

Feature Brief



Server Packages

Introducing Server Packages in Otter 1.6

Repetitive manual processes are subject to occasional human error and add up to a significant loss of time and productivity. A prime example of this is manually logging into servers to view configuration and installed packages.

A central portion of Otter's functionality is a quick view of basic server configurations. With the Server Packages feature in 1.6, users can get that same ease of access when requesting reports on installed packages and deeper server configurations.

DevOps best practices call for process automation and clear communication strategies. The traditional system for accessing server information falls short of these goals in two major ways.

As stated, these systems rely heavily on manual tasks. However, Server Packages offer a responsible way to automate those tasks, cutting down on work-hours and risk of error.

Feature Brief

📦 Server Packages

Secondly, it creates a barrier for users unfamiliar with the workings of a server in general or the set-up of a specific server. This creates a bottleneck for any task dependent on information derived from those traditional systems, since only a small segment of personnel can actually access that data on their own.

Server Packages

With the Server Packages feature, users can configure Otter to collect information about the Chocolatey, PowerShell and Universal packages that are installed on their servers. From the Packages Tab, users can view and configure this feature so that packages are displayed in an easy-to-read report. This gives permitted users the ability to view installed packages without manually logging-in to the server.

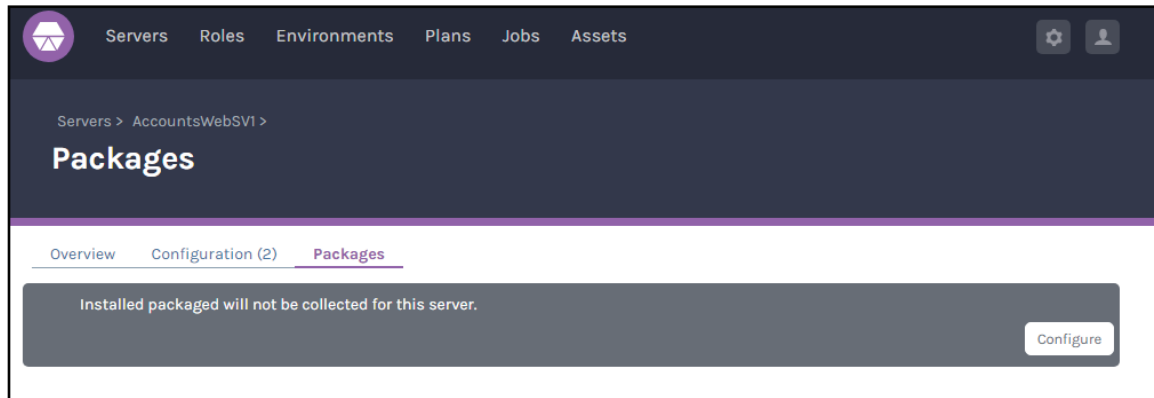
Servers > LOCALHOST >

Packages				
Overview Configuration Packages (24)				
DSC Module Packages				
Name	Version	Collected	Details URL	
Archive	1.1	4/21/2017 4:09:58 PM	-	✖
Environment	1.1	4/21/2017 4:09:58 PM	-	✖
File	N/A	4/21/2017 4:09:58 PM	-	✖
Group	1.1	4/21/2017 4:09:58 PM	-	✖
GroupSet	1.1	4/21/2017 4:09:58 PM	-	✖
Log	1.1	4/21/2017 4:09:58 PM	-	✖
Package	1.1	4/21/2017 4:09:58 PM	-	✖
ProcessSet	1.1	4/21/2017 4:09:58 PM	-	✖
Registry	1.1	4/21/2017 4:09:58 PM	-	✖

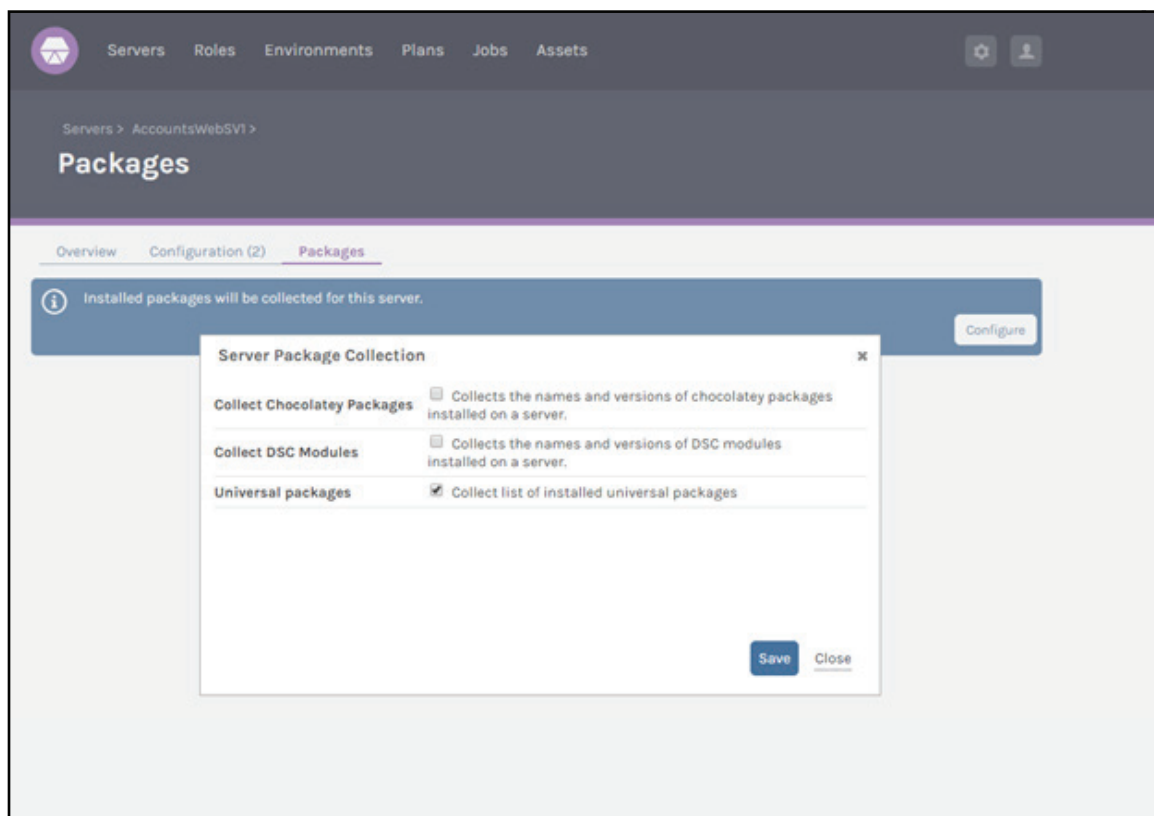
Feature Brief

Server Packages

When a user first goes to the packages tab a gray box will appear. This indicates that Otter is not configured to collect details about which packages are installed.



This feature is configured at the server level. Navigating to the packages tab on each server will display the option to configure and collect information about Chocolatey Packages, DSC modules, and ProGet Universal Packages. Otter will only display information when the boxes are selected.

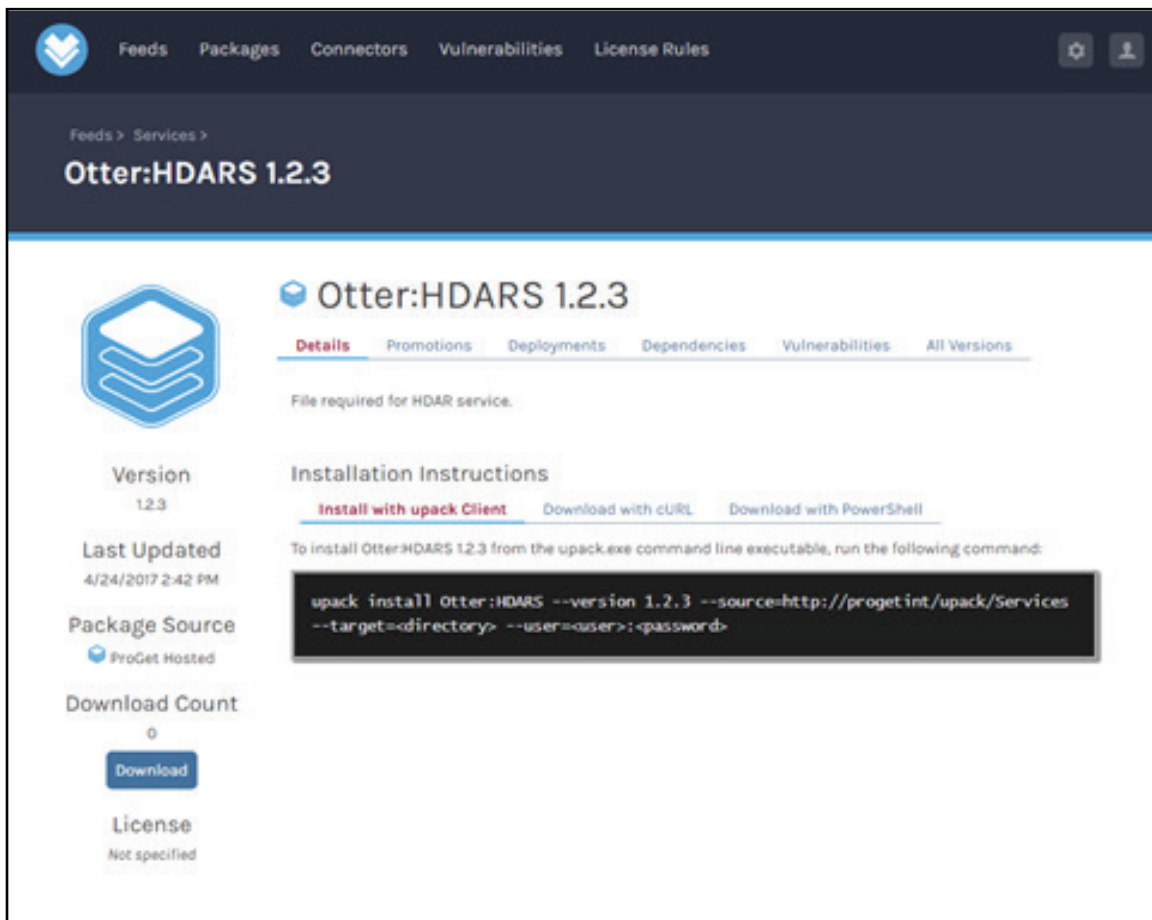


Feature Brief

📦 Server Packages

Built-In Support for ProGet Universal Packages, Chocolatey, and PowerShell DSC Modules

By default, the Server Packages feature integrates with ProGet Universal Packages, Chocolatey packages, and PowerShell DSC modules. Running the installations through the built-in Otter operations will record package/module names, versions, and, in some cases, links to further details – especially in conjunction with ProGet:



The screenshot shows the ProGet web interface for the 'Otter:HDARS 1.2.3' package. The top navigation bar includes 'Feeds', 'Packages', 'Connectors', 'Vulnerabilities', and 'License Rules'. The breadcrumb trail is 'Feeds > Services >'. The package name 'Otter:HDARS 1.2.3' is prominently displayed. On the left, a sidebar lists package details: Version (1.2.3), Last Updated (4/24/2017 2:42 PM), Package Source (ProGet Hosted), Download Count (0), and License (Not specified). The main content area features a hexagonal icon and tabs for 'Details', 'Promotions', 'Deployments', 'Dependencies', 'Vulnerabilities', and 'All Versions'. Under the 'Details' tab, it states 'File required for HDAR service.' and provides 'Installation Instructions' with three options: 'Install with upack Client' (selected), 'Download with curl', and 'Download with PowerShell'. A command box shows the installation command: `upack install Otter:HDARS --version 1.2.3 --source=http://progetint/upack/Services --target=<directory> --user=<user>:<password>`.

Feature Brief

📦 Server Packages

Customizable and Extensible

In parity with our other DevOps solutions, Server Package details can be included in any custom Collect or Ensure operation, making them fully extensible:

```
240         using (var sourceStream = entry.Open())
241         {
242             await sourceStream.CopyToAsync(targetStream).ConfigureAwait(false);
243         }
244
245         await fileOps.SetLastWriteTimeAsync(targetPath, entry.LastWriteTime.DateTime)
246     }
247 }
248 }
249
250 #if Otter
251     this.LogDebug("Recording server package information...");
252     await new DB.Context(false).ServerPackages_CreateOrUpdatePackageAsync(
253         Server_Id: context.ServerId,
254         PackageType_Name: "ProGet",
255         Package_Name: packageId.ToString(),
256         Package_Version: version,
257         CollectedOn_Execution_Id: context.ExecutionId,
258         Url_Text: client.GetViewPackageUrl(packageId, version)
259     ).ConfigureAwait(false);
260 #endif
261 }
262 catch (ProGetException ex)
263 {
264     this.LogError(ex.FullMessage);
265     return;
266 }
267
268     this.LogInformation("Package deployed!");
269 }
```

The extensibility points are limitless; with little effort, Server Packages could also be used to capture:

- MSIs installed on a Windows server
- Packages installed on a Linux server
- Docker containers present on a server

... and much more.