Directory Structure

In the spirit of convention over configuration, the following data structure is recommended for all Aspire installations.

- This data structure is assumed by all of the prepackaged applications provided by Search Technologies.

The specified directory structure specified in this page is a “standard” implementation. Thus, there may be folders and files that are missing on the initial default Aspire distribution, but are created later in response to actions performed in Aspire.

On this page:

Data Structure

bin

Binary files and shell scripts for managing the Aspire application program.

- *Startup and Shutdown Commands*
- *Master Password Commands*
- *Windows Service Commands*

  prunsrv*These files are required to install Aspire as a Windows Service. Listed on the wiki as “Windows Service Commands.”*
  - **amd64**
  - **ia64**
  - **x86**

bundles

- **aspire**
  - *Aspire component and AppBundle jar files*

Note that in installations with internet access, Aspire bundles (except for the Aspire application) will be automatically loaded from the local Maven repository. Installations without internet access can store the bundles in the ${aspire.home}/bundles directory. Bundles in ${aspire.home} /bundles are preferred over Maven bundles (although the exact order is specified in the settings.xml file)

- **system**
  - *Felix OSGi component jar files*

cache

Copies of read-only files from components. Can be deleted without loss of data. Automatically regenerated on restart / reload, if necessary. Do not place a custom version of any of these files in this directory. When Aspire is restarted it will be replaced with the original file. To use a custom one, it needs to be placed outside of the cache folder (${appbundle.home}). Important to note that this directory only exists on compiled distributions. For example, after running “mvn clean package” on a distribution project.

- **appbundles** / <group-id> / <artifact-id> / <version-number> --> ${appbundle.home}

  Holds the application bundle contents decompressed from the original app bundle JAR file. Note: There will be no data sub-directory for app bundles in this location. If application bundle requires static (read-only) data files, they should be put into the “config” directory (for now). All transient data must be created as needed in the ${app.data.dir} (i.e. under ${aspire.home}/data/${app.name})

  config - *config files and sub directories for the App Bundle*
  - application.xml - *The standard Aspire configuration file which describes all of the components, component configurations, and pipelines and how they're all tied to gether to make an application. This file must be called "application.xml" for all application bundles.*
  - application-dxf.xml - *The DXF form for editing the properties required when installing this application. Must be called "application-dxf.xml" for all application bundles.*
  - **xsl** - *Holds default XSL transforms distributed with the App Bundle*
- **lib** - *jar files distributed with the App Bundle*

  web
  - **httpfeeder** / <servlet-name> - *Web files for supporting HTTP feeders*

felix / **bundle#**

Directories and files as required by Apache Felix. The entire cache/felix directory structure is automatically deleted whenever Aspire is shutdown and restarted with the standard startup scripts to ensure a clean startup to a known state.

data
- aspire-persisted-components.xml - Holds component configurations

resources / <implementation-class>

- Holds decompressed files from the "resources/" directory from a component's JAR file. Files are added to this folder as requested from the Aspire admin interface, and then fetched from this location (instead of pulling them from the Jar) thereafter. The implementation class is the actual fully-qualified Java class name, e.g. "com.searchtechnologies.aspire.application.AspireApplicationComponent".

config

- application.xml

  The standard Aspire configuration file, describes components, component configurations, and pipelines for an Aspire application. There can be any number of application.xml files (and the names can change) stored in the config directory. Not required if all Applications are downloaded from Maven as App Bundles. Note that for distributions that are destined to App Bundles, the file must be called "application.xml".
- settings.xml

  Holds settings for the Aspire framework appropriate to this computer. Includes property settings, port addresses, Maven repository settings, etc.
- application-dxf.xml

  Holds the DXF form for setting properties for the associated application.xml file. If you have multiple application.xml files (with different names), you can also have multiple application-dxf.xml files, as long as the base file name (the part before "-dxf.xml") is the same. For distributions that are destined to become App Bundles, this file must be called "application-dxf.xml".
- felix.properties

  Holds the main admin port address for Felix, plus other Felix properties such as Felix system components to be auto-loaded and the OSGi web console user/password.
- felix-template.properties

  The felix.properties template which is modified and then copied to remote servers when a remote installation is performed. This is typically the same as felix.properties except that the port address setting is removed.
- available-applications-template.xml

  A sample available-applications.xml file, which can be edited and then renamed to "available-applications.xml" if the user wants to hard-code / augment the available applications for their user interface.
- layout.xml

  Holds the layout of the content sources in the Content Source Management page.
- entlitlements.xml

  Contains the information of the connectors, publishers and applications, that you can use in the Aspire server.
- aspireShellDefault.xml

  A sample aspireShell template file, which can be edited and then renamed to "aspireShell.xml" if the user wants to hard-code / augment the available applications for their aspire shell.
- **content-sources** - *Contains all the content sources configurations for this server.*
  - Content-Source [Example]
    - connector.xml

      Has the Advanced Connector Properties of the content source.
    - content-source.xml

      Has the Content Source Properties of the content source.
    - general.xml

      The General Tab information that includes Name, Schedule, if it is active, ...
    - workflow.xml

      Contains all the rules and applications used in the different Workflow Trees, as well as the Trees structure
- **workflow-libraries** - *Contains all the shared libraries for this server.*
  - Library.xml [Example] - Has the rules and applications shared to this library.
- **zooKeeper** - *Contains embedded zooKeeper data.*
- **passwords**
  - master.pswd - Holds the master password which is used to encrypt and decrypt all other passwords stored in Aspire. Aspire will use that path to look up master passwords. but if it doesn't exist, Aspire uses a default value.

    Administrators are supposed to create the file and the folder if they want to override the default master password. Note that the conscientious administrator will ensure that only the user which is running the Aspire Framework has access to the master.paswd file. Most passwords are stored in settings.xml and are encrypted either using the RESTFul interface (and then installed as an application property) or through the encryptPassword command-line utility.
- **content-sources**

  Holds the content sources configurations created from the UI or synchronized with ZooKeeper.

  - ContentSourceExample

general.xml - Contains the general information associated with the content source. content-source.xml - Contains the content source specific configuration to connect and perform a crawl against a given repository. connector.xml - Contains the connector application definition and property values to install the connector into Aspire. workflow.xml - Contains the workflow configuration of all workflow trees configured for the content source: afterScan, onPublish, onAddUpdate, onDelete and/or onError.

- **workflow-libraries**

  Holds the shared workflow libraries created from the UI or synchronized with ZooKeeper.
- **xsl**

  Holds example XSLT files for using the post-http component. These are not used by the standard post-to-{searchengine} App Bundles (unless specifically configured to do so by the administrator).

  - README.txt - *Explains the purpose of this directory.*
  - aspire2Fast.xsl - *Sample XSLT for sending documents to Microsoft Fast.*
  - aspire2GSA.xsl - *Sample XSLT for sending documents to the Google Search Appliance.*
  - aspire2Solr.xsl - *Sample XSLT for sending documents to SOLR.*

data

Main data directory for all persistent data for all applications installed into this instance of Aspire. The contents of this directory can be completely removed if you wish to start up a completely empty system (do not delete the data directory itself, however). All components will create their required sub-directories as needed.

aspire - *Reserved location for holding data required by the Aspire Application itself.*
uploads - *Holds data files uploaded to the Aspire Application servlet. Currently only used to upload private key files for remote installations by the Administration GUI. only appears when a user attempts an upload.*

- pnelson.pem [Example]

- **shared** --> ${shared.data.dir}

  Intended for data shared across applications installed into Aspire. Currently, there are no examples of such data.
- {Application-Name} --> ${app.data.dir}

  Holds data for each Aspire installed application (either an App Bundle or application.xml configuration file). The sub-directory name will be the same as the application name, which is also the name of the top-level component. For example, "CSManager", "CIFSConnector", etc. All data which is needed by an individual application should be stored here.

  - CIFSConnector [Example]

    The CIFSConnector is one of several connectors which use a snapshot method for crawling content sources.

    - snapshots [Example]

      Holds connector snapshots.

      content-source-32
      Holds all snapshots for the specified content source. The number specified is the same as the RDBMS ID for the content source.

      - SNAPSHOT-5.snapshot
        Snapshot files which have been successfully completed and closed end in ".snapshot". Some (configurable) number of old versions of these files will be kept for backup and system recovery.
      - SNAPSHOT-6.snapshot
      - SHAPSHOT-7.in-progress
        ".in-progress" snapshots are being actively written by the connector. If these files exist when the connector starts up, then these are partially written files from the previous connector run. If this happens, then the file will be renamed ".recovery" and used to recover as much of the previous crawl as possible.
      - SNAPSHOT-7.recovery

        Once recovery has begun, the most recent ".in-progress" snapshot is renamed ".recovery" and then a new ".in-progress" snapshot will be created, which will contain all recoverable jobs from the previous .in-progress snapshot. Any ".recovery" snapshots which exist on startup are deleted.
- CSManager [Example]

  The CSManager is the App Bundle which is responsible for managing the database of content-sources and scheduling content sources for crawling on connectors.

  - db
    When configured to use an internal RDB, the CSManager will launch an embedded version of Apache Derby. The 'db' directory holds the Apache Derby database on disk.

    - *Apache Derby files go here.*

lib

- *Additional jar files required by Aspire go here.*

The lib directory is primarily used for JDBC drivers required for connecting to relational databases. Other 3rd party jars required by individual connectors (say) are usually bundled inside the component jar files and downloaded with the connector that requires them.

log

Holds all log files for Aspire and all nested components and applications. After initial installation, logs won't exist. They are created when Aspire starts up the first time. Thus, the user can delete the logs folder at any time, and after starting Aspire again, the folder will be re-created along with the corresponding logs.

- aspire.log

  The top-level aspire.log file. Holds logs from the Aspire Application, which manages all of the other configurations loaded into Aspire.
- autoStart.log

  Holds the status of the auto-start, including any errors which occurred while starting up the configurations specified in the settings.xml file.
  {Application-Name}

  Holds log data for individual installed applications.

  - incoming.jobs - *Holds jobs received by the installed top-level application (configuration or App Bundle).*

    For packaged applications, turning on debug will write a journal of all jobs received by the application. Note that this is not default behavior of the Aspire framework, but must be specified in the configuration file.
  - outgoing.jobs - *Holds jobs produced by the installed top-level application.*

    For packaged applications, turning on debug will write all jobs produced by the application into this file. Again, this is not default behavior of the framework, but must be configured.
  - {nested component logs}
    Each component can have its own set of log files. These are stored on a component-by-component basis in directories with the same path as the component names.

resources

Holds resources required by individual components. Resources which may need to be updated by the system administrator are provided here as part of the distribution. Currently this is only the "DomainSuffixes" directory. This directory can also be used to hold a copy of a component's resource files (typically stored in the component jar) for easy editing and updating of the administration interface. When accessing admin web interface files for a component, the Aspire Application servlet will first check these directories, then it will check the "cache/felix/bundle#/data /resources" directory. If the file is still not found, it will extract it from the component Jar file and save a copy in "cache/felix/bundle#/data /resources".

- **com.searchtechnologies.aspire.framework.DomainSuffixes**
  Holds a list of top-level domain names which can be used to extract the domain name from a host name. Note that these files originally came from Apache Nutch. It is likelt that these files will be refactored in the near future.

  - domain-suffixes.xml
  - domain-suffixes.xsd
- com.searchtechnologies.aspire.components.ADUserGroupExpansion [Example]
  As an example, components which require direct access to Windows C++ APIs, will first need to unpack the DLL from the jar into the resources directory before the DLL can be accessed by Aspire using java JNI. The ADUserGroupExpansion component is one such component.

  - AD_UserGroupExpansion_32bit.dll [Example]
  - AD_UserGroupExpansion_64bit.dll [Example]

web

The web directory holds static web pages which are served up either for administrative or end-user interfaces. Some simple UIs can be created on top of Aspire, using Aspire pipelines to provide the necessary RESTful interfaces, and then using jQuery or XSLT to create the necessary user interface tools.

- **components**
- **httpfeeder** / <servlet-name> - *Web files for supporting HTTP feeders.*
- If your application has an HTTP feeder, you can access files from the servlet by specifying the URL http://{server:port}/{servlet}/{file}. These files are stored in the web/{servlet} directory. The {servlet} name is configured as part of the HTTP feeder configuration.

Related Pages

- Launch control
- Aspire shell
- In-depth administration