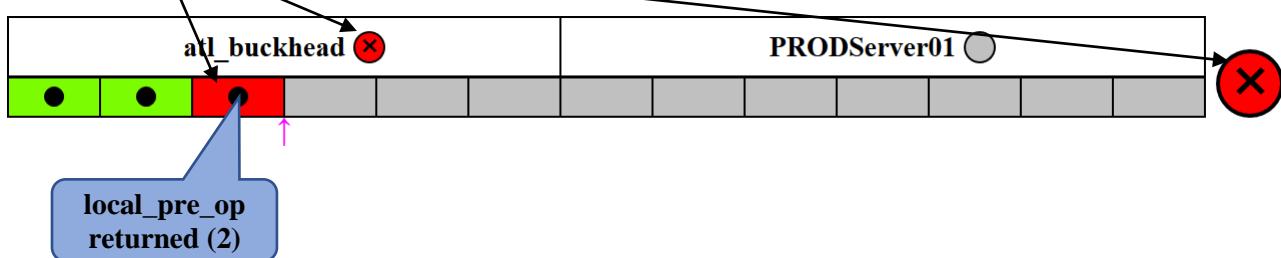
Returning a terminal error code from the **local_post_op trigger** is shown as a warning if sending or copying of the packet to the destination was successful. This is the exception to the terminal error rule as the local_post_op trigger, if defined, fires after the packet was sent to the destination. A failure of the trigger will not stop nor effect the processing at the remote site. In this case any non-success return code is treated as warning, but the step and location and deployment state are updated to reflect as shown below:
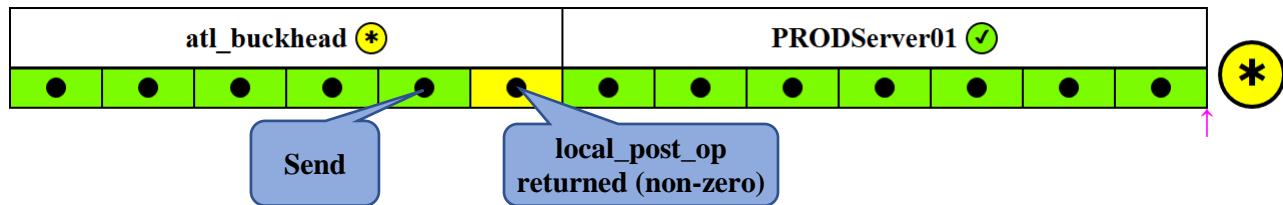

Returning a terminal error code from the **receiver_pre_op trigger** stops the deploy at that step and marks the deployment as incomplete in the error state as shown below:
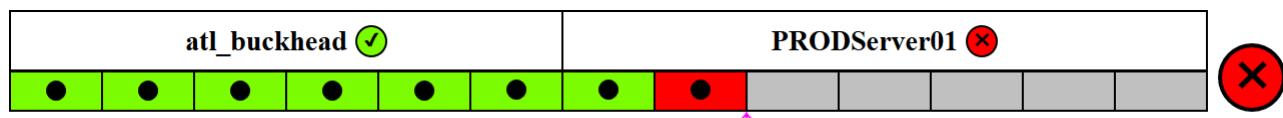
Returning a terminal error code from the **sender_pre_op_action trigger** skips the mandatory deploy step (the *placement* step) and proceeds to a post operation triggers if any, as those could be responsible for any clean-up or email notification. It would also mark that step, location and deployment as completed in the *error state* as depicted below:
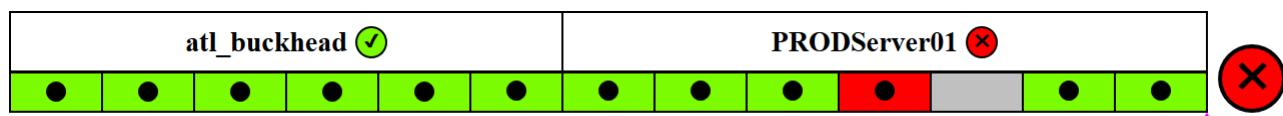

Returning a terminal error code from the **receiver_post_op trigger** or **sender_pre_op_action triggers** will mark the step, location and deployment as completed in the error state as shown below, both triggers if defined will fire as these are post-op triggers and potentially where deployment clean-up and emails might be sent.
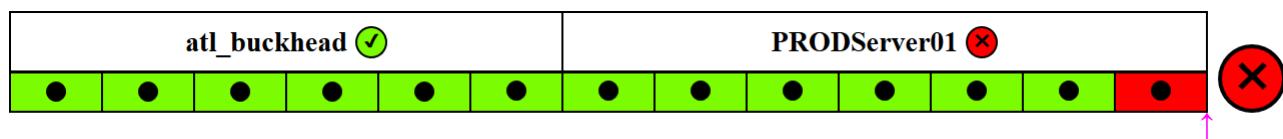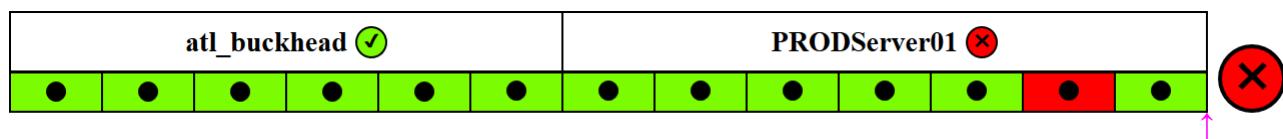
The Creating a new Deployment End-to-End Steps

The deployment is broken into 13 distinct tasks with six (6) tasks responsible for packaging and sending a deployment to a location; seven (7) tasks to unpackage and install the deployment at that location. All steps are predefined when you create a **deployment_key_definition**, the administrator determines if steps should be skipped or if customizations should be added.
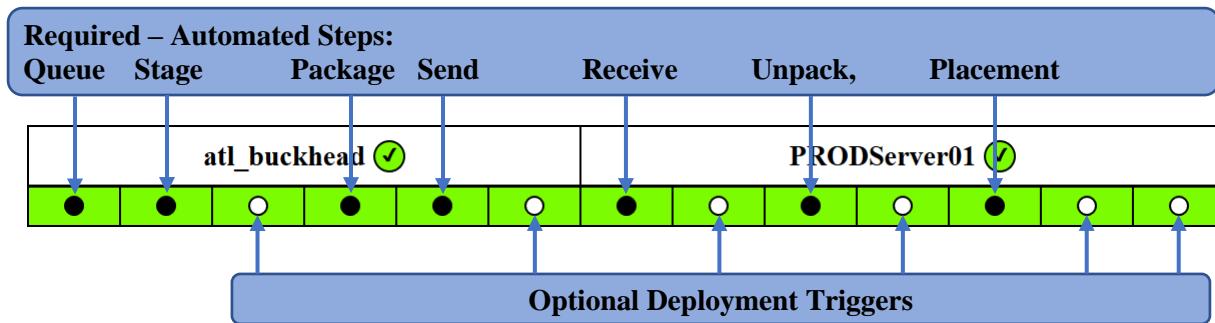
With the core steps of a deployment being automatically created, most deployments are created within 2-4 steps:

1) **Prep** the **receiver location**, this is only necessary for a brand new location. When deploying to a known location, proceed to step #2.

   a. Install and configure the **iDeployIt_receiver** software at the remote location.
   b. Create a **hostmap file** for that location at the sending location.
   c. Test communication to the location with the **ideployIt_shiptest** executable.

2) **Create** a **Deployment_Key_Definition** in your **shipper.conf** file.
3) **Optionally create** **Deployment Triggers.**
4) **Run iDeployIt_deploy** to start a deployment.

Miscellaneous GUI Components, Tricks and Shortcuts

This section discusses GUI functionality that applies to many GUI pages or Dialogs and were best discussed once rather than in each section. It also contains information that might have been a distraction while initially learning the GUI components or GUI information that just did not easily fit anywhere else.

Shortcut Pulldowns

The **Bill of Materials Dialog** or the **Time Usage Graph Dialog** can be opened a separate window by clicking on the appropriate icon from the **Deployment Details View Dialog**.

Selecting the **Bill of Materials Icon** opens the **Bill of Materials Dialog** to view details about the file contents of the deployment.

Selecting the **Time Usage Graph Icon** opens the **Time Usage Graph Dialog** to view time allocation details of each of the deployment steps.

These dialogs can also be opened directly from the **Deployment History View Page**. Perform a slow-click depress the mouse for **½ second** without moving or releasing it to display the pulldown menu and then release it once the pulldown menu is displayed. The three (3) links are always shown, but if the associated dialog does not exist then that link is de-sensitized.

Drag & Drop Instead of Multiple Dialogs

The **Bill of Materials Dialog** or the **Time Usage Graph Dialog** can be opened a separate window by clicking on the appropriate icon from the **Deployment Details View Dialog**.

Selecting the **Bill of Materials Icon** opens the **Bill of Materials Dialog** to view details about the file contents of the deployment.

Selecting the **Time Usage Graph Icon** opens the **Time Usage Graph Dialog** to view time allocation details of each of the deployment steps.

To view the associated dialog in the right-hand side of the *same* window, click down on the mouse while over the selected icon, then before you release the mouse – drag it to the right until just right of the icon, then release the mouse.
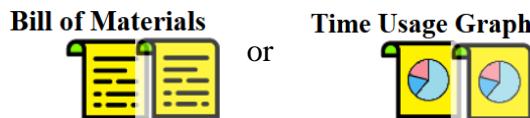
This will open the requested information in the same window as depicted below:

To clear the window, drag the icon to the left and then release it.
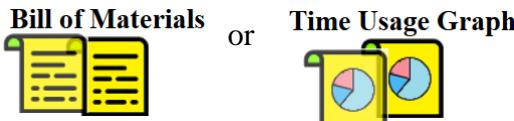
About Dialogs

Display version and build information on **iDeployIt** by clicking the **iDeployIt Logo** at the bottom of most pages and dialogs.
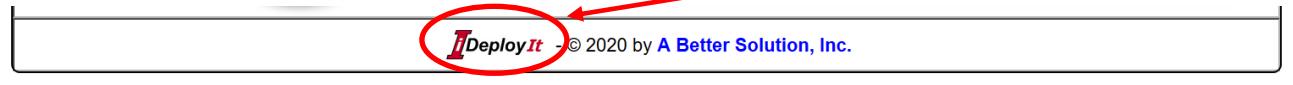
Clicking the logo will open the **iDeployIt About Dialog** which displays the product *Version*, *Build Number*, *Current License Mode* and a support email link.
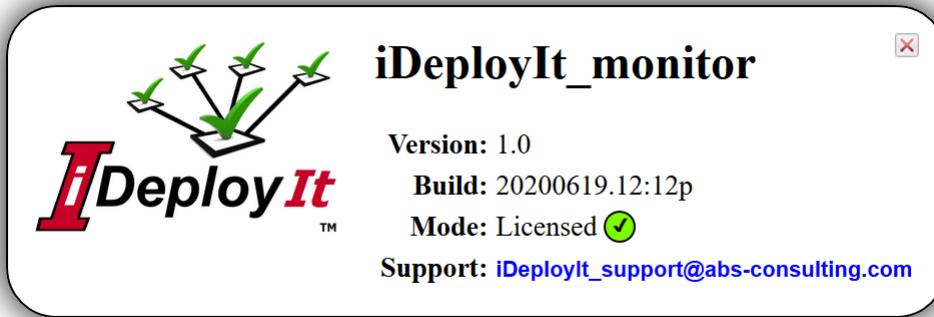
Advanced Topics

The topics are mostly stand-alone Unlock some of the more powerful features of iDeployIt.

Testing Communication with a new Location

The **iDeployIt_shiptest** program sends a test packet to test communication between iDeployIt remote locations.

Purpose:
- determine if the hostmap is properly defined and configured for communication. The sender creates and sends a test packet to the destination location to test validity of a **shipper.conf** file and the associated **location.hostmap** file for the location_name.
- determine the true bandwidth expected between iDeployIt Locations. The command returns the actual bandwidth experience when sending test packet of the requested size.

USAGE: iDeployIt_**shiptest -ver** | **-help** | [iDeployIt_**depot remote_location [class=0..9] [meg=1..100])]**

Such that:

-ver   - produces iDeployIt_shiptest **product version message**

-help   - produces iDeployIt_shiptest **USAGE message**

iDeployIt_**depot -** The **full path** to the associated iDeployIt Depot. This can be a network (UNIX) or UNC (Windows) path, but the local path is preferred if the process is to run on the same machine that holds the iDeployIt Depot.

remote_location - A required and valid location_name that is associated with the **hostmap** file of the name **<location>.hostmap** where <location> is the name given to the "receiver" location by the "shipper" location (i.e. "atlanta", or "london"). There is a 1-to-1 correspondence between the **location_name** (e.g. "atlanta") provided and a file named "**atlanta.hostmap**" in the **{iDeployIt_depot}\config\hostmaps** directory.

 [class=0..9]   - Optional: use a shipper.conf file other than the default "shipper.conf" in "**{iDeployIt_depot}**\config" directory in the associated depot. When used the shipper_**[class].**conf file is used. If class is "2" then "shipper_**2**.conf" will be used. Values 0-9 are valid. When omitted the default "classless" shipper.conf file is used.

 [meg=1..100]   - Optional: The size (in Meg) of the test packet to send to the remote location. If omitted the default of meg=10 is used

The program checks the appropriate **shipper.conf** files (as per the exclusion or use of the "**class=**" values passed in) in the associated depot and tests the validly of the runtime variables within the **shipper.conf** file. If the associated **shipper.conf** file is valid it uses those parameters and the encrypted values within the associated **hostmap file** (**location_host_entry**) to send a test packet to the remote location. It then displays any errors encountered if unsuccessful or the bandwidth experienced if successful.

46

To test an iDeployIt environment between Atlanta and PRODServer01 one would use the command below:

Which test the associated default **shipper.conf** file.
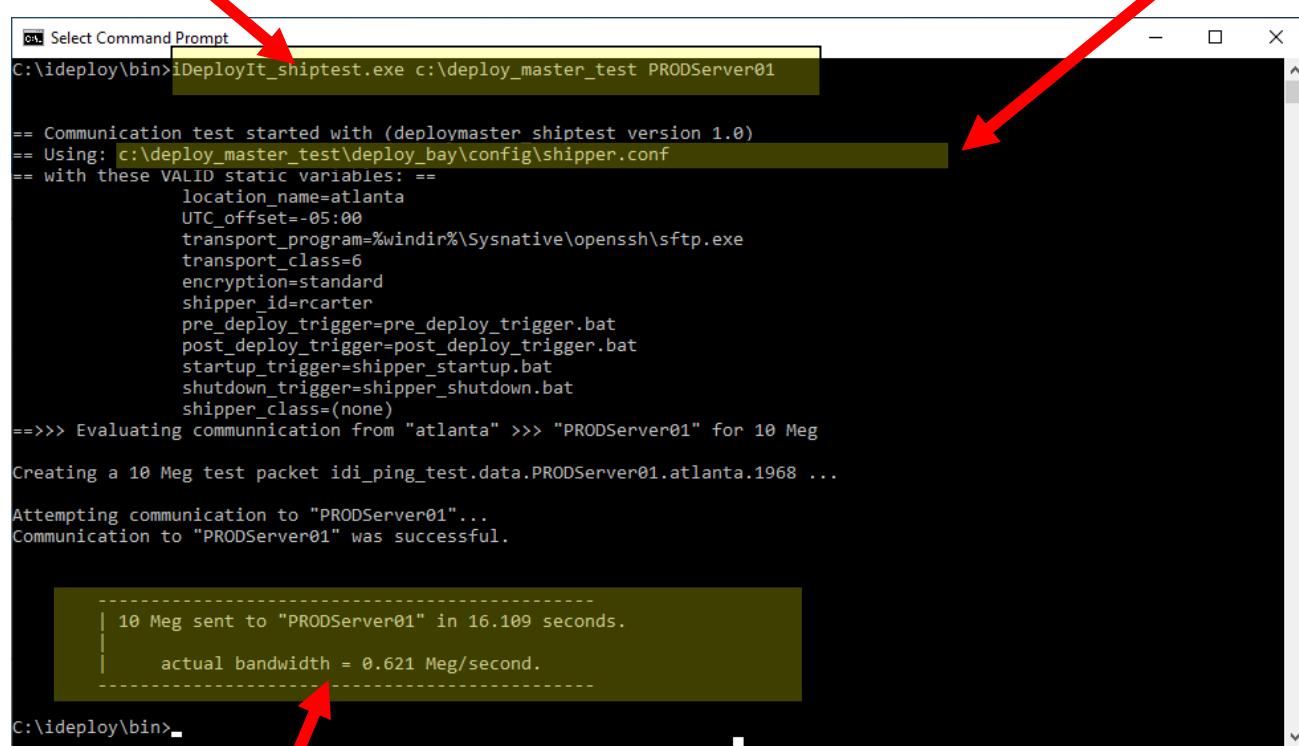
```
Select Command Prompt                                                    —  □  ×

C:\ideploy\bin>iDeployIt_shiptest.exe c:\deploy_master_test PRODServer01


== Communication test started with (deploymaster shiptest version 1.0)
== Using: c:\deploy_master_test\deploy_bay\config\shipper.conf
== with these VALID static variables: ==
            location_name=atlanta
            UTC_offset=-05:00
            transport_program=%windir%\Sysnative\openssh\sftp.exe
            transport_class=6
            encryption=standard
            shipper_id=rcarter
            pre_deploy_trigger=pre_deploy_trigger.bat
            post_deploy_trigger=post_deploy_trigger.bat
            startup_trigger=shipper_startup.bat
            shutdown_trigger=shipper_shutdown.bat
            shipper_class=(none)
==>>> Evaluating communnication from "atlanta" >>> "PRODServer01" for 10 Meg

Creating a 10 Meg test packet idi_ping_test.data.PRODServer01.atlanta.1968 ...

Attempting communication to "PRODServer01"...
Communication to "PRODServer01" was successful.


      -----------------------------------------------
      | 10 Meg sent to "PRODServer01" in 16.109 seconds.
      |
      |    actual bandwidth = 0.621 Meg/second.
      -----------------------------------------------

C:\ideploy\bin>
```

Which results in a **10Meg** (the default) test packet being sent to "PRODServer01" sent in 16.109 seconds for a bandwidth of **0.621 Meg/Second**.

Changing Who Can Use Specific Deployment Keys

By default the user associated with the **shipper_id**, as defined in the **shipper.conf** file, is the only user that can run a deployment using the **iDeployIt_deploy** executable.

The administrator can allow others to perform deployment on a key-specific basis by adding a user to a **deployment_permissions** file. The permissions for the deployment keys are is located in the *{iDeployIt_depot}\config\users* directory. If the associated file for a key does not exist then it is assumed that only the shipper_id can deploy for that key. Create a file for each key in the format, *key#.user.txt*. Inside the file is a list of user ids allowed to deploy that specific key. For Example, given the three (3) files below that exists in the *{iDeployIt_depot}\config\users* directory,

100.user.txt	101.user.txt	102.user.txt
rcarter slewand	rcarter slewand	rcarter

the users *rcarter* and *slewand* are authorized to deploy using **deployment_keys** 100 and 10, while only *rcarter* can deploy using *deployment_key* 102. All other keys could only be deployed by the **shipper_id**.

Using iDeployIt_deploy to Start a Deployment

The **iDeployIt_deploy** program is used to initial a deployment based on a pre-defined **deployment_key_definition** in a **shipper.conf** file. ***Deployment_Keys*** are created by defining and associating a File or Directory on the origination machine, a Destination Location or Environment. Given the example ***Deployment_Key_Definition***.

```
611>c:\WebProduct\dev>/www/WebProduct>WebProduct_p>PRODServer01
```

Once created, an authorized user can deploy the current contents of the c:\WebProducts\dev directory to the PRODServer01 begin by executing the *iDeployIt_deploy* command and refer to this specific deployment_key_definition by its ***key_number*** or it ***product:location*** pair value. The *iDeployIt_deploy* program uses a ***Key_Number Syntax*** and ***Natural Language Syntax*** for requesting deployments.

Specifically, the user could release the ***WebProduct_p*** product (the current contents of the c:\WebProduct\dev directory) to the ***PRODServer01*** server as release ***REL_9.0*** by executing either of the commands below:

```
iDeployIt_deploy {iDeployIt_home} 611 REL_9.0

    or

iDeployIt_deploy {iDeployIt_home} WebProduct_p PRODServer01 REL_9.0
```

Either variant of the command would produce the same result which is to queue the product for deployment as depicted below:
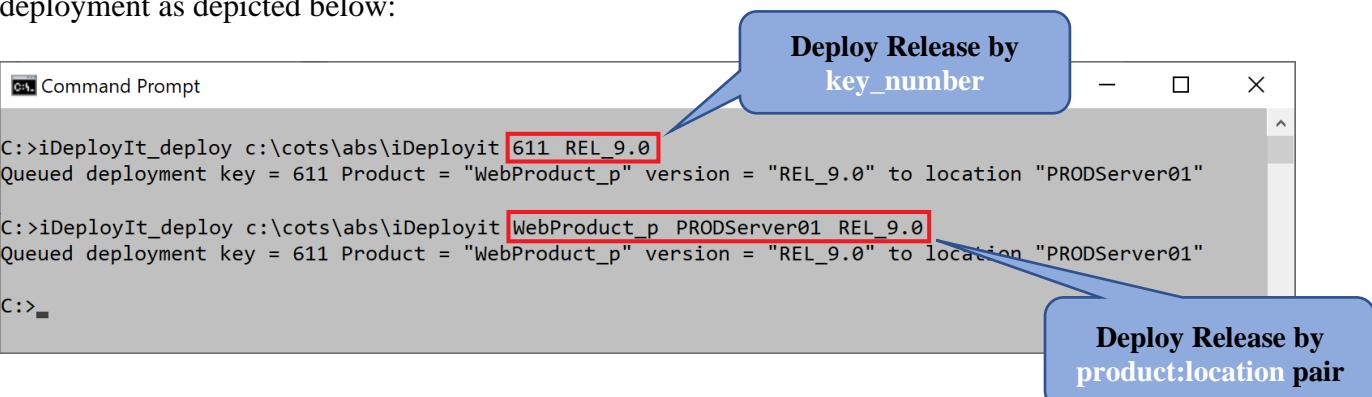
It is anticipated that iDeployIt administrators may group and conceptualize their deployments using the key numbers and invoke deployments using those numbers in topology/policy organization and script automation. Similarly, users that "deploy software" might conceptualize deployments based on product:location pairs.

Both syntax variants are equivalent and provided for convenience.

Adding Pause to Deployments (Pause_Indicators)

iDeployIt provides the ability to pause a deployment after it is packaged, but before an attempt to send it to the destination location to allow someone the opportunity to review the deliverable first for correctness or provide approval for final delivery.

Adding a **pause indicator** to the associated *deployment_definition* (see below):

```
102>c:\ProjectOne>c:\ProjectOne\destination\PROD>ProjectOne>PRODServer01>ship>pause
```

causes the deployment to "pause" during the predefined *Send* step, after the final package is created but before communicating with the remote site allowing for "approval" or "rejection".

Continuing the example above, deploying the *ProjectOne* product to the *godaddy* server as *REL_1.0* via one of the commands below:

```
iDeployIt_deploy c:\cost\abs\iDeployIt 102 REL1.0
            or
iDeployIt_deploy c:\cost\abs\iDeployIt ProjectOne godaddy REL1.0
```

The deployment is started, then paused at the Send step, after the **Bill-of-Materials** is created.
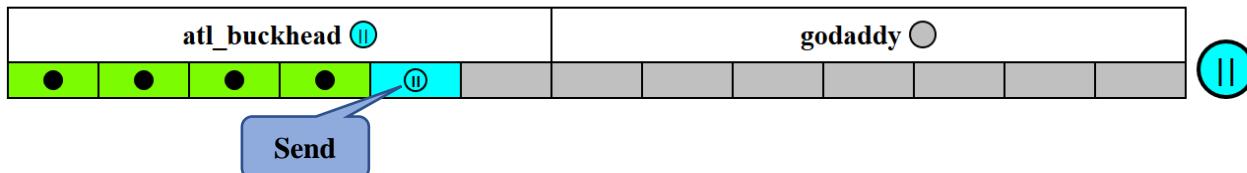
The deployment will stay in this paused state until an authorized user performs one of the commands below to *approve/resume* the deployment.

```
iDeployIt_deploy c:\cost\abs\iDeployIt 102 REL1.0 -resume
            or
iDeployIt_deploy c:\cost\abs\iDeployIt ProjectOne godaddy REL1.0 -resume
```

The user could also *reject/abort* the deployment after the review by entering one of the commands below:

```
iDeployIt_deploy c:\cost\abs\iDeployIt 102 REL1.0 -abort
            or
iDeployIt_deploy c:\cost\abs\iDeployIt ProjectOne godaddy REL1.0 -abort
```

Adding a Pause Deployment Trigger Script

A **Pause Deployment Trigger Script** can be called during a defined deployment pause in order to invoke some customized process. This script can then be called individually for any deployment by adding a **pause_indicator** to the **deployment_key_definition**.

To create a *Deployment Pause Trigger Script*, define it in the associated **shipper.conf** file then create the associated script in the *{iDeployIt_depot}\local_triggers* directory (see below):

    pause_deploy_trigger=some_pause_deploy.bat

Business logic including emails to be sent can be placed in this script. Refer to the section entitled **Deployment Trigger Environmental Variables** to view the variables available when the pause_deploy_trigger program is executed.
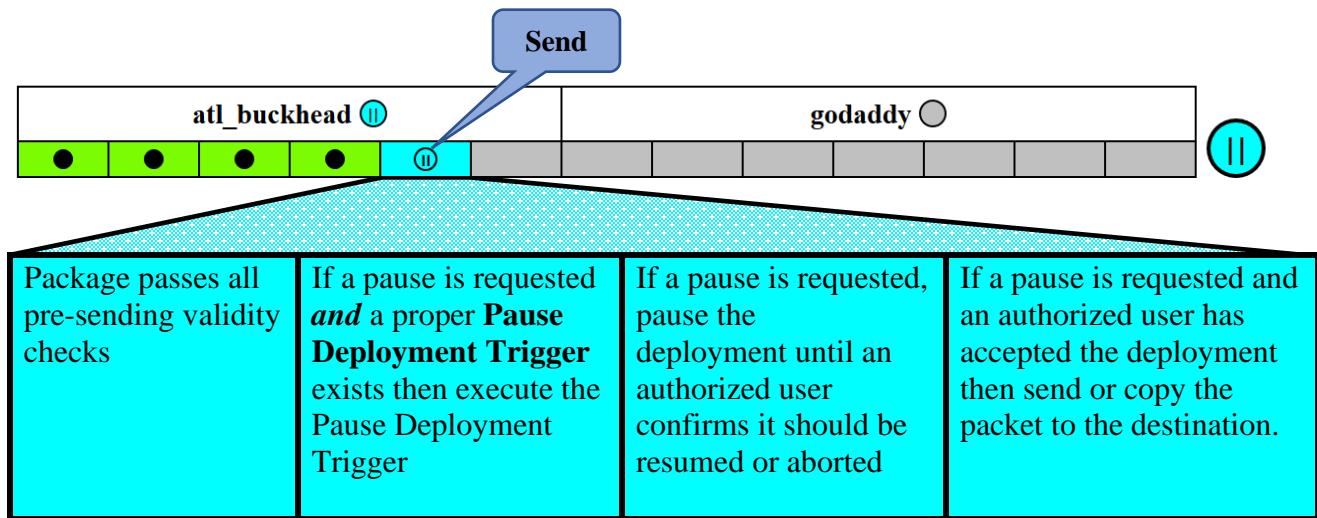

Any non-zero return code in a Pause Deployment Trigger is treated as a warning, the overriding user acceptance or rejection determines if the deployment continues or not.

Pause_Indicators vs. Pause Deployment Trigger Scripts

Pause_indicators and **Pause Deployment Trigger Scripts** are related, but not the same.

A *Pause_indicator* is optionally defined *per* **deployment_key_definition** and will always cause the associated deployment to *pause* and wait for the proper invocation of the matching *iDeployIt_deploy* *-resume* or *-abort* command in order to *release* the pause. A p*ause_indicator* does *NOT* require that a *Pause Deployment Trigger Script* is defined, it only indicates that a *pause* (and subsequent -*resume* or -*abort*) is invoked.

Pause Deployment Trigger Scripts are optionally defined *per* **shipper.conf** file will always cause the associated script to be called during the pause of any deployment associated with a shipper.conf file. Only the shipper.conf files paused deployments call the script which is called immediately before the deployment is paused.

Encrypting Packages with peer_to_peer encryption

The generated key is required by both shipper and receiver to enable data encryption. The key is case sensitive and MUST be entered *exactly* as printed including the surrounding square brackets.

From the **{iDeployIt_bin}** directory, run the **iDeployIt_crypt_key_gen.exe**

Send the contents to a file by redirecting its output and place the file in the **{iDeployIt_depot}**\config\keys directory of the shipper and receiver.
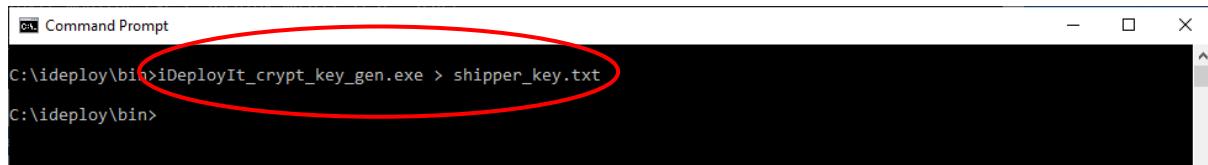
Modify the shipper.conf file located in **{iDeployIt_depot}**\config, remove the "#" from the following line and save the file.

```
#encryption=peer_to_peer
```

If multiple encryption keys are needed for different locations, create a copy of the shipper.conf file and name the next file shipper_1.conf. In the *shipper.conf* file uncomment or add the line

```
#add_class=1
```

Within the *shipper_1.conf* file uncomment or add the line

```
#encryption=peer_to_peer
```

Allows for shipper processing from an additional shipper process. Class 0 to 9 allowing for 11 unique encryption keys per iDeployIt depot with one key for the classless shipper file. (shipper.conf) and 10 keys for the class files (shipper_0.conf – shipper_9.conf).

Creating additional class_definitions

Create the **class_definition** only after creating the associated **shipper.conf** file, as soon as it's create the class_definition in the shipper.conf file is "active" and iDeployIt will expect that the file exists. Creating a class_definition can provide three (3) distinct benefits:

- **Provide improved performance** by allowing multiple shippers to run in parallel on a single machine or different machines.

- **Provide administrative separation** between project groups

- **Provide different transport mechanisms** when different transport mechanisms (FTP/SFTP classes) are required between certain keys or sites

When iDeployIt receives a request it will read the appropriate **shipper.conf** file to determine the deploy key to use. If additional class_definitions are defined the associated shipper.conf file will be read for that class (i.e. "class 4" will cause the interrogation of the "shipper_**4**.conf" file) after reading the original shipper.conf file.

Up to ten shipper classes can be defined (classes 0..9). The class_definition is defined in the original shipper.conf file (the only "class-less" shipper.conf file). The class_definition can exist on a line by itself in the shipper.conf file of the form:

    add_class=#

Where # is ('0' – '9')

```
add_class=0
add_class=1
add_class=9
```

Class_definitions:

- define which additional shipper[class].conf file to interrogate.
- can only exist in the "class-less" shipper.conf file
- cannot be duplicated in the shipper.conf file.

Using Multiple Shipper.conf Files to Provide Better Performance:

Use multiple **shipper.conf** files to work on the same set of **deployment_keys** allowing any class to process the keys. This allows processing of all iDeployIt_shipper process simultaneously allowing for increased packet creation performance.

A site with 90 deployments has deployment_keys related to deployments 1-90 defined in *shipper.conf*, *shipper_1.conf* and *shipper_2.conf*. Processing for these 90 keys is distributed between three (3) shipper classes and processes allowing 3 parallel threads.
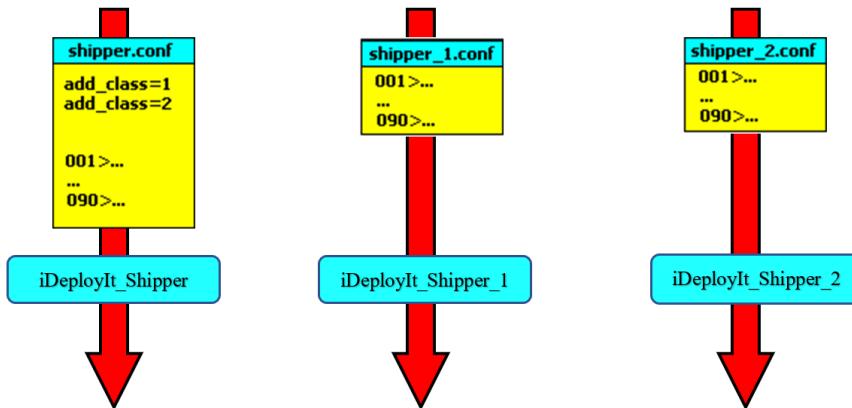
The first process that gets to a deployment packet will process it, while the others immediately process the next available deployment packet.

Note: The different iDeployIt shipper processes can be run from different machines to further take advantage of available CPU and bandwidth.

Using Multiple Shipper.conf Files to Provide Project Separation

Use multiple **shipper.conf** files to allow different administrators to configure and manage their own unique set of **deployment_keys**.

A site with 90 deployments might have all deployment_keys related to deployments 1-30 defined in shipper.conf managed by the user *cclarke*. All deployment_keys related to deployments 31-60 defined in shipper_1.conf managed by the user *rcarter*. All deployment_keys related to deployments 61-90 defined in shipper_2.conf managed by the user *slewand*. Users can run their associated iDeployIt_shipper process at times or with configuration settings that suit them.
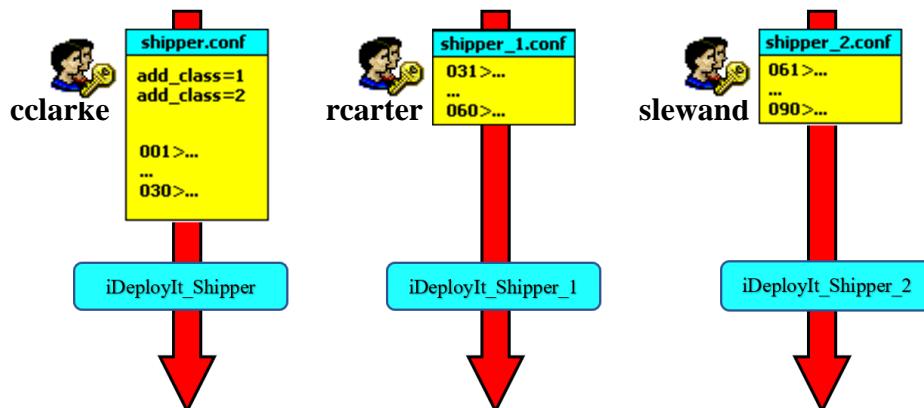
Using Multiple Shipper.conf Files to Facilitate different Transport Mechanisms

Use multiple **shipper.conf** files to allow different transport mechanisms between different sets of locations.

A site with 90 deployments might have all deployment_keys related to deployments 1-30 defined in shipper.conf sent to sites that allow *standard FTP protocol* (class 0). All deployment_keys requiring communications to locations that *require* **SFTP (class 4)** use deployments 31-60 defined in shipper_1.conf. Finally, all deployment_keys related to deployments 61-90 defined in shipper_2.conf *require SFTP (class 6)* to communicate with their destination machines.
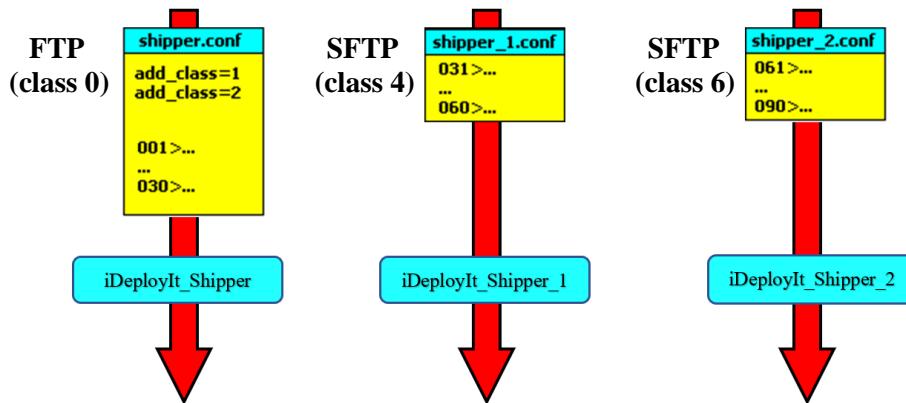
Using shipper.conf include files

An iDeployIt administrator can define several deployments that they place in a series of **shipper.conf** files so that each file has a set of **deployment_keys** without the administrator having to modify or maintain each file. The administrator can place keys in a normal file that can then be included in the shipper.conf files. Multiple include files are allowed in shipper.conf files.

A site with many deployments might have all deployment_keys 1-100 defined in acme.txt and additional deployment keys 101-200 defined in cots.txt. Multiple include files can be included in a shipper.conf file. These files can contain deployment keys and the contained keys are treated as if they were in the *original* shipper.conf file.

So, given the example depicted, the processing demand for the *"acme"* keys (000-100) are spread over the three (3) shipper process that are associated with the files; *shipper.conf, shipper_1.conf* and *shipper_2.conf*.

While the keys in the *cots.txt* file are only processed via the shipper process associated with *shipper_2.conf*.
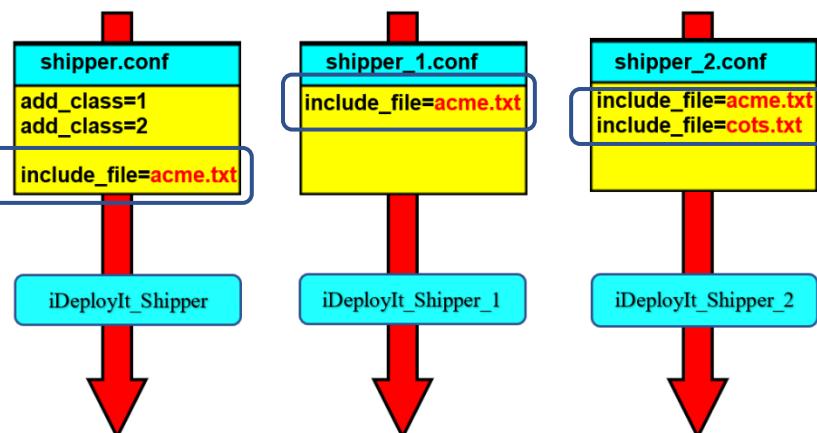
Relocating iDeployIt directories

By default all iDeployIt writes are performed within directories underneath the {iDeployIt_depot} directory. For organizations that wish to run iDeployIt from a read-only share, media or device, iDeployIt allows for *directory relocation*.

The iDeployIt directories (listed relative to the iDeployIt "depot") and the binaries that write to them are listed in the table below:

"depot" relative directory	Binaries that write to this directory			
	iDeployIt **shipper**	iDeployIt **receiver**	iDeployIt **monitor**	iDeployIt **deploy**
{iDeployIt_depot}_runtime	☑	☑	☑	☑
{iDeployIt_depot}_hold_out	☑	☐	☐	☐
{iDeployIt_depot}_staging	☑	☐	☑	☐
{iDeployIt_depot}\logs	☑	☑	☑	☐
{iDeployIt_depot}\web_pages	☐	☐	☑	☐

☑= **Directory is written to by this program**
☐= **Directory is NOT written to by this program**

Create "_iDeployIt_relocated.txt" create a single line that contains the relocated destination directory. Populate the *first* line only, it is evaluated and expected to contain the relocated destination directory, all other lines are treated as a comment and ignored.

When selecting a relocated destination directory consider these imitations:
- Only absolute directory names should be used
- Drive resident names are allowed (Windows)
- UNC directory names are NOT allowed (Windows)
- Network directory names are allowed (UNIX)

Examples of valid content for a _iDeployIt_**relocated.txt** are below:

Receiver Log file	Containing entries in chronological order
Windows local	C:\some_other\logs
Windows mapped drive	J:\some_other\mapped_shipout
UNIX	/some_other/directory/logs
UNIX network path	/net/machine/some_other_dir/logs

Deployment Definition Examples

This section provides examples of deployment definitions. Syntax is not discussed in this section. Example deployment scenarios are described and a **deployment_key** that satisfies that scenario is provided.

Deployment to Multiple Locations with One Key

Define multiple destination locations for a deployment in one **deployment_key_definition**. In the **shipper.conf** file use the following key:

```
200>c:\ProjectOne>c:\ProjectOne\dev>ProjectOne>DevServer01 DevServer02>ship
```

In this example the *deployment_key* 200 defines a deployment of *ProjectOne* to the locations *DevServer01* **and** *DevServer02*. Deploy *Release_2* of *ProjectOne* to **both** locations by using the **iDeployIt_deploy** variant that uses the *key_number* below:

```
iDeployIt_deploy {path_to_depot} 200 Release_2
```

To deploy to just the DevServer01 location use the **iDeployIt_deploy** variant that uses the *Product:Location* pair like below:

```
iDeployIt_deploy {path_to_depot} ProjectOne DevServer01 Release_2
```

There is no limit to the number of locations that can be defined in a *deployment_key_definition*, but the definition itself is limited to 512 characters.

Programming a Pause in a Deployment

Define a pause for the deployment in the **deployment_key_definition**. In the **shipper.conf** file we have the following key:

```
300>c:\ProjectOne>c:\ProjectOne\dev>ProjectOne>godaddy>ship>pause
```

Deployment of the *ProjectOne* product to the *godaddy* environment/location :

```
iDeployIt_deploy c:\cost\abs\iDeployIt 300 REL1.0
      or
iDeployIt_deploy c:\cost\abs\iDeployIt ProjectOne godaddy REL1.0
```

The deployment starts, but will pause during the Send step, after the **Bill-of-Materials** is created and before the package is sent to the destination environment or location.
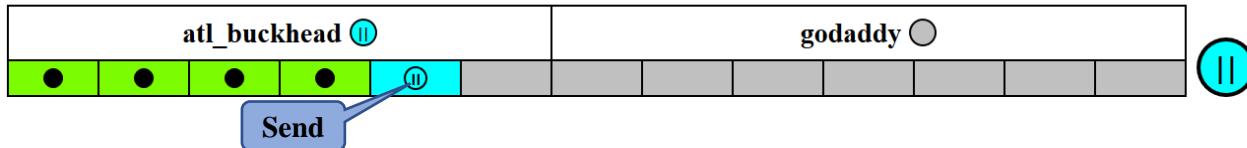
The deployment will stay in this paused state until an authorized user performs one of the following commands below to *approve/resume* the deployment.

```
iDeployIt_deploy c:\cost\abs\iDeployIt 300 REL1.0 -resume
      or
iDeployIt_deploy c:\cost\abs\iDeployIt ProjectOne godaddy REL1.0 -resume
```

iDeployIt Log Files

The process produces an extensive set of log files for both the "shipper" and "receiver" processes. Shipper log files are located in the **{iDeployIt_depot}\logs** directory.

"Shipper" Logs

Shipper Log file	Containing entries in chronological order
shipper_all.txt	All shipper log messages (except deployment details and TOC details)
shipper_deploys.txt	All shipper *deployment details* (version list of files contained in "deployment "Packets.
shipper_errors.txt	All shipper *error* messages
shipper_start_stop.txt	All shipper *start* or *stop* messages
shipper_TOCs.txt	All shipper *TOC details* (version list of files contained in "deployment "Packets.
shipper_warnings.txt	All shipper *warning* messages

Running the iDeployIt_**shipper** against these files produces…		
shipper.conf	**shipper_0.conf**	**shipper_1.conf**
shipper_all.txt	shipper_all_**0**.txt	shipper_all_**1**.txt
shipper_deploys.txt	shipper_deploys_**0**.txt	shipper_deploys_**1**.txt
shipper_errors.txt	shipper_errors_**0**.txt	shipper_errors_**1**.txt
shipper_start_stop.txt	shipper_start_stop_**0**.txt	shipper_start_stop_**1**.txt
shipper_TOCs.txt	shipper_TOCs_**0**.txt	shipper_TOCs_**1**.txt
shipper_warnings.txt	shipper_warnings_**0**.txt	shipper_warnings_**1**.txt

"Receiver" Logs

Receiver Log file	Containing entries in chronological order
receiver_all.txt	All receiver log messages (except deployment details and TOC details)
receiver_deploys.txt	All receiver *deployment details*
receiver_errors.txt	All receiver *error* messages
receiver_start_stop.txt	All receiver *start* or *stop* messages
receiver_TOCs.txt	All receiver *TOC details* (version list of files contained in "deployment "Packets.
receiver_warnings.txt	All iDeployIt receiver *warning* messages

Running the iDeployIt_**receiver** against these files produces…		
receiver.conf	**receiver_atl.conf**	**receiver_london.conf**
receiver_all.txt	receiver_all_**atl**.txt	receiver_all_**london**.txt
receiver_deploys.txt	receiver_deploys_**atl**.txt	receiver_deploys_**london**.txt
receiver_errors.txt	receiver_errors_**atl**.txt	receiver_errors_**london**.txt
receiver_start_stop.txt	receiver_start_stop_**atl**.txt	receiver_start_stop_**london**.txt
receiver_TOCs.txt	receiver_TOCs_**atl**.txt	receiver_TOCs_**london**.txt
receiver_warnings.txt	receiver_warnings_**atl**.txt	receiver_warnings_**london**.txt

Trigger Environment Variable Reference

iDeployIt allows the creation of triggerable actions when **Shippers** and **Receiver** processes are started or stopped allowing the creation of custom actions at these critical event points. Start/stop other services, send email or write additional information to system logs, etc. Create or extend actions to **deployments** and add automation points to deployments at either the origination or destination locations.

Any script or program called at a customization point has access to the system environmental variables as well as numerous iDeployIt created environmental variables. The following two (2) sections list the environmental variables available in each case.

iDeployIt Start/Stop Trigger Environment Variables

This section lists Start/Stop Environmental Variables that are set and available during the execution of Start/Stop Trigger Scripts that are called as a result of one of the starting or stopping of shippers or receivers.

- o **receiver_stop**
- o **receiver_start**
- o **shipper_stop**
- o **shipper_start**

Environmental variable					Meaning/Example
IDEPLOYIT_LOGGING_DIR	X	X	X	X	The current iDeployIt Logging directory
IDEPLOYIT_THIS_LOCATION	X	X	X	X	Associated pneumonic that the shipper or receiver calls itself (i.e. **"atlanta"**)
IDEPLOYIT_RECEIVER_ACTION			X	X	"startup" during **receiver_start** trigger or "shutdown" during **receiver_stop** trigger
IDEPLOYIT_RECEIVER _ENDED_AT				X	Seconds since epoch at the time the receiver ended/stopped
IDEPLOYIT_RECEIVER _VERSION			X	X	The version of iDeployIt_receiver running this process
IDEPLOYIT_RECEIVER _STARTED_AT				X	Seconds since epoch at the time the receiver started
IDEPLOYIT_RECEIVER_WHICH			X	X	The receiving class of the processing receiver or not set for he "classless" (default) receiver.
IDEPLOYIT_SHIPPER_ACTION	X	X			"startup" during **shipper_start** trigger or "shutdown" during **shipper_stop** trigger
IDEPLOYIT_SHIPPER_ENDED_AT		X			Seconds since epoch at the time the shipper ended/stopped
IDEPLOYIT_SHIPPER_VERSION	X	X			The version of iDeployIt_shipper running this process
IDEPLOYIT_SHIPPER_STARTED_AT	X	X			Seconds since epoch at the time the shipper started
IDEPLOYIT_SHIPPING_CLASS	X	X			The shipping class (0..9) of the processing shipper or not set for the "classless" (default) shipper.
IDEPLOYIT_SHIPPER_DEPLOYMENT_USER				X	The User ID that the shipper process is running under.

Whereas	Means
X	Always set for the **replica trigger type.**

iDeployIt Deployment Trigger Environmental Variables

This section lists the iDeployIt Deployment Environmental Variables that are set and available during the execution of iDeployIt Deployment Trigger Scripts that are called as a result of one of the iDeployIt Trigger Types.

- o **sender_post_deploy_trigger**
- o **receiver_post_deploy_trigger**
- o **sender_pre_deploy_trigger**
- o **receiver_pre_deploy_trigger**
- o **post_deploy_trigger**
- o **pause_deploy_trigger**
- o **pre_deploy_trigger**

Environmental variable	sender_post	receiver_post	sender_pre	receiver_pre	post	pause	pre	Meaning/Example
IDEPLOYIT_ALLOW_SENDER_TRIGGERS				X	X	X	X	Determines if the receiving location allows for the sender to run script at the receiver's location. Valid values are "yes" or "no".
IDEPLOYIT_DEPLOYMENT_DIR						X	X	If and only if the processed packet was a "Deployment" packet this has the **destination directory** of the deployed to location.
IDEPLOYIT_DEPLOYMENT_KEY	X	X	X					The **deployment key**.
IDEPLOYIT_KEY_ID	X	X	X	X	X	X	X	Key number associated with the **deployment key** (i.e. **"100"** for the definition "**100**>c:\ideploy\source>c:\ideploy\destination\dev>ProductTest>DevServer01>hold**")
IDEPLOYIT_LOGGING_DIR	X	X	X	X	X	X	X	The current iDeployIt Logging directory
IDEPLOYIT_PACKET_DEPLOYMENT_TO_FROM_OS	X	X	X			X	X	Displays the deployment TO and FROM Operating System Architectures. For example, **Windows** or **Linux**.
IDEPLOYIT_PACKAGE_NAME	X			X	X	X	X	The name of the packet that caused this script to fire (i.e. **"deploy.102.atlanta.london"**)
IDEPLOYIT_PACKET_PRODUCT_NAME	X	X	X					The Product Name from the deployment key that is contained in this packet.
IDEPLOYIT_PACKET_PRODUCT_RELEASE	X	X	X					The Product Release from the deployment key that is contained in this packet.
IDEPLOYIT_PACKET_STATUS	X	X	X	X	X	X	X	Status of the current packet that caused this script to fire. From: "**OK**" – Packet was successfully processed (shipped/received) "**OK with Warnings**" - Packet was successfully processed (shipped/received), but there were warnings written to logs files. "**NOK**" – Packet was NOT successfully processed "**Pending**" – always for pre_ship_trigger
IDEPLOYIT_PACKET_UUID	X	X	X	X	X	X	X	The Packet's UUID or unique identifier.
IDEPLOYIT_PENDING_DIR	X	X	X					The location where the packet is stored waiting to be processed.

Variable								Description
IDEPLOYIT_PRODUCT_NAME				X	X	X	X	The Product Name from the deployment key that is contained in this packet.
IDEPLOYIT_PRODUCT_RELEASE				X	X	X	X	The Product Release from the deployment key that is contained in this packet.
IDEPLOYIT_RECEIVER_VERSION				X	X	X	X	The version of iDeployIt_receiver running this process
IDEPLOYIT_RECEIVING_DIR				X	X	X	X	The **receiver_directory** that received the current packet.
IDEPLOYIT_RECEIVING_LOCATION	X	X	X	X	X	X	X	The Location from the **deployment key** (i.e. **"DevServer01"** for the definition "100>c:\ideploy\source>c:\ideploy\dest\dev>ProductTest>**DevServer01"**
IDEPLOYIT_SHIPPER_DEPLOYMENT_USER				X	X	X	X	The User ID that the shipper process is running under.
IDEPLOYIT_SHIPPER_VERSION	X	X	X	X	X	X	X	The version of iDeployIt_shipper running this process
IDEPLOYIT_SHIPPING_DEPLOY_DIR_OR_FILE	X	X	X					The Source Location from the **deployment key** (i.e. **"c:\deploy\source"** for the definition "100>**c:\deploy\source**>c:\ideploy\destination\dev>ProductTest>DevServer01>hold**"**
IDEPLOYIT_SHIPPING_DEPLOY_TYPE	X	X	X					Denotes the deployment type is **Directory** or **File** deployment.
IDEPLOYIT_SHIPPING_LOCATION	X	X	X	X	X	X	X	Associated pneumonic that the shipper calls itself (i.e. **"atlanta"**)
IDEPLOYIT_THIS_LOCATION	X	X	X	X	X	X	X	The associated pneumonic that the location calls itself (i.e. **"atlanta"**)
IDEPLOYIT_TRIGGER_TYPE	X	X	X	X	X	X	X	Type of **trigger** being executed. From: • "pre_deploy" • "post_deploy" • "pause_deploy" • "pre_receive" • "post_receive"
IDEPLOYIT_LOCAL_UTC				X	X	X	X	The receiver location's UTC Time zone designator of the form +-HH:MM
IDEPLOYIT_REMOTE_UTC				X	X	X	X	The sender location's UTC Time zone designator of the form +-HH:MM

Whereas	Means
X	Always set for the **deployment trigger type.**

Command Reference

This section list the **Usage Statements** and **Manual Pages** of the programs delivered with iDeployIt.

iDeployit_shipper

USAGE: iDeployIt_**shipper** -ver | -help | **path_to_iDeployIt_depot** (-start | -once | -stop | -restart | -reonce | -queue_stop | -dequeue_stop) [0..9] [-wait]

Such that:

-ver	-	produces iDeployIt_shipper **product version message**
-help	-	produces iDeployIt_shipper **USAGE message**
path_to_iDeployIt_depot	-	The **full local path** to the associated iDeployIt Depot. The iDeployIt_shipper normally runs on the same machine that holds the iDeployIt Depot.
-start	-	**starts the process** and continually runs for **infinite passes** as per the configuration parameters in the shipper.conf file.
-once	-	**starts the process** and runs for **one (1) pass** as per the configuration parameters in the shipper.conf file.
-stop	-	**Stops any** iDeployIt_shipper process for the associated depot
-restart	-	Performs an iDeployIt_shipper –**stop** then a –**start** for the associated depot.
-reonce	-	Performs an iDeployIt_shipper –**stop** then a –**once** for the associated depot.
-queue_stop	-	Attempt the stop of an iDeployIt_shipper process (–**stop**), if the process cannot stop because it is working on a packet the stop is queued to be invoked when the process has completed its current packet.
-dequeue_stop	-	Remove a "queued" stop for a iDeployIt_shipper process
[0..9]	-	use a shipper.conf file other than the default in "deploy_bay/config" directory in the associated depot. When used the shipper_**[class].**conf file is used. If class is "2" then "shipper_**2**.conf" will be used. Values 0-9 are valid.
-wait	-	Start the shipper process in the foreground, the calling process will wait for the shipper process to complete. The default shipper runs in the background and immediately returns control to the calling process. Useful for the monitoring of shipper runs as a scheduled task.

iDeployit_receiver

USAGE: iDeployIt_**receiver -ver | -help | path_to_iDeployIt_depot** (-start | -once | -stop | -restart | -reonce | -queue_stop | -dequeue_stop) [which] [-wait]

Such that:

-ver	-	produces iDeployIt_receiver **product version message**
-help	-	produces iDeployIt_receiver **USAGE message**
path_to_iDeployIt_depot	-	The **full local path** to the associated iDeployIt Depot. The iDeployIt_receiver runs on the same machine that holds the iDeployIt Depot.
-start	-	**starts the process** and continually runs for **infinite passes** as per the configuration parameters in the receiver.conf file.
-once	-	**starts the process** and runs for **one (1) pass** as per the configuration parameters in the receiver.conf file.
-stop	-	**Stops any** iDeployIt_receiver process for the associated depot
-restart	-	Performs an iDeployIt_receiver –**stop** then a –**start** for the associated depot.
-reonce	-	Performs an iDeployIt_receiver –**stop** then a –**once** for the associated depot.
-queue_stop	-	Attempt the stop of an iDeployIt_receiver process (–**stop**), if the process cannot stop because it is working on a packet the stop is queued to be invoked when the process has completed its current packet.
-dequeue_stop	-	Remove a "queued" stop for an iDeployIt_receiver process.
[which]	-	use a receiver.conf file other than the default "receiver.conf" in "deploy_bay/config" directory in the associated depot. When invoked the receiver_**[which].**conf file is used. If which is "2" then "receiver_**2**.conf" will be used while if which is "from_atl" then "receiver_**from_atl**.conf" will be used.
-wait	-	Start the receiver process in the foreground, the calling process will wait for the receiver process to complete. The default is for the receiver to run in the background and immediately return control to the calling process. Useful for the monitoring of receiver runs as a scheduled task.

iDeployIt_shiptest

USAGE: iDeployIt_**shiptest -ver | -help | (path_to_iDeployIt_depot remote_location [class=0..9] [meg=1..100])**

Such that:

-ver   -   produces iDeployIt_shiptest **product version message**

-help   -   produces iDeployIt_shiptest **USAGE message**

path_to_iDeployIt_depot   -   The **full local path** to the associated iDeployIt Depot. This can be a network (UNIX) or UNC (Windows) path, but the local path is preferred if the process is to run on the same machine that holds the iDeployIt Depot.

remote_location   -   A required and valid **location_name**
    that is associated with a **location.hostmap** file of the name **<location>.hostmap** where <location> is the name given to the "receiver" location by the "shipper" location (i.e. "atlanta", or "london"). There is a 1-to-1 correspondence between the **location_name** (e.g. "atlanta") provided and a file named **"atlanta.hostmap"** in the **{depot}/deploy_bay/config/hostmaps** directory.

[class=0..9]   -   Optional: use a shipper.conf file other than the default "shipper.conf" file in "deploy_bay/config" directory in the associated depot. When invoked the shipper_**[class].**conf file is used. If class is "2" then "shipper_**2**.conf" will be used. Values 0-9 are valid. When omitted the default "classless" shipper.conf file is used.

[meg=1..100]   -   Optional: The size (in Meg) of the test packet to send to the remote location. If omitted the default of meg=10 is used

     63

iDeployIt_hostentry

The iDeployIt_**hostentry** program takes no parameters and can be executed by anyone responsible for creating a deployment_key. They key is provided so the administrator of the "receiver" site does not have to provide the "shipper" a userid, password, machine name or name of any "receiving" directory.

The program asks for:
- The machine name of the "receiver"
- The login id that will be used by the transport program
- The password for the login id
- The directory that the "receiver' allows the "shipper" to send to.

The **iDeployIt_hostentry** program generates a hostkey with the provided information. Executing the iDeployIt_hostentry program from the Windows OS shown below:

Enter the name (or IP address) of the "receiver" machine that will run the iDeployIt_**receiver** program then enter a carriage return to accept the input.

Enter the userid used to ftp the encrypted packets to the "receiver" machine then enter a carriage return to accept the input.

Enter the password for the userid. The password is not echoed to the terminal and must be entered twice. Once entered then enter a carriage return to accept the input.


Enter the **receiver_directory** where the shipper is allowed to send iDeployIt deployment packets. The directory can be an absolute path from the ftp root or relative to the ftp root. The provided directory should exist and be writable by the input userid. Once entered then enter a carriage return to accept the input:

If the information displayed is correct, then enter 'Y' to display the generated **deployment_key**.

If the information displayed is correct, then enter 'Y' to display the generated **deployment_key**.

Copy the key as needed before hitting any key to clear the screen and end the program.

Provide the key to the iDeployIt administrator for the "shipper" site to use to send iDeployIt packets to the receiver. The key is case sensitive and MUST be entered exactly as printed including the surrounding square brackets.

iDeployIt_crypt_key_gen

The iDeployIt_**crypt_key_gen** program takes no parameters and can be executed by anyone responsible for creating **"peer_to_peer" encryption keys**. The command can be run from either the shipper or receiver, but the generated key must be used by both to enable data encrypted at the shipper to be read at the receiver. It is expected that an iDeployIt administrator from one site will generate the key and share that key with the iDeployIt Administrator of the other site.

From the **{iDeployIt_bin}** directory, run the **iDeployIt_crypt_key_gen.exe**

Send the contents to a file by redirecting its output to a file.
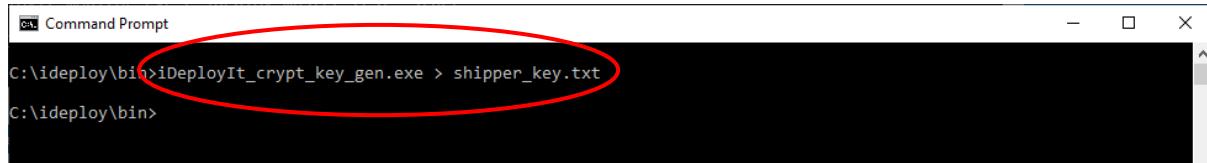
To enable peer-to-peer encryption for the classless shipper process the generated contents would be placed in the file **{iDeployIt_depot}**\config\keys**shipper_key.txt** while to enable peer-to-peer encryption for the class 1 shipper one would place the contents in the **{iDeployIt_depot}**\config\keys**shipper_key_1.txt** file.

66

IDeployIt_deploy

USAGE can be one of the two forms:

```
iDeployIt_deploy -hostid | -help | -ver | {path_to_iDeployIt_home product location
release_string [-show | -dump [-quiet | -resume | -abort]]}

   or

iDeployIt_deploy -hostid | -help | -ver | {path_to_iDeployIt_home key_range release_string [-
show | -dump [-quiet | -resume | -abort]]}

 where key_range is of [key_number[-|[-key_number]|,]..[key_number[-|[-key_number]|,]
```

Such that:

-ver	-	produces iDeployIt_deploy **product version message**
-help	-	produces iDeployIt_deploy **USAGE message**
-hostid	-	produces the unique host id information that is delivered to A Better Solution in order to request a license key.
path_to_iDeployIt_depot	-	The **full local path** to the associated iDeployIt Depot.
product	-	The Product name of the **product_name:location_name** pair to queue for deployment.
location	-	The Location Name of the **product_name:location_name** pair to queue for deployment.
key_range	-	The Deployment Key or a range of Deployment Keys to be queued for deployment. Can be the form of 1 Key or Multiple Keys separated by commas or a range of keys or a range of key separated by a comma. Example: 100 or 100,101,103 or 100-110 or 100-110,200-210.
release_string	-	Text to represent the **release name** of the current deployment.
-show	-	Produces a deployment summary report of matching keys based on **key_number** parameter.
-dump	-	Produces a deployment key report of matching keys based on the **key_number** parameter.
-quiet	-	Requested action is performed but, no output is displayed.
-resume	-	Resume a "paused" deployment.
-abort	-	Remove a "queued" deployment.

iDeployIt_monitor

```
USAGE: iDeployIt_monitor.exe -ver | -help | path_to_iDeployIt_depot ( -start | -
once | -stop | -restart | -reonce | -queue_stop | -dequeue_stop ) [-wait])
```

Such that:

-ver	-	produces iDeployIt_monitor **product version message**
-help	-	produces iDeployIt_monitor **USAGE message**
path_to_iDeployIt_depot	-	The **full local path** to the associated iDeployIt Depot. iDeployIt monitor runs on the same machine that holds the iDeployIt Depot.
-start	-	**starts the process** and continually runs for **infinite passes** as per the configuration parameters in the shipper.conf file.
-once	-	**starts the process** and runs for **one (1) pass** as per the configuration parameters in the shipper.conf file.
-stop	-	**Stops any** iDeployIt_**monitor** process for the associated depot
-restart	-	Performs an iDeployIt_monitor –**stop** then a –**start** for the associated depot.
-reonce	-	Performs an iDeployIt_monitor –**stop** then a –**once** for the associated depot.
-queue_stop	-	Attempts the stop of an iDeployIt_shipper (–**stop**) if the process cannot stop because it is working on a packet the stop is queued to be invoked when the process has completed its current packet.
-dequeue_stop	-	Remove a "queued" stop for an iDeployIt_shipper process.
[which]	-	use a receiver.conf file other than the default of "receiver.conf" in "deploy_bay/config" directory in the associated depot. When used the receiver_**[which].**conf file is use. If which is "2" then "receiver_**2**.conf" will be used while if which is "from_atl" then "receiver_**from_atl**.conf" will be used.
-wait	-	Start the receiver process in the foreground, the calling process will wait for the receiver process to complete. The default is for the receiver to run in the background and immediately return control to the calling process. Useful for the monitoring of receiver runs as a scheduled task.

iDeployIt_echo

The purpose of this utility is to provide users the ability to log information to the Deployment Log during a deployment. Simply, replace the call to you normal logging function (printf, echo, etc.) with a call to this utility. Output will be written to the Deployment Log if it is called in the context of a deployment. If the script is called outside the context of a deployment it will write to STDOUT.

```
USAGE: iDeployIt_echo.exe -ver | "string to echo"
```

Such that:

-ver - produces iDeployIt_echo **product version message**

-help - produces iDeployIt_echo **USAGE message**

string to echo - The text string you want sent to the iDeployIt log if called during the context of a deployment and written to STDOUT otherwise.

iDeployIt_print

The purpose of this utility is to provide users the ability to log information to the Deployment Log during a deployment. Simply, replace the call to you normal logging function (printf, echo, etc.) with a call to this utility. Output will be written to the Deployment Log only.

```
USAGE: iDeployIt_print.exe -ver | "string to print"
```

Such that:

-ver - produces iDeployIt_print **product version message**

-help - produces iDeployIt_print **USAGE message**

string to print - The text string you want sent to the iDeployIt log if and only if called during the context of a deployment and written to STDOUT otherwise.

Customer Support

To obtain additional information on iDeployIt or other services offered by A Better Solution, Inc., visit our web site at www.abs-consulting.com. To report problems with the iDeployIt software or documentation, please send e-mail to iDeployIt_team@abs-consulting.com for licenses please send an email to support@abs-consulting.com

Product Limitations

As of print, any known product limitations are outlined in the sections that follow.

iDeployIt Packet Sizes

All iDeployIt Packets are encrypted and compressed before sending. The total size of a deployment is only limited by the amount of space available on the shipper or receiver, but the maximum size of a *single file* within the deployment is **100 Gig**. For example, iDeployIt can deploy a package containing 100 files that are 99 Gig each (totaling 999 Gig), but cannot process a deployment which contains a single file greater than 100 Gig in size.