Push-Pull-Framework

The Push Pull framework is responsible for downloading remote content (pull), or accepting the delivery of remote content (push) to a local staging area. Content in the staging area is ingested into the File Manager system by the Crawler Framework. The Push Pull framework is extensible and provides a fully tailorable Java-based API for the acquisition of remote content.

Architecture

This section describes the architecture of the Push Pull framework, including its constituent components, object model, and extension points.

Architectural Components

The major components of the Push Pull Framework are the Daemon Launcher, the Daemon, the Protocol Layer, and the File Retrieval System, to name a few. The relationship between all of these components are shown in Figure 1.

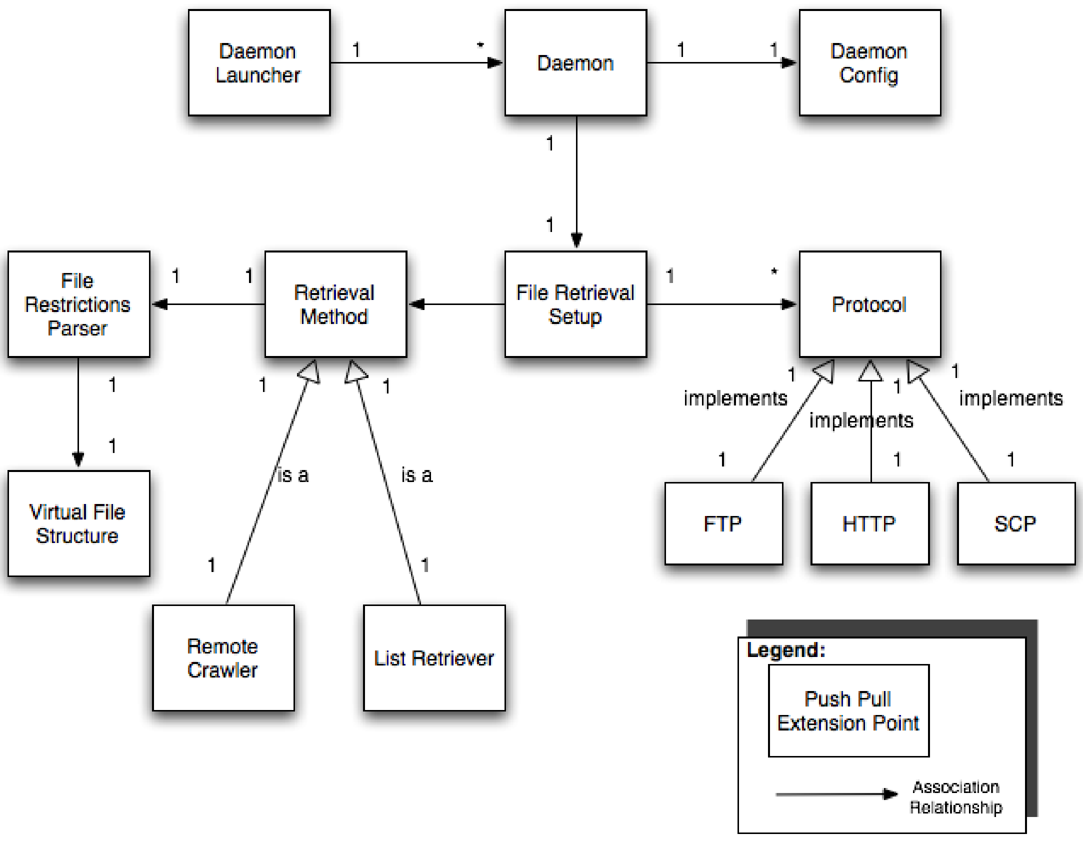

Figure 1. Push Pull Components

The Push Pull Framework provides a Daemon Launcher, responsible for creating new Daemon instances. Each Daemon has an associated Daemon Configuration, and has the ability to use a File Retrieval Setup extension point. This class is responsible for leveraging both a Protocol and a File Retrieval System to obtain ProtocolFiles, based on a File Restrictions Parser that yields eventually a VirtualFileStructure (VFS) model. The VFS defines what files to accept and pull down from a remote site.

Object Model

The critical objects managed by the Push Pull Framework include:

- **Protocol** – A means of obtaining content over some file acquisition method, e.g., FTP, SCP, HTTP, etc.
- **Protocol File** - *Metadata* information about a remote file, including its *ProtocolPath*.
- **Protocol Path** - A pointer to a remote *Product* file's (or files') location, which can be used to derive metadata and determine where to place the file in the local staging area built by the Push Pull Framework.
- **Remote Site** - Descriptive information about a remote site, including the username/password combination, as well as an origin directory to start interrogating.

Each *Protocol* delivers one or more Protocol Files. Each *ProtocoFile* is associated with a single *RemoteSite*, and each *ProtocolFile* is associated with a single *ProtocolPath*. These relationships are shown in Figure 2.

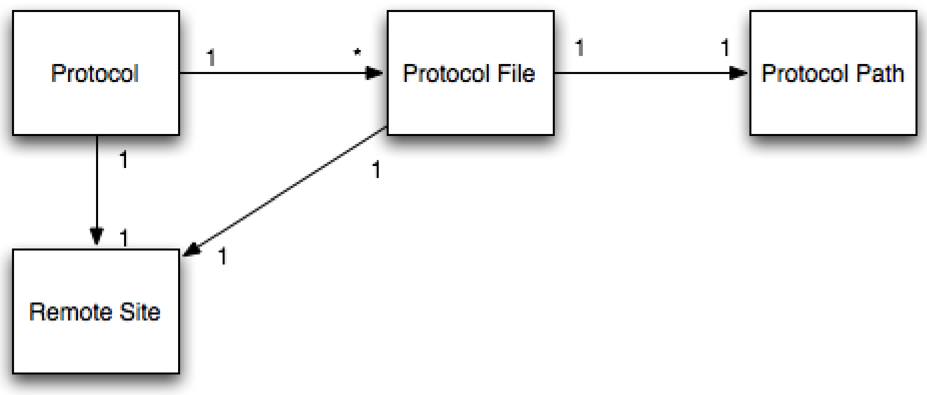

Figure 2. Push Pull Object Model

Extension Points

Push Pull Framework was constructed by making use of the Factory Method pattern. The use of this pattern was intended to provide multiple extension points for the Push Pull Framework. An extension point is an interface within the Push Pull Framework that can have many implementations. This is particularly useful when it comes to software component configuration because it allows different implementations of an existing interface to be selected at deployment time. Each of the core extension points for the Push Pull Framework is described in TABLE I.

TABLE I. Push Pull Extension Points

Extension Point	**Description**
Protocol	The Protocol extension point is the heart of the Push Pull framework. It is responsible for modeling remote sites and for obtaining their content via different Retrieval Methods, using different File Restrictions Parsers.
Retrieval Method	The Retrieval Method extension point is responsible for orchestrating download (pull) and acceptance (push) of remote content.
File Restrictions Parser	The File Restrictions Parser extension point is responsible for defining how to accept or decline files encountered by a Retrieval Method, in essence modeling remote file and directory structures.
System	The extension point that provides the external interface to the Push Pull Framework services. This includes the Daemon Launcher interface, as well as the associated Daemon interface, that is managed by the Daemon Launcher.