# Operations Center

F/X Adapter Guide

**September 2016**

# Contents

# About This Guide

The *F/X Adapter Guide* provides instructions for administering the F/X adapters.

## Audience

This guide is intended for Operations Center system administrators.

## Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the *User Comments* feature at the bottom of each page of the online documentation.

## Additional Documentation & Documentation Updates

This guide is part of the Operations Center documentation set. For the most recent version of the *F/X Adapter Guide* and a complete list of publications supporting Operations Center, visit our Online Documentation Web Site at Operations Center online documentation.

The Operations Center documentation set is also available as PDF files on the installation CD or ISO; and is delivered as part of the online help accessible from multiple locations in Operations Center depending on the product component.

## Additional Resources

We encourage you to use the following additional resources on the Web:

- NetIQ User Community (https://www.netiq.com/communities/): A Web-based community with a variety of discussion topics.

- NetIQ Support Knowledgebase  (https://www.netiq.com/support/kb/?product%5B%5D=Operations_Center): A collection of in-depth technical articles.
- NetIQ Support Forums (https://forums.netiq.com/forumdisplay.php?26-Operations-Center): A Web location where product users can discuss NetIQ product functionality and advice with other product users.

## Technical Support

You can learn more about the policies and procedures of NetIQ Technical Support by accessing its Technical Support Guide (https://www.netiq.com/Support/process.asp#_Maintenance_Programs_and).

Use these resources for support specific to Operations Center:

- Telephone in Canada and the United States: 1-800-858-4000
- Telephone outside the United States: 1-801-861-4000
- E-mail: support@netiq.com (support@netiq.com)
- Submit a Service Request: http://support.novell.com/contact/ (http://support.novell.com/contact/)

## Documentation Conventions

A greater-than symbol (>) is used to separate actions within a step and items in a cross-reference path. The > symbol is also used to connect consecutive links in an element tree structure where you can either click a plus symbol (+) or double-click each element to expand them.

When a single pathname can be written with a backslash for some platforms or a forward slash for other platforms, the pathname is presented with a forward slash to preserve case considerations in the UNIX* or Linux* operating systems.

A trademark symbol (®, ™, etc.) denotes a NetIQ trademark. An asterisk (*) denotes a third-party trademark.

# 1 Introduction

The Operations Center F/X (File and XML) adapter provides the capability to distribute file and XML-based data collection, parsing, processing, and alarm generation using F/X Monitors into multiple Operations Center servers.

This guide describes the options for starting and stopping the F/X Monitor, as well as the range and sophistication of parsing options in the F/X Monitor. The examples illustrate how alarms are surfaced via an F/X adapter and how to manipulate them using a standard *MODL* hierarchy configuration.

F/ X adapters can host any number of F/X Monitors. F/X Monitors can deliver alarm content to any number of F/X adapters. F/X Monitors can be configured to multiplex alarms or operate a grouping policy where F/X adapter locations are defined in terms of Primary and Backup mode.

# 2 F/X Adapter Configuration

The F/X adapter is a hierarchy-based adapter with the ability to receive alarms from multiple F/X Monitors. Review the following sections to configure and start the adapter:

## 2.1 Defining the Adapter Properties

To define F/X Adapter properties:

1. In the *Explorer* pane, expand the *Administration* root element.
2. Right-click *Adapters*, then select *Create Adapter* to open the Create Adapter dialog box.
3. Click the *Type* drop-down list, then select *NOC - F/X Adapter*.

   A default name of the Event Manager adapter (Adapter: Operations Center Event Manager™) displays in the *Name* field. Default adapter properties specific to the Event Manager display in the Properties table.
4. In the *Name* field, replace the default name with one that defines your system.
5. Create the adapter.
6. After creating the adapter, define the following adapter properties:

   **Adapter Listen Port:** The TCP/IP port on which the adapter listens for incoming connection requests. This port must match the target port setting in the F/X Monitor configuration. The default is 50001.

   **Alarm Columns:** A comma-separated list that determines which alarm columns display and the order in which the alarm items display (date/time, rule, and so on) in the *Alarms* view. The default list is:

   ```
   FilePath,CollectorType,Parser Name=ParserName,alarmMessage
   ```

   **Hierarchy File:** A file in the `/`*`OperationsCenter_install_path`*`/database` directory that contains an XML description of the element hierarchy to build below the adapter element. The default is:

   ```
   examples/FXAdapterHierarchy.xml
   ```

   **Maximum Alarms:** The maximum number of alarms that the adapter should maintain. Enter -1 to allow an unlimited number of alarms.

   **Monitor Handshake Max Wait:** The maximum amount of time that the adapter waits for a valid handshake to be completed with a remote F/X Monitor. If a valid handshake is not established within the specified time interval, the adapter drops the remote connection. The default is 10 seconds.

   **Persist Alarms:** If set to yes, the adapter automatically stores and recovers alarms when the adapter shuts down and restarts. The adapter also performs background writes of alarm persistence data to ensure an alarm persistence source is available even when an unclean shutdown of the Operations Center session occurs.

**Script.onError:** A script that executes if the adapter fails for any reason. For example, the script can print the reason for the failure as msg using log.info(msg).

**Script.onInitialized:** A script that executes when the adapter initializes. All of the `Script.` properties are optional.

**Script.onStarted:** A script that executes when the adapter starts, either manually or automatically, when the Operations Center server starts.

**Script.onStopped:** A script that executes after manually stopping the adapter.

**StylesheetFile:** The stylesheet file in the `/`*`OperationsCenter_install_path`*`/database` directory that applies to the HierarchyFile as a style markup to produce the final output.

**UseAlarmTimesFor CondChanges:** The date/time stamp to use for all alarm data stored by the Data Warehouse. If True, the date/time stamp information originates directly from the alarm information. If False, the date/time stamp is when the Operations Center server received the alarm. The default is True.

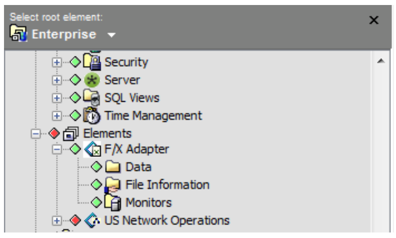
7 Continue to Section 2.2, “Starting the Adapter,” on page 10.

## 2.2 Starting the Adapter

To start the F/X Adapter:

1 To automatically start the F/X adapter when the Operations Center server starts, select the *Start Adapter Automatically* check box in the Create Adapter (or Adapter Settings) dialog box.

After the adapter is started, it displays as a top-level object in the *Elements* hierarchy in the *Explorer* pane. Depending on the hierarchy file used, the structure beneath the adapter varies. The following figure shows the default hierarchy, with no alarms received yet:



2 Because the F/X adapter processes connection requests from remote F/X Monitors, configure F/X Monitors to send alarms to the adapter.

For instructions, see Chapter 4, “The F/X Monitor,” on page 15.

# 3 Viewing F/X Alarms

When the adapter receives an alarm, it generates an alarm and processes it through the hierarchy. F/X Monitors dynamically create rich alarms with properties and values derived from a data source, using the parsing rules defined in the Monitor for the data source.

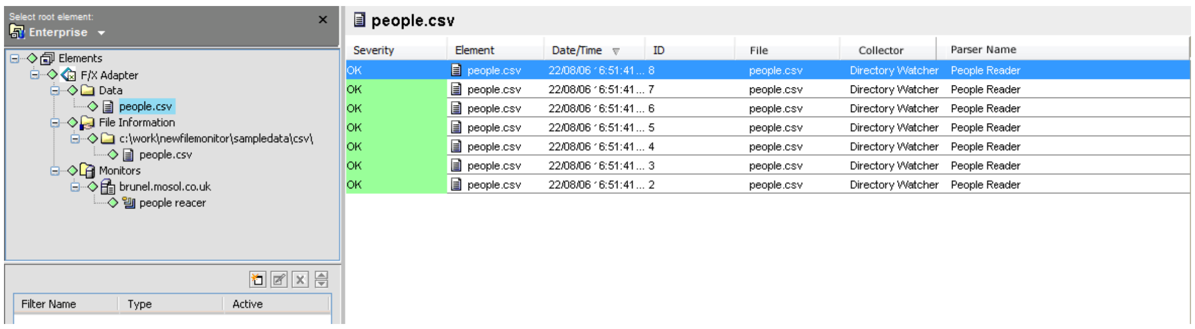
For example, consider the following CSV source file:

```
type,firstname,lastname,email,cellphone,office,state
Employee,Fred,Bloggs,fred@bloggs.com,17049249942,Dallas,TX
Employee,Joe,Schmoe,jschmoe@users.com,1843800002,NYC,NY
Employee,Diane,Tomeetu,dtomeetu@anyone.com,14829730924,San Jose,CA
Employee,Jane,Shame,j.shame@bigcorp.com,19320000752,Las Vegas,NV
Friend,Simon,Bennett,sbennett@people.com,13829492900,Denver,CO
Friend,Justin,Woodpuddle,justin@woodypuddle.com,13284619900,Miami,FL
```

It is possible to configure an F/X Monitor to monitor CSV files and parse in the content when a new file appears or is updated. By default, the Monitor parses the file content and renders alarms, in this case using the first row of the CSV file to define property (or cell) names.

In Figure 3-1, the alarms received by an F/X adapter display in the Operations Center console's *Alarms* view:

***Figure 3-1*** *The Alarms View*



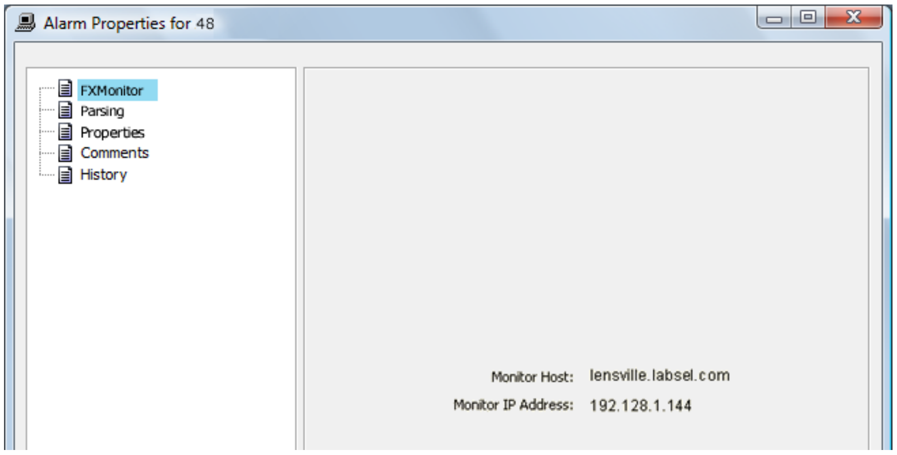
| Severity | Element | Date/Time | ID | File | Collector | Parser Name |
|---|---|---|---|---|---|---|
| OK | people.csv | 22/08/06 ' 6:51:41 ... | 8 | people.csv | Directory Watcher | People Reader |
| OK | people.csv | 22/08/06 ' 6:51:41 ... | 7 | people.csv | Directory Watcher | People Reader |
| OK | people.csv | 22/08/06 ' 6:51:41 ... | 6 | people.csv | Directory Watcher | People Reader |
| OK | people.csv | 22/08/06 ' 6:51:41 ... | 5 | people.csv | Directory Watcher | People Reader |
| OK | people.csv | 22/08/06 ' 6:51:41 ... | 4 | people.csv | Directory Watcher | People Reader |
| OK | people.csv | 22/08/06 ' 6:51:41 ... | 3 | people.csv | Directory Watcher | People Reader |
| OK | people.csv | 22/08/06 ' 6:51:41 ... | 2 | people.csv | Directory Watcher | People Reader |

Each alarm displayed in the *data* section represents a row in the CSV file. The F/X Monitor automatically parses the CSV content as alarm properties and values.

To view the details of an alarm:

1 In the *Alarms* view, double-click an alarm or right-click an alarm, then select *Properties* to display the alarm properties.

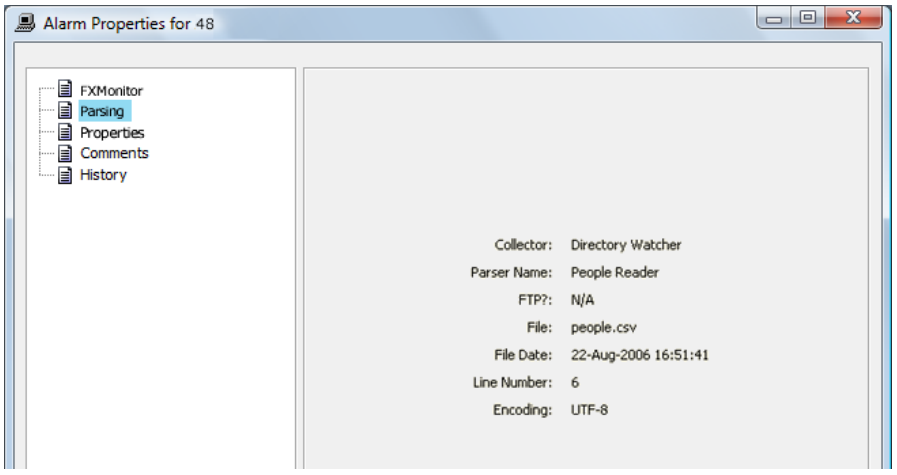
   The F/X Monitor page displays the hostname and IP address of the F/X Monitor that generated the alarm:



Monitor Host: lensville.labsel.com
Monitor IP Address: 192.128.1.144

2 Click *Parsing*.

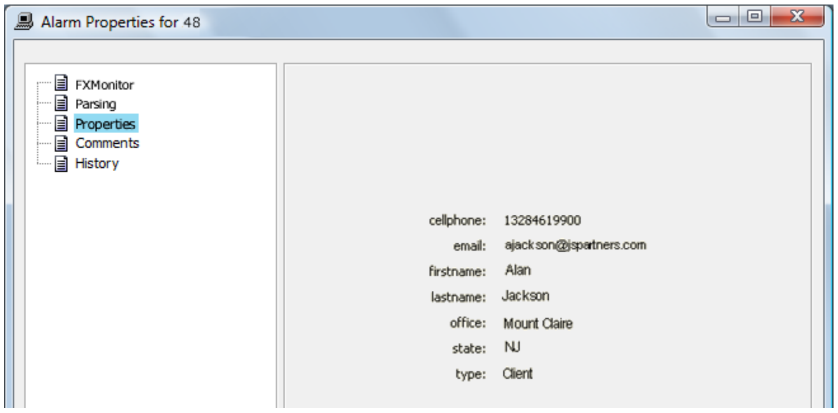
   The Parsing page provides full details on the parsing used to generate the alarm. This provides information about the original file the alarm was generated from, the parser used, the file line number and date stamp, and any encoding used:



Collector: Directory Watcher
Parser Name: People Reader
FTP?: N/A
File: people.csv
File Date: 22-Aug-2006 16:51:41
Line Number: 6
Encoding: UTF-8

**3** Click *Properties*.

The Properties page provides data details, indicating the alarm payload in terms of data properties and values:

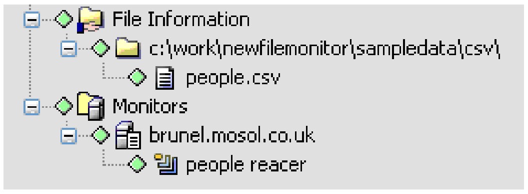


|  |  |
|---|---|
| cellphone: | 13284619900 |
| email: | ajackson@jspartners.com |
| firstname: | Alan |
| lastname: | Jackson |
| office: | Mount Claire |
| state: | NJ |
| type: | Client |

**4** To open the Hierarchy editor, right-click the F/X adapter element, then select *Edit Hierarchy Definition*.

For information on using the editor, see *The Operations Center XML Editor* in the *Operations Center Server Configuration Guide*.

F/X Monitors also generate diagnostic information pertaining to Monitor status and file status, which can provide details on parsing and files encountered. In the default hierarchy, these alarms generate a file view (*File Information*) showing directories that are monitored and matched files. The *Monitors* hierarchy shows the location of connected Monitors and the active parsers:



Administrators who are familiar with the configuration of hierarchy files can use Operations Center’s Hierarchy editor to manipulate the element hierarchy to refine the structure and presentation of data.

# 4 The F/X Monitor

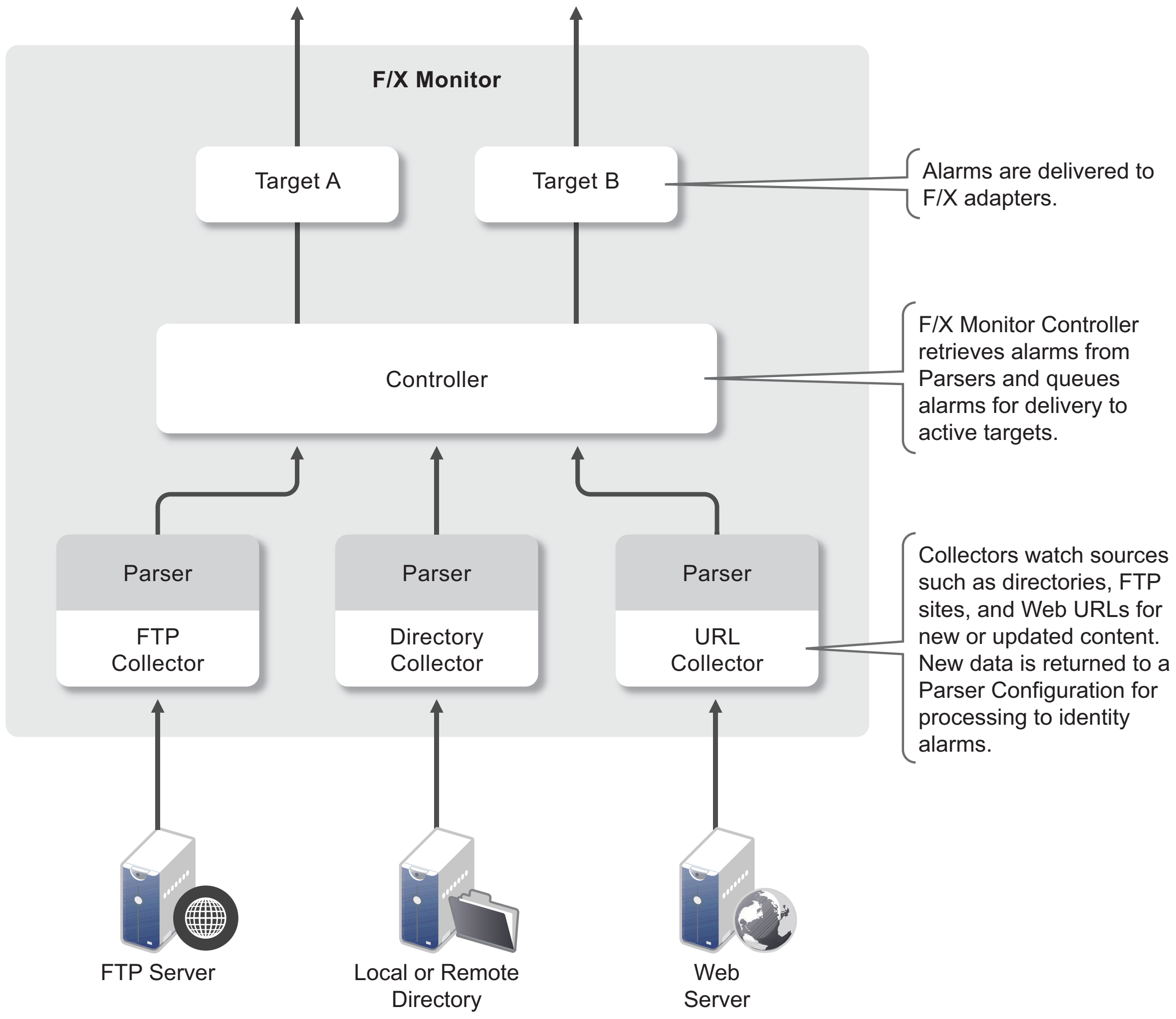
You must configure F/X Monitors to send alarms to the F/X adapter. Identify the F/X Monitor using either an IP address or a hostname and the TCP/IP listen port configured for the adapter. This section describes the packaging of the F/X Monitor, explains the related terminology, and focuses on configuring the Monitor to perform various collection and parsing tasks.

## 4.1 F/X Monitor Architecture

F/X Monitors are comprised of Alarm Delivery Instances (management of connection with remote adapters and alarm delivery), central processing of alarms, and collector/parser combinations. The collector defines the mechanism used to collect file content. The parser describes how to parse the data from files into alarms.

Both collectors and parsers reference plug-in classes, enabling extra collection and parsing capabilities to be added to the F/X adapter base capabilities.

*Figure 4-1 F/X Monitor Architecture*


F/X Monitor
Target A
Target B
Alarms are delivered to F/X adapters.
Controller
F/X Monitor Controller retrieves alarms from Parsers and queues alarms for delivery to active targets.
Parser
FTP Collector
Parser
Directory Collector
Parser
URL Collector
Collectors watch sources such as directories, FTP sites, and Web URLs for new or updated content. New data is returned to a Parser Configuration for processing to identity alarms.
FTP Server
Local or Remote Directory
Web Server

## 4.2 Starting an F/X Monitor

The F/X Monitor is delivered in the `/`*`OperationsCenter_install_path`*`/html/classes/fxmonitor.jar` file. It can operate in any Java 1.5 (or above) Runtime Environment. There are no license requirements for the operation of an F/X Monitor; only F/X adapters are licensed.

To start the F/X Monitor, issue the following command:

```
java -jar fxmonitor.jar -c configuration_filename_and_path -l
logging_configuration_filename_and_path [-m monitor_name] [-s
signal_file_directory]
```

where:

The command arguments are as follows and can be listed in any order:

- **-c:** Specifies the F/X Monitor XML configuration file for F/X Monitor operation. The full file path is required.

  For information on configuration files, seeSection 4.5, “F/X Monitor Configuration File,” on page 18.
- **-l:** Specifies the log4j compliant configuration file for logging. The full file path is required.

  For information about logging, see Chapter C, “Logging Configuration File,” on page 49.

  For a full set of return codes, see Chapter A, “F/X Monitor Return Codes,” on page 45.
- **-m:** (Optional) Specifies the monitor name to identify the F/X Monitor session so that it can be closed at some future point in time. Monitor Names must be unique on a given host system—a call to start a session.
- **-s:** (Optional) Specifies the directory for storing monitor signal files. If unspecified, uses the default temporary directory of the operating system that the monitor runs on:
  - **Windows:** `c:\documents and settings\administrator\local settings\temp`

    Using the default can be a problem for Windows implementations where group policy prohibits access to local user `\temp` directories.
  - **Unix:** `/tmp`

For example:

```
java -jar fxmonitor.jar -c D:\MyConfig\config.xml -l D:\MyLogs\logging.xml -m
MyMonitor
```

## 4.3 Closing/Cleaning an F/X Monitor

Close the F/X Monitor by calling the F/X Monitor class by specifying the `Close` command and the name of the Monitor to close.

To close an F/X Monitor, issue the following command:

```
java -jar fxmonitor.jar -C [-X] -m monitor_name [-s signal_file_directory]
```

where:

- **-X:** (Optional) Cleans up the logging and signal directories.
- **-m:** (Required) The name of the F/X Monitor session.

  If the F/X Monitor is started without a Monitor Name, then the `Close` command cannot be used.
- **-s:** The signal file directory.

  The signal file parameter is required if it is was used to start the monitor.

## 4.4 Legacy Mode Start/Stop Commands

Legacy mode is provided for compatibility purposes only, and should be avoided, if possible.

To start the F/X Monitor using legacy mode, issue the following command:

```
java -jar fxmonitor.jar configuration_filename_and_path log
configuration_filename_and_path [monitor_name]
```

To stop the F/X Monitor using legacy mode, issue the following commands:

```
java -jar fxmonitor.jar CLOSE monitor_name
java -jar fxmonitor.jar CLEAN monitor_name
```

The parameters must appear in the order shown. The signal file directory path cannot be defined using the legacy mode.

## 4.5 F/X Monitor Configuration File

The F/X Monitor ships with some sample configuration files for the Monitor and an example. For more information, see Chapter C, “Logging Configuration File,” on page 49.

F/X Monitors are configured using an XML configuration file (with the exception of logging configuration).

The configuration file structure logically follows the F/X Monitor architecture. It defines the following items:

- F/X Monitor level settings
- Adapter definitions, which are the location and type of recipients
- Reader definitions, which are of collectors and the parsers they use
- Parser configurations, which are named parser configurations that are referenced by reader definitions

  Currently, there is an *n*-to-1 relationship between reader definitions and parser configurations.

As an XML document, the header in the F/X Monitor configuration file is an XML syntax declaration. Also the F/X Monitor configuration must be encapsulated within F/X Monitor open and close tags.

The following code shows the XML tag structure of an F/X Monitor configuration file:

```
<?xml version="1.0"?>

<fxmonitor>
  <adapters>
      <!--
        Add adapter definitions here Adapter definitions are the destinations
        for alarms generated by readers
      -->
  </adapters>

  <settings>
    <!--
        Optional ability to specify a filestore for temporary files created by
        the FX Monitor if not specified the FX Monitor uses a temporary
       directory underneath the O/S reported temporary directory.
    -->
      <filestore></filestore>
  </settings>

   <readers>
     <!--
      Add reader definitions here.  Readers combine a collector and a parser to
     collect and parse data in order to generate alarms
      -->
  </readers>

  <parsers>
    <!--
      Add parser definitions here. Parser definitions define a named parser
      configuration,which uses one of the provided Parser classes and
      configures it to process data content as required to generate alarms
      with a set of properties (or 'columns' or 'fields')
      -->
  </parsers>

</fxmonitor>
```

The following sections describe the F/X Monitor Configuration file definitions and tags in further detail:

- Section 4.5.1, "Adapter Definitions," on page 19
- Section 4.5.2, "Monitor Configuration Setting Definitions," on page 21
- Section 4.5.3, "Reader Definitions," on page 21
- Section 4.5.4, "Parser Definitions," on page 27

## 4.5.1 Adapter Definitions

The first main component of the F/X Monitor configuration file is the `adapters` section, which identifies the target/s for alarms generated by the F/X Monitor.

Specify any number of adapter targets and optionally put them into arbitrary groups that determine how the F/X Monitor responds to failure to deliver alarms to targets.

Adapter targets are defined in the configuration file within an `adapters` tag. Each target definition is contained within an `adapter` tag, as illustrated:

```
<?xml version="1.0"?>
<fxmonitor>
  <adapters>
    <adapter>
    <!-- individual adapter settings go here -->
    </adapter>
  </adapters>
  <!-- remaining fx monitor configuration content goes here -->
</fxmonitor>
```

There are no limitations on the number of adapter definitions.

An excerpt from a configuration file is shown in the following example. It defines an adapter for the F/X Monitor to send alarms to as an F/X adapter on the host mainserver.mycorp.com on port 50001.

```
<adapter>
<name>Main FX Adapter</name>
  <type>PRIMARY</type>
  <group>My Production Group</group>
  <alarmformat>XML</alarmformat>
  <adapterport>50001</adapterport>
  <adapterhost>mainserver.mycorp.com</adapterhost>
  <adapterretry>60</adapterretry>
</adapter>
```

The following list describes the required and optional adapter parameters:

- **<adapterhost>:** (Required) The hostname or IP address of the listening adapter:

  ```
  <adapterhost>hostname_or_IP_address</adapterhost>
  ```

- **<adapterport>:** (Required) The TCP/IP port on which the adapter is listening for connections:

  ```
  <adapterport>adapter_port</adapterport>
  ```

- **<adapterretry>:** (Required) The frequency (in seconds) with which to attempt connection in the event of failure:

  ```
  <adapterretry>retry_frequency</retry>
  ```

- **<alarmformat>:** Specifies the alarm format produced for the adapter. For F/X adapters, use the default XML type:

  ```
  <alarmformat>format_type</alarmformat>
  ```

  where *format_type* is one of the following:

  - **RL:** Reception Log format for legacy universal and TEC adapters.
  - **XML:** FX XML event format (default).

  The value is not case-sensitive.

- **<group>:** The group the adapter definition belongs to:

  ```
  <group>group_name</group>
  ```

  If unspecified, the DEFAULT_GROUP group name is used.

- **<maxwait>:** The maximum time (in seconds) that an alarm waits if it cannot be sent:

  ```
  <maxwait>maximum_wait_time</maxwait>
  ```

  The default is 365 days. If an alarm is not sent to the target within the `<maxwait>` time period, it is deleted from the delivery queue by the F/X Monitor.

- **<name>:** Used only for reference:

  ```
  <name>text_name_of_adapter</name>
  ```

  Specify a text name representing the adapter definition.

- **<type>:** Specifies the operational role of the adapter within its adapter group:

  ```
  <type>operational_role</type>
  ```

  where *operational_role* is one of the following:

  - **primary:** Preferred target within the group.
  - **backup:** Fail-over target within the group.

    Groups can have 1 primary and any number of backup definitions. If more than one primary is specified, the second and subsequent definitions are downgraded to backup by the F/X Monitor.

  - **multicast:** Indicates that this target is in a multicast group and always receives alarms irrespective of the availability of other members of the group.

  The value is not case-sensitive.

### 4.5.2 Monitor Configuration Setting Definitions

There is only one F/X Monitor-level configuration parameter, and it is optional. The `filestore` tag provides the directory location where the F/X Monitor stores temporary files. The default is a subdirectory of the System temporary directory.

```
<settings>
    <filestore>directory_location</filestore>
</settings>
```

### 4.5.3 Reader Definitions

Readers are the definition of a collector to gather file or URI sourced data and a parser to process that data into alarms. The majority of a reader definition consists of the configuration of the chosen collector. The F/X Monitor maintains all readers as separate threads, ensuring that any latency incurred by the collection or parsing process does not impact other collection threads.

To enable parser definitions to be reused across readers, they are referenced by name and defined in a separate `<parsers>` section of the configuration file, as illustrated:

```
<readers>
  <reader>
    <name>name_of_reader</name>
    <class>class_name_of_collector_to_use</class>
    <parser>parser_name_to_use</parser>
    <settings>
      <!-- collector configuration here -->
    </settings>
  </reader>
</readers>
```

An example reader definition is shown in the following example. Note the name, class, and parser definitions and the detail of the collector configuration contained in the `<settings>` tag.

```
<readers>
   <reader>
    <name>Blah News Reader</name>
    <class>com.mosol.integration.fx.collectors.URICollector</class>
    <parser>Generic RSS Parser</parser>
    <settings>
      <!-- Every 60 seconds get the latest RSS news feed -->
      <uri>http://blah.com/feeds/rss.xml</uri>
      <timeout>10</timeout>
      <poll>60</poll>
      <retry>10</retry>
    </settings>
  </reader>
</readers>
```

The following list describes the required parameters for the `<reader>` tag:

- **<name>:** Text name representing the reader:

  `<name>name_of_this_reader</name>`

- **<parser>:** Name of the parser to use with this reader:

  `<parser>name_of_the_parser_to_use</parser>`

  The named parser must be defined in the `<parsers>` section of the configuration file.

- **<class>:** The Java class and package name of the collector to use:

  `<class>full_collector_class_name</class>`

The above tags are children of the `<reader>` tag and are not part of the `<settings>` tag. The `<settings>` tag contains only the configuration for the chosen collector.

The standard collectors delivered are:

- **Directory Collector:** Monitors the directory specified for files matching the file name matching criteria (defined by a regular expression). When new files are written to the directory or files are updated, the contents are read and passed to the associated parser configuration for processing.
- **FTP Collector:** Monitors an FTP server, with optional directory location, for files meeting the specified file matching criteria. When new files are available or files are updated, the file is transferred to a local cache and its contents passed to the associated parser configuration for processing.
- **URI Collector:** Connects to a URI (HTTP or HTTPS) and returns the content for processing by the associated parser. The collector caches the returned content and only passes it to the parser in the event of a content change.
- **Command Collector:** Issues a command to the F/X Monitor host and returns the content from `stdout` and (optionally) `stderr` for the command.

The following defines common collector parameters specified inside the `settings` tag are common for all collectors. The only exception for the required parameters is in the case of user-developed collectors, which do not require a polling model. All of the standard F/X collectors implement polling, thereby requiring these properties.

- **<delay>:** Delay is an optional stagger to defer collection in situations where an F/X Monitor has a large number of collectors, which the administrator does not want to invoke simultaneously:

  ```
  <delay>delay_in_seconds</delay>
  ```

  The default is 0 (zero).
- **<poll>:** (Required) The time period in which the collector polls the source to identify new or updated content:

  ```
  <poll>poll_frequency_in_seconds</poll>
  ```
- **<retry>:** The frequency to attempt connection in the event of failure:

  ```
  <retry>retry_frequency_in_seconds</retry>
  ```

Each collector might add extra configuration settings to a set of standard collector parameters, which are described in the following sections:

- "Directory Collector Parameters" on page 22
- "FTP Collector" on page 24
- "URI Collector" on page 26
- "Command Collector" on page 26

## Directory Collector Parameters

Classname: com.mosol.integration.fx.collectors.DirectoryCollector

The Directory Collector adds a `<directory>` and `<filematch>` setting to the standard set.

The following lists the Directory Collector parameters:

- **<directory>:** (Required) Directory to monitor for file activity:

```
<directory>explicit_directory</directory>
```

- **<encoding>:** Enables the encoding of the file to be specified:

  ```
  <encoding>encoding_type</encoding>
  ```

  Default is UTF-8.

  To specify UNICODE encoding instead of default UTF-8:

  ```
  <encoding>UNICODE</encoding>
  ```

- **<filematch>:** (Required) Regular expression match on the file names to qualify for collection:

  ```
  <filematch>regular_expression</filematch>
  ```

  For example, to collect all `.csv` files:

  ```
  <filematch>.+\.csv</filematch>
  ```

- **<filemaxage>:** Restricts file matching based on the last update time reported by the operating system:

  ```
  <filemaxage type="seconds|minutes|hours|DAYS|months">number_of_units</
  filemaxage>
  ```

  Matched files can be no older than the specified maximum file age, defined in terms of a number of seconds, minutes, hours, days, or months. Case insensitive.

  The default is days. The value must be a positive integer.

  If no `<filemaxage>` tag is specified, then the collector does not differentiate between files based on their date stamp (last modified time).

  Examples:

  To process files less than 12 hours old:

  ```
  <filemaxage type="hours">12</filemaxage>
  ```

  To process files less than 2 months old:

  ```
  <filemaxage type="months">2</filemaxage>
  ```

- **<persist>:** Set to `yes` and the F/X Monitor persists the details of processed files:

  ```
  <persist>yes|NO</persist>
  ```

  If a Monitor stops and restarts, it continues from the last recorded point and does not resend alarms previously delivered.

  This tag applies only to F/X Monitors with session names. Also, progress information is maintained between F/X Monitors with the same name.

  For more information on F/X Monitor startup and naming, see Section 4.2, “Starting an F/X Monitor,” on page 16. Progress information persists only between F/X Monitor sessions of the same name because it is possible for more than one F/X Monitor to match files from the same source, but have differing progress in processing files.

  If the tag is not added or is set to `NO`, persistence does not occur.

  This feature works for both tail and nontail readers across all collector types.

- **<sendfile>:** Instructs the F/X Monitor to send an alarm describing the file, whether content was processed or not:

  ```
  <sendfile>ALL|CHANGES</sendfile>
  ```

where:

  - **ALL:** Send an alarm for each file, irrespective of whether it has changed since the last poll.
  - **CHANGES:** Only send an alarm for new, or changed, files.

  The alarm details the file name, size, and date stamp.

- **<tailfile>:** Set to `yes` to provide the ability to process only recently added content, rather than rescanning complete files:

  ```
  <tailfile>yes|NO</tailfile>
  ```

  If an updated file is a smaller size than the version found during the previous scan, it is assumed that the file has changed fundamentally and the Monitor automatically reprocesses the complete file.

## FTP Collector

Classname: com.mosol.integration.fx.collectors.FTPCollector

The FTP Collector adds a `<directory>` and `<filematch>` setting to the standard set much like the Directory Collector, but also adds FTP server authentication settings.

The following lists the FTP Collector parameters:

- **<directory>:** (Required) Directory to monitor for file activity on the FTP server:

  ```
  <directory>relative_directory</directory>
  ```

  This is a relative directory on the FTP server.

- **<encoding>:** Enables specifying encoding type of the file:

  ```
  <encoding>encoding_type</encoding>
  ```

  The default is UTF-8.

  To specify UNICODE encoding instead of default UTF-8:

  ```
  <encoding>UNICODE</encoding>
  ```

- **<filematch>:** (Required) Regular expression match on the file names to qualify for collection:

  ```
  <filematch>regular_expression</filematch>
  ```

  For example, to collect all `.csv` files;

  ```
  <filematch>.+\.csv</filematch>
  ```

- **<filemaxage>:** Restricts file matching based on the last update time reported by the operating system:

  ```
  <filemaxage type="seconds|minutes|hours|DAYS|months">number_of_units</
  filemaxage>
  ```

  Matched files can be no older than the specified maximum file age, defined in terms of a number of seconds, minutes, hours, days, or months. Case insensitive.

  The default is days. The value must be a positive integer.

  If no `<filemaxage>` tag is specified, then the collector does not differentiate between files based on their date stamp (last modified time).

  Examples:

  To process files less than 12 hours old:

```
<filemaxage type="hours">12</filemaxage>
```

To process files less than 2 months old:

```
<filemaxage type="months">2</filemaxage>
```

- **<filepolicy>:** Determines the policy to be applied to files transferred to the monitor for parsing:

  ```
  <filepolicy [suffix="renamed file suffix"]>LEAVE|DELETEALL|DELETELOCAL|
  DELETEREMOTE|RENAME</filepolicy>
  ```

  The F/X Monitor transfers and caches a local copy of all processed (matched) files.

  The options are:

  - **LEAVE:** Leave transferred files on the FTP server.
  - **DELETEALL:** Delete files on the FTP server and the F/X Monitor's local copy.
  - **DELETELOCAL:** Delete just the F/X Monitor's local copy after parsing.
  - **DELETEREMOTE:** Delete just the FTP server's file.
  - **RENAME:** Rename the file on the FTP server after transferring, appending the suffix value defined in the `filepolicy` suffix argument.

- **<password>:** (Required) Shows FTP password in the clear:

  ```
  <password>clear_password</password>
  ```

  TEA (tiny encryption engine) encrypts the password. For example:

  ```
  <password cipher="TEA">encypted_password</password>
  ```

- **<persist>:** Set to `yes` and the F/X Monitor persists the details of processed files:

  ```
  <persist>yes|NO</persist>
  ```

  If a Monitor stops and restarts, it continues from the last recorded point and does not resend alarms previously delivered.

  This tag applies only to F/X Monitors with session names. Also, progress information is maintained between F/X Monitors with the same name.

  For details on F/X Monitor startup and naming, see Section 4.2, "Starting an F/X Monitor," on page 16. Progress information persists only between F/X Monitor sessions of the same name because it is possible for more than one F/X Monitor to match files from the same source, but have differing progress in processing files.

  If the tag is not added or is set to `NO`, persistence does not occur.

  This feature works for both tail and nontail readers across all collector types.

- **<port>:** (Required) FTP server port:

  ```
  <port>port_number</port>
  ```

  The default is 22

- **<sendfile>:** Instructs the F/X Monitor to send an alarm describing the file, whether content was processed:

  ```
  <sendfile>ALL|CHANGES</sendfile>
  ```

  where:

  - **ALL:** Send an alarm for each file, irrespective of whether it has changed since the last poll.
  - **CHANGES:** Only send an alarm for new, or changed, files.

  The alarm details the file name, size, and date stamp.

- **<server>:** (Required) Hostname or IP address of the FTP server:

  `<server>`*`hostname_or_IP_address_of_FTP_Server`*`</server>`

- **<user>:** (Required) FTP user name:

  `<user>`*`relative_directory`*`</user>`

## URI Collector

The URI Collector takes a URI and uses it to collect content for parsing.

The following lists the URI Collector parameters:

- **<uri>:** (Required) URI, such as a Web URL:

  `<uri>`*`URL`*`</uri>`

  For example, to recover the CNN RSS feed:

  ```
  <uri>http://rss.cnn.com/rss/cnn_topstories.rss</uri>
  ```

- **<timeout>:** Timeout value for connection attempts:

  `<timeout>`*`timeout_in_seconds`*`</timeout>`

## Command Collector

(Required) The Command Collector takes a command string, which the F/X Monitor executes each poll period, returning the `stdout` and `stderr` content for parsing. The Command Collector also inherits the standard collector properties.

The following is the syntax for the Command Collector parameter:

```
<command [timeout=timeout_seconds] [includestderr="NO|yes"]>command_to_execute</
command>
```

This tag defines the command to run and optionally provides a timeout value. If the command has not returned (completed) by the timeout period, the invoked process is terminated by the F/X Monitor.

On a Windows system, this example lists the users, but terminate if the command takes more than 5 seconds to complete:

```
<command timeout=5>cmd /c net user</command>
```

You can specify that any `stderr` output of the process also be captured for processing using the `includestderr` argument. On a UNIX system, this example lists the running processes that contain `mos` and capture `stderr`:

```
<command timeout=10 includestderr="yes">ps -ef | grep mos*</command>
```

## 4.5.4 Parser Definitions

As previously described, parsers are utilized by defining parser configurations. Parser configurations are defined with a name for reference. They identify the parser class to use, then supply the configuration for that parser class.

In a similar manner to collectors, parsers inherit common configuration parameters and can extend them further to the processing type. This document describes the standard (common) parameters, then details the additional parameters available with each Operations Center parser.

Parser definitions appear in the F/X configuration file within the `<parsers>` section, in the following format:

```
<parsers>
  <parser>
    <name>name of parser</name>
    <class>class name of parser to use</class>
      <settings>
      <!-- parser configuration here -->
      </settings>
  </parser>
</parsers>
```

Two parameters are required beneath the `<parser>` tag:

- **<name>:** Text name representing the parser:

  `<name>`*`parser_name`*`</name>`

- **<class>:** The java class and package name of the parser to use:

  `<class>`*`full_parser_class_name`*`</class>`

The parser specific configuration parameters are contained within the `<settings>` tag.

For all of the Operations Center supplied parsers, the class name to use is:

`com.mosol.integration.fx.parsers.`*`parser_class`*

The following describes the standard class parsers:

- **DelimitedFileParser:** Processes delimited content. The capabilities include:
  - Definition of the delimiter to use
  - Automatic property naming using `headerline`
  - Handles nested delimited content
- **CSVParser:** A specialization of the Delimited File Parser for comma-separated files.
- **TabDelimiterParser:** A specialization of the Delimited File Parser for tab-delimited files.
- **XMLParser:** Processes XML datasets. Its capabilities include:
  - XPath-based identification of sections of interest for alarm generation
  - Relative addressing (incorporation) of parent and child tag values and attribute values
  - Automated conversion of date and numeric formats
  - Date conversions to specify alarm last updated time from content
- **RegExpParser:** Applies regular expression grouping by file line. Enables splitting of files into alarms with properties. Severity matching is based on content matching. It specifies the alarm last updated time using the content.
- **SimpleTextParser:** Performs line by line processing of text files, using content match to determine which lines are promoted as alarms.

The following topics cover using parsers:

- "Standard Parser Configuration Parameters" on page 28
- "More about Severity Rule Tags" on page 32
- "Multiple Severity Rules" on page 33
- "Delimited Parsers (DelimitedFileParser, TabDelimiterParser, and CSVParser)" on page 34
- "XML Parser" on page 34
- "Regular Expression Parser" on page 38
- "Simple Text Parser" on page 39

## Standard Parser Configuration Parameters

Since the original release of the F/X adapter, a great effort was made to make available to all parsers a standard set of parsing configuration parameters. The result is the following standard configuration parameters that apply to dataset processing using any parser:

- **<cleandata>:** Enables cleaning of column data by providing a regular expression that identifies a group to extract and set as the column value:

  ```
  <cleandata column="column_to_clean">matcher</cleandata>
  ```

  If the matcher does not match the column value, no changes are made. This also enables multiple `<cleandata>` tags to be defined for the same column, if required.

  For example, to clean the *TicketID* column by removing the `TKT` prefix from the ticket number:

  ```
  <cleandata column="TicketID">^TKT([0-9]+).*</cleandata>
  ```

  Assume the sample column value is `TKT12345`. Then, after processing by the `<cleandata>` tag, the value of the *TicketID* column is `12345`.

- **<columnaliases>:** Replaces column names (if found). Generally, `<columnaliases>` is used when automatic headers are found, such as with `.csv` files:

  ```
  <columnaliases>comma-separated_list_of_alias_definitions_column=alias_name</
  columnaliases>
  ```

  For example, to replace the original column named *Bldg* with *Building* and replace the column named *Usr* with *User*:

  ```
  <columnaliases>Bldg=Building,Usr=User</columnaliases>
  ```

- **<datecolumns> and <dateformats>:** Used in conjunction, a comma-separated list of  columns containing dates and the corresponding date format strings:

  ```
  <datecolumns>comma-separated_list_of_columns</datecolumns>
  ```

  `<datecolumns>` is a comma-separated list of columns containing date or time stamp data.

  ```
  <dateformats>yyyy-MM-dd HH:mm:ss,yyyy-MM-dd HH:mm:ss</dateformats>
  ```

  `<dateformats>` is a comma-separated list of date formats that describes how to convert the date string into a real date for processing. It uses the Java `SimpleDateFormat` syntax.

  The contents of the columns are parsed using the corresponding `<dateformat>` entry.  For all datecolumns successfully parsed, the column value is replaced by the long representation of the date/time and a new column is added using the name `columnname.original` with the original content.

It is assumed that the `<datecolumns>` and `<dateformats>` references are listed in the same order. If a `<dateformats>` entry is not defined for a given column, the default local date/time parsing format is used.

For example, two columns named *Timestamp* and *DateField* are available in a row. List *Timestamp* first to indicate that its value is the one to use as the resultant alarms date/time.

```
<datecolumns>Timestamp,DateField</datecolumns>
```

You also must supply the corresponding format definitions in a `<dateformats>` tag. If the sample data is:

```
2006-08-31 13:32:02
```

This corresponds to a date format string of:

yyyy-MM-dd HH:mm:ss

- **<lifetime>:** Specifies the TTL (time to live) for an alarm when it arrives in the Operations Center server:

  ```
  <lifetime type="mins|hours|DAYS|weeks|months" relative="create|DATE">amount</
  lifetime>
  ```

  Type defines the time measurement. Amount defines the number of time units. The default type is DAYS.

  relative specifies whether the alarm lifetime should be relative to the time the alarm was received and created by the adapter (CREATE) or to the date stamp of the alarm (DATE).

  For more details on the implementation of alarm lifetime, see Chapter 5, “Alarm Lifetime,” on page 41.

  An example of defining a 5-day TTL for alarms:

  ```
  <lifetime>5</lifetime>
  ```

  A 2-hour lifetime based on when the alarms arrive at the adapter:

  ```
  <lifetime type="hours">2</lifetime>
  ```

- **<multiplex>:** Columns containing comma-separated values can be identified by the `<multiplex>` tag:

  ```
  <multiplex [delimiter="delimiter_string"]>comma-separated_list_of_columns</
  multiplex>
  ```

  An alarm is generated for each value in the multiplex column.

  Mulitplexing can make hierarchy processing easier when columns are known to contain multiple values, which should be treated distinctly.

  For example, to identify the *Departments* column as a multiplex column:

  ```
  <multiplex>Departments</multiplex>
  ```

  During content processing, if the value `Finance,HR` is found for the *Departments* column, the F/X Monitor generates two alarms for the row. The first alarm contains `Department=Finance`. The second alarm contains `Department=HR`.

  If the content of the multiplex columns is not comma-separated the alternative delimiter to be used can be specified using the `delimiter` argument. This delimiter applies to the *column data* and not the list of multiplex columns themselves, which are always comma-separated!

  For example, to specify that columns *Service Groups* and *Support Centers* are multiplex columns and their content is separated by the # character, the following definition should be used:

```
<multiplex delimiter="#">Service Groups,Support Centers</multiplex>
```

- **<replacements>:** Selectively copies column values based on content matching.:

  ```
  <replacements>

  <copycol fromcolumn="column_name" tocolumn="column_name">matcher</copycol>

  …

  </replacements>
  ```

  The `<replacements>` tag must contain the `<copycol>` tag.

  Matching can be empty, which implies that the copy should always occur.

  For example, to copy the *TicketID* column to a new column named *CheckedTicketID* if the content begins with the text `TKT` and is followed by at least one number:

  ```
  <replacements>

  <copycol fromcolumn="TicketID" tocolumn="CheckedTicketID">^TKT[0-9]+$</
  copycol>

  </replacements>
  ```

- **<severities>:** The column values can be matched against severity definitions to provide the generated alarms with severity status:

  ```
  <severities>

  <default>UNKNOWN</default>

  <severity column="Result" condition="OK">pass</severity>

  <severity column="Result" condition="CRITICAL">fail</severity>

  <severity column="Business_Risk" condition="MINOR">PRODUCTION

  <sevRule column="Estimated_Down_Time" condition="MINOR" equation="gt">0</
  sevRule>

  <sevRule column="Estimated_Down_Time" condition="MAJOR" equation="gt">2</
  sevRule>

  </severity>

  <severity column="Business_Risk" condition="INFO">Medium

  <sevRule equation="script">

  var returnType=true;

  return returnType;

  </sevRule>

  </severity>

  </severities>
  ```

  The `<severity>` tag can contain the `<default>` tag and multiple embedded `<severity>` tags, as well as `<severity>` tags with embedded `<severity rules>` tags.

  Valid severities are OK, UNKNOWN, MINOR, MAJOR, and CRITICAL.

  Any number of severities can be defined and an optional default severity can be set.

  In the case where more than one severity definition matches, the highest severity match is set as the alarm severity.

CLEAR is a special delete severity, usually used to close deduplicated alarms generated by use of the `uniquecolumns` parameter.

For an additional explanation and usage examples, see "More about Severity Rule Tags" on page 32.

- **<skipcolumns>:** (Optional) Skips columns based on the column content:

  ```
  <skipcolumns>

  <skip column="column_name">matcher</skip>

  …

  </skipcolumns>
  ```

  The `<skipcolumns>` tag must contain at least one `<skip>` tag.

  Skipping columns is optionally qualified with a regular expression match. If no regular expression match is defined, then the column is always removed, regardless of content.

  If a column's content is matched, the line is skipped. It is legal to have multiple skip definitions for the same column.

  For example, to skip the *ID* column when it contains a number:

  ```
  <skipcolumns>

  <skip column="ID">[0-9]+</skip>

  </skipcolumns>
  ```

- **<skiprows>:** Skips rows based on matching column values:

  ```
  <skiprows>

  <skip column="column_name" [invert="yes"|"no"]>matcher</skip>

  …

  </skiprows>
  ```

  The `<skiprows>` tag must contain at least one `<skip>` tag.

  Column values can also be marked as inverted, to skip a column if the column value does not match a specified value. This can be useful for enforcing data structure or content.

  For example, to define a policy where rows are skipped if the *Status* field contains Deleted, or if the *Hostname* field contains the word Test:

  ```
  <skiprows>

  <skip column="Status">Deleted</skip>

  <skip column="Hostname">.*Test.*</skip>

  </skiprows>
  ```

- **<splits>:** Split definitions enable separating complex content in one column into subcomponents in new columns:

  ```
  <splits>

  <split fromcolumn="column_containing_data_to_split" tocolumns="comma-
  separated_list_of_columns_to_create"
  name="name_of_split">regular_expression_group_pattern</split>

  …

  </splits>
  ```

`<splits>` must contain `<split>`.

The `<splits>` tag can contain any number of split definitions, but must contain at least one. Split definitions consist of the originating column, the list of columns to split into and the regular expression that defines the splits.

For example, to split a name into *firstname* and *lastname* components:

```
<splits>

<split fromcolumn="name" tocolumns="firstname,lastname" name="Name
Splitter">(.*) (.*)</split>

</splits>
```

If data contained in the *name* column contains the value `Joe Bloggs`, the split definition creates two new columns: `firstname = Joe` and `lastname = Bloggs`.

Splits can contain any number of components (elements extracted, which are encapsulated in parentheses).

- **<uniquecolumns>:** Identifies data from one or more columns that collectively constitutes a unique identifier for the data:

  ```
  <uniquecolumns>list_of_columns_that_determine_uniqueness</unique>
  ```

  This enables differentiating alarm content between being a new alarm and an update to a previous alarm at the adapter level.

  For example, if the column's hostname and IP address collectively identify a unique entity, then the adapter treats alarms that have identical hostnames and IP address properties as updates to the same alarm in Operations Center:

  ```
  <uniquecolumns>Hostname,IPAddress</uniquecolumns>
  ```

  For more detailed information on alarm uniqueness and deduplication, see Chapter 6, "Alarm Unique Key / Deduplication Capability," on page 43.

  It is possible to list all alarm columns, which can cause an overloaded list of columns. The adapter then generates the key for the alarm based on columns available per alarm.

## More about Severity Rule Tags

There are two levels of capability for using the severity rules tags within the F/X parser. The first level is the simple operator tag, allows you to compare an alarm column value with a fixed value (called the pattern). There are four operators:

- **gt:** greater than
- **lt:** less than
- **eq:** equals
- **gt:** greater than

If the result of the equation is `True`, then the severity is set to a specific value. If the value is `False`, then the normal severity value from the parent tag is used.

In the following example, if the column named *Business_Risk* contains the text `PRODUCTION`, then set the alarm condition to `MINOR`, unless the column *Estimated_Down_Time* has a numeric value greater than 0, in which case, set the alarm condition to `CRITICAL`.

```
  <severities>
    <severity column="Business_Risk" condition="MINOR">PRODUCTION
       <sevRule column="Estimated_Down_Time" condition="CRITICAL" equation="gt">0</
sevRule>
    </severity>
 </severities>
```

These operators can only evaluate against numeric values only. A warning message displays if the column value used for comparison is not numeric. An error occurs if you set the value text of this tag to a nonnumeric character, unless you use the script equation described in the following example.

The tag does not operate against strings, nor does it perform any sorting of string comparisons. This is already performed throughout the adapter's configuration when building the alarm stream and processing raw data.

However, complex string comparisons still can be performed at this point. Set the “equation” attribute equal to “script”. This enables a developer to create a script fragment (in NOC Script) to process any column within the alarm in any way and make a call to the `setSeverity(`*`severity`*`)` method. An example:

```
<severities>
   <severity column="Business_Risk" condition="MINOR">PRODUCTION
      <sevRule equation="script">
         if (alarm.get("Node_Center_Name") == “Dallas”) {
            fxAdapter.setSeverity("MAJOR");
         }
      </sevRule>
   </severity>
</severities>
```

## Multiple Severity Rules

It is possible to use more than one severity rule with a severity tag. In this case, the last severity rule in the list of rules that returns True is used to set the severity. Here is an example:

```
<severities>
    <severity column="Business_Risk" condition="OK">PRODUCTION
       <sevRule column="Estimated_Down_Time" condition="INFO" equation="lt">3.5</
sevRule>
       <sevRule column="Estimated_Up_Time" condition="MINOR" equation="gt">100</
sevRule>
       <sevRule column="Standard_Mean_Time" condition="MAJOR" equation="eq">333</
sevRule>
      <sevRule column="emergency" condition="CRITICAL" equation="neq">-1</sevRule>
    </severity>
 </severities>
```

In this case, if the first, second, and third rules are all triggered, the third rule is used and the severity is set to MAJOR.

Script type rules can be combined with equation style rules.

## Delimited Parsers (DelimitedFileParser, TabDelimiterParser, and CSVParser)

All the delimited type parsers share a common configuration parameter set in addition to the standard configuration parameters already described:

- **<columns>:** List of column names to use for the delimited content:

  `<columns>`*`comma-separated_list_of_columns`*`</columns>`

  If the main content has more columns than are defined in this tag, the remaining columns are omitted from generated alarms.

  If a `<headerrow>` tag is defined and found by the parser, the results are used in preference to the values of the `<columns>` tag. As such, the `<columns>` tag can be considered as a fail-safe.

  For example, to identify 4 column names:

  ```
  <columns>Date,Building,Name,Value</columns>
  ```

- **<delimiter>:** (Required) Character or string used to delimit content:

  `<delimiter>`*`delimiter_marker`*`</delimiter>`

  For TabDelimiterParser and CSVParser, this is already defined.

  For example, use `<delimiter>|</delimiter>` for content delimited by the **|** (pipe) character.

- **<headerline>:** Identifies the line that contains column names:

  `<headerline>`*`line_number`*`</headerline>`

  The Parser expects the column delimiter to be the same as the delimiter used for the main content.

  For example, to define the first line of a file as the `<headerline>`:

  ```
  <headerline>0</headerline>
  ```

- **<startline>:** The line from which the parser should start to generate alarm content:

  `<startline>`*`line_number`*`</startline>`

  If a `<headerline>` is defined, the default is set to line 1.

  For example, to set the start line to 2 (the third line of a file):

  ```
  <startline>2</startline>
  ```

## XML Parser

(Required) The XML Parser works by identifying target elements within an XML document to reference for the overall parsing requirements. The target elements are identified using XMLPath notation (see http://www.w3.org/TR/xpath (http://www.w3.org/TR/xpath) for more details and examples of use). XPath syntax enables target elements to be identified by referencing the XML document structure and, optionally, conditional matching of element values and attributes.

Associated processing is encapsulated within containers known as filters. The XMLParser automatically adds all attributes of the matched element as alarm properties. The configuration for each filter can contain any number of XMLParser parameters that enable the incorporation of additional tag values and attribute values into the alarm.

The following lists the required and optional XML Parser tags:

- **<filter>:** (Required) Filter definition matches XML content and generates an alarm from each match:

  ```
  <filters>

  <filter name="filter_name" path="xpath_definition">

  …

  </filter>

  </filters>
  ```

  where:

  - **Name:** A free text name given to the filter that is bound to any generated alarms using the ALARM_XMLFILTERNAME property.
  - **Path:** An XPath definition.

  The `<filters>` tag must contain `<filter>` tags.

  The alarm is automatically populated with the matching node's value and attributes. Additional configuration parameters determine other properties to add to the alarm.

  The following example illustrates the structure of an XMLParser tag, in this case containing two filters. Each filter can match more than one element in the XML document. Each match generates a single alarm. The (optional) parser parameters add properties to the alarm.

  ```
  <parsers>
  <parser>
  <name>PSO Parser</name>
      <class>com.mosol.integration.fx.parsers.XMLParser</class>
      <settings>
        <filters>
          <filter name="PSO Node Numeric Info" path="PerformanceMonitor/Node//
  PerfNumericInfo">
          <parentname levelsup="1" tofield="Metric" />
          <parentname levelsup="2" tofield="MetricGroup" />
          <parentattribute node="Server" attribute="name" tofield="ServerName" />
            <parentattribute node="Node" attribute="name" tofield="NodeName" />
            <convert field="time" type="DATE" inputformat="long"
  outputformat="ddMMyyyy hh:mm:ss" tofield="recdate"/>
            <convert field="val" type="NUMERIC" />
          </filter>
        </filters>
      </settings>
  </parser>
  </parsers>
  ```

  To omit data in long XML strings, such as those generated by RSS feeds, use the double-backslash to exclude tags. For example, assume there are four tags that an XPATH_definition can parse: country, state, city and address. To include only the state, city and address in the Operations Center events/alarms, use the syntax:

  ```
  path="//state/city/address"
  ```

  where the double backslash represents a parent tag that should not be included in the parsed data.

- **<childtag>:** Identifies which child tags to incorporate into the alarm:

  ```
  <childtag>matcher</childtag>
  ```

  Stores both the child tag value and all attributes.

The matcher is a regular expression identifying which child tag (names) to match.

For example, to match all child tags:

```
<childtag>.*</childtag>
```

To match only the `Person` or `Resource` child tags:

```
<childtag>(Person|Resource)</childtag>
```

- **<convert>:** Enables the conversion of date or numeric fields from the XML String format into a native format and conversion to an alternative format, if required:

```
<convert field="field_to_convert" type="DATE|NUMERIC"
[inputformat="input_format_type" outputformat="output_format_type"]
[tofield="field_to_output_result_to"] />
```

The converted value can be stored in the original field (default) or in an alternative field using the `tofield` argument.

If the type is NUMERIC, then only the field must be identified.

For example, if the source data is:

```
<myval>1.242E+08</myval>
```

Use the following to convert `myval` to a numeric and store in the new field *numeric_myval*:

```
<convert type="NUMERIC" field="myval" tofield="numeric_myval" />
```

If type is DATE, then define the input and output formats using the `SimpleDateFormat` syntax.

As a special case, if type is DATE and the source data is a long representation of a date (such as the number of seconds since 1 January 1970), the value long in the `inputformat` argument instructs the F/X Monitor to convert from a long format.

For example, assume the sample data is:

```
<adate>2006-08-31 13:32:02</adate>
```

This corresponds to the date format string:

yyyy-MM-dd HH:mm:ss

Assume the requirement is to just show the date in MM/dd/yyyy form. The corresponding `<convert ...>` tag is:

```
<convert field="a_date" type="DATE" inputformat="yyyy-MM-dd HH:mm:ss"
outputformat="MM/dd/yyyy" tofield="the_day" />
```

- **<parentattribute>:** Stores the value of the specified attribute for the named parent node in the specified field name:

```
<parentattribute node="name_of_the_parent_tag"
attribute="attribute_name_to_collect" tofield="field_name" />
```

For example, consider the same source XML data again, but this time collect the value of the state attribute of the `<Location>` tag:

```
<Location state="LA">

<Office name="myoffice"/>

</Location>
```

Use this attribute:

```
<parentattribute node="Location" tofield="State" />
```

This results in adding the following name/value pair to the alarm:

```
State=LA
```

- **<parentname>:** Stores the name of the parent tag in the specified field name:

  ```
  <parentname levelsup="number_of_levels_up" tofield="field_name" />
  ```

  The number of levels ‘up’ is relative to the matched element.

  For example, consider the following source XML data:

  ```
  <Location state="LA">

  <Office name="myoffice"/>

  </Location>
  ```

  Assume the filter matched the *Office* element. To store the tag name of its immediate parent in a field named *Folder Type*:

  ```
  <parentname levelsup="1" tofield="Folder Type" />
  ```

  This results in adding the following name/value pair to the alarm:

  ```
  Folder Type=Location
  ```

- **<pathattribute>:** Enables use of XPath to identify attribute values from the XML document to be added as properties to the alarm:

  ```
  <pathattribute path="xpath_definition" [scope="document|FILTER]"
  attributes="list_of_attributes_to_collect"
  [tocolumns=l"ist_of_column_names_to_use_for_the attributes"] />
  ```

  If the provided XPath expression matches multiple nodes, all of the matched attributes are collected. The attributes are distinguished by the `.n` suffix added to the attribute name in the alarm.

  The XPath expression is processed against the matched element from the filter by default, unless scope is set to document, in which case the scope of the expression is the complete document. Normally the filter (default) scope is required.

  An attribute name or comma-separated list of attributes must be specified.

  (Optional) The `tocolumns` attribute can be used to provide alternative column names for the attributes (in cases where the original XML attribute names were not desirable).

  Examples:

  - To find the attribute `ID` from the child tag `ticket`:

    ```
    <pathattribute path="//ticket" attributes="ID" />
    ```

  - To collect all the user ID attributes of user tags and use the column name *username* substitution:

    ```
    <pathattribute path="some_info/user" attributes="userid"
    tocolumns="username" />
    ```

- **<pathvalue>:** Enables use of XPath to identify tag values (and the tag’s attribute) from the XML document to be added as properties to the alarm:

  ```
  <pathvalue path="xpath_definition" [scope="document|FILTER]"
  includeattributes="yes|NO" showempty="yes|NO" [tocolumn="column_name"] />
  ```

  If the provided XPath expression matches multiple nodes, all of the matched tags are collected. The attributes are distinguished by a `.n` suffix being added to the tag’s column name in the alarm.

The XPath expression is processed against the matched element from the filter by default, unless scope is set to "Document," in which case the scope of the expression is the complete document. Normally the filter (default) scope is required.

The `includeattrs` attribute specifies whether a matched node's attributes should be included as alarm properties. The default is no. If attributes are included they are named as:

```
<columnname>.<attributename>
```

The `showempty` attribute specifies whether to include the node in the alarm is it's value is null or empty. Default behavior is not to add the property in this case.

`tocolumn` identifies the required column name for the tag value, if this is not specified the tag name is used.

For example, to gather the value of leaf node `address`:

```
<pathvalue path="//address "/>
```

## Regular Expression Parser

(Required) The Regular Expression parser is intended for use with nondelimited file content where regular expression matching provides a convenient way to filter content into alarms. The syntax for the `<expressions>` tag is:

```
<expressions>

<expression name=" columns="comma-separated_list_of_column_names">matcher</
expression>

</expressions>
```

The `<expressions>` tag must contain `<expression>` tags.

Each expression component is matched against a line of the file. If the match passes, then columns are synthesized and the values of the columns are set to the corresponding groups in the match definition.

The first matching expression is applied to a given line. Subsequent expressions are ignored for that line.

For convenience, the name of the matching expression is also added to the alarm as a property named `ALARM_MATCHERNAME`.

For example, consider a line that has the following format:

```
140502 SERVER Failed The Server Failed
```

This can be broken into columns by the following expression definition:

```
<expressions>

<expression name="My Expression 1" columns="time,component,status,message">
([0-9]{6}) (.*) (.*) (.*)</expression>

</expressions>
```

As with any regular expression, be sure to escape special characters with a backslash. In addition, the following special characters in XML must be replaced with the HTML character entity as follows:

- For ampersands (&), use `&amp;`
- For less-than signs (<), use `&lt;`
- For greater-than signs (>), use `&gt;`

- For quotation marks (“), use `&quot;`
- For apostrophes (‘), use `&apos;`

## Simple Text Parser

The Simple Text parser, as the name implies, has a simple approach to parsing content. Line by line files are processed by comparing the content, to filter in and out expressions that determine which lines constitute alarm content:

- **<severities>:** Lines can be matched against severity definitions to provide the generated alarms with severity status:

  ```
  <severities>
  [<default>condition_as_default</default>]
  <severity condition="condition_to_set">matcher</severity>
  …
  </severities>
  ```

  The `<severities>` tag can contain the `<default>` and `<severity>` tags.

  Any number of severities can be defined and an optional default severity can also be set.

  In the case where more than one severity definition matches, the highest severity match is set as the alarm severity.

  Valid severities are OK, UNKNOWN, MINOR, MAJOR, and CRITICAL.

- **<filterin>:** Lines are candidates for alarms only if they are filtered in by one or more of the in filter definitions:

  ```
  <filterin>
  <in>matcher</in>
  …
  </filterin>
  ```

  The `<filterin>` tag must contain the `<in>` tag.

  For example, to filter in all lines:

  ```
  <filterin>
  <in>.*</in>
  </filterin>
  ```

- **<filterout>:** Lines can be filtered out (or, not promoted as alarms) by matching an out filter definition:

  ```
  <filterout>
  <out>matcher</out>
  …
  </filterout>
  ```

  The `<filterout>` tag must contain the `<out>` tag.

  Only alarms previously filtered by `filterin` statements are candidates for filtering out.

For example, to filter out lines beginning with the word TEST:

```
<filterout>

<out>TEST.*</out>

</filterout>
```

# 5 Alarm Lifetime

Use the `lifetime` parameter to automatically age out alarms parsers. This optional parameter defines the length of time (in minutes, hours, days, weeks, or months) that an alarm can live. Upon receipt of an alarm, the F/X adapter calculates the expiration date, then closes it automatically.

It is possible to configure the expiration time two ways:

- Based on the date stamp of the alarm (which F/X can configure to relate to any alarm field). This is the default behavior.
- Based on the time when the adapter receives the alarm.

The syntax of the `<lifetime>` tag is shown in the following example where the options are separated by the pipe character (|) and the defaults are denoted by parentheses:

```
<lifetime type="MINS|HOURS|(DAYS)|WEEKS|MONTHS" relative="CREATE|(DATE)">Amount</
lifetime>
```

where:

- **type:** The time interval.
- **relative:** Specifies whether the alarm lifetime should be relative to the time the alarm was received and created by the adapter (CREATE) or to the date stamp of the alarm (DATE).
- ***Amount*:** This value is the number of date periods to calculate.

If the type or relative arguments are omitted, the defaults are used.

For example, to specify that alarms should have a lifetime of 30 minutes from the specified date stamp in the alarm, use the following syntax:

```
<lifetime type="mins">30</lifetime>
```

To specify that alarms should be persisted for five days from the point of receipt, use the following syntax:

```
<lifetime type="days" relative="create">5</lifetime>
```

The `<lifetime>` tag is contained within the `<settings>` tag of any parser. For example:

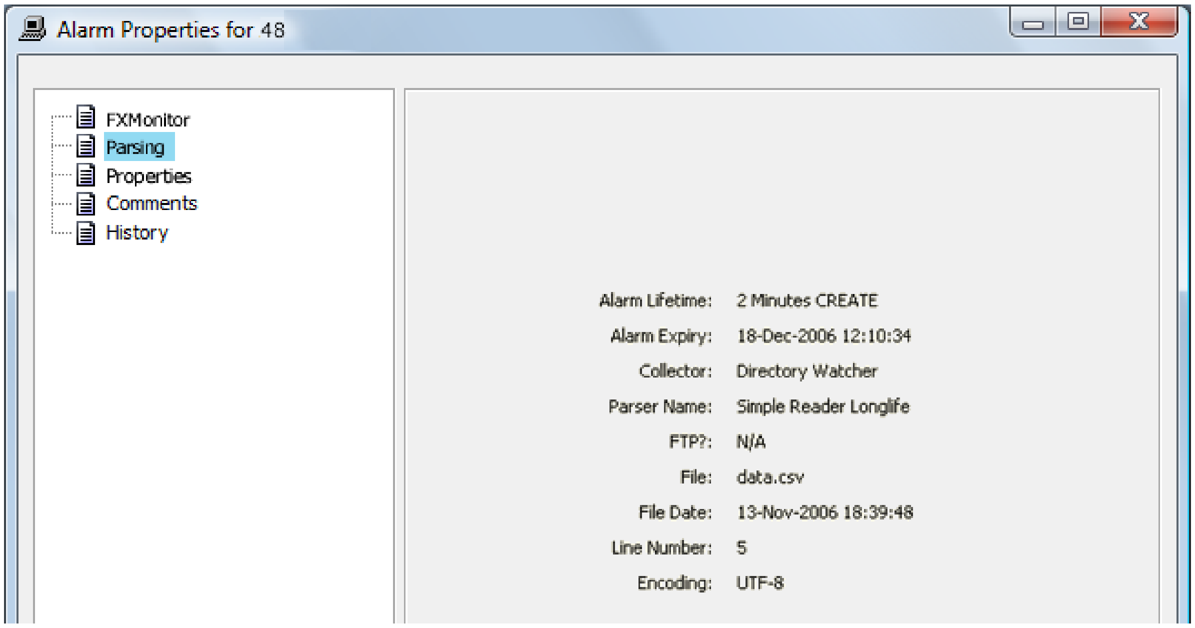
```
<parser>
<name>Simple Parser with Lifetime</name>
    <class>com.mosol.integration.fx.parsers.CSVParser</class>
    <settings>
    <lifetime type="days" relative="create">5</lifetime>
    <headerline>0</headerline>
    <startline>1</startline>
    <severities>
      <default>OK</default>
      <severity column="message" condition="CRITICAL">.*error.*</severity>
    </severities>
  </settings>
</parser>
```

To enable an operator to validate operations, two new properties are exposed in the Alarm Properties dialog box in Operations Center. The properties show the defined lifetime for an alarm and the calculated expiry time of that alarm.

The alarm lifetime capability is recognized by the adapter when in alarm persistence mode. All persisted alarms that expire during adapter downtime are not re-created when the adapter restarts.

Figure 5-1 illustrates Alarm Lifetime and Alarm Expiry showing the defined lifetime for an alarm and the calculated expiry time:

***Figure 5-1*** *Alarm Properties Parsing Tab*



| | |
|---|---|
| Alarm Lifetime: | 2 Minutes CREATE |
| Alarm Expiry: | 18-Dec-2006 12:10:34 |
| Collector: | Directory Watcher |
| Parser Name: | Simple Reader Longlife |
| FTP?: | N/A |
| File: | data.csv |
| File Date: | 13-Nov-2006 18:39:48 |
| Line Number: | 5 |
| Encoding: | UTF-8 |

# 6 Alarm Unique Key / Deduplication Capability

The concept of a unique alarm key enables the F/X adapter to deduplicate alarms from F/X Monitors based on alarm content. If used, this instruction defines a column (or columns), which collectively contain data that identify a unique alarm. For example, an alarm field might contain a Trouble Ticket ID, and the desired behavior is that alarms that share the same Trouble Ticket ID are updates to the original alarm, rather than separate alarms.

To identify the columns that mark an alarm as unique, add a `<uniquecolumns>` tag to the parser. The content of this tag should be an alarm column name or a comma-separated list of alarm columns.

For example, to define the column *TicketID* as a unique column:

```
<uniquecolumns>TicketID</uniquecolumns>
```

In the following example, two columns are used to form the key. In this case, the columns are the first name and surname:

```
<parser>
      <name>People Parser</name>
  <class>com.mosol.integration.fx.parsers.CSVParser</class>
  <settings>
    <uniquecolumns>FirstName,Surname</uniquecolumns>
    <severities>
      <severity column="Mood" condition="OK">HAPPY</severity>
      <severity column="Mood" condition="CRITICAL">(MISERABLE|SAD)</severity>
      <severity column="Mood" condition="MINOR">ANNOYED</severity>
      <severity column="Mood" condition="MAJOR">ANGRY</severity>
     <severity column="Mood" condition="CLEAR">DEAD</severity>
   </severities>
    <headerline>0</headerline>
    <startline>1</startline>
  </settings>
</parser>
```

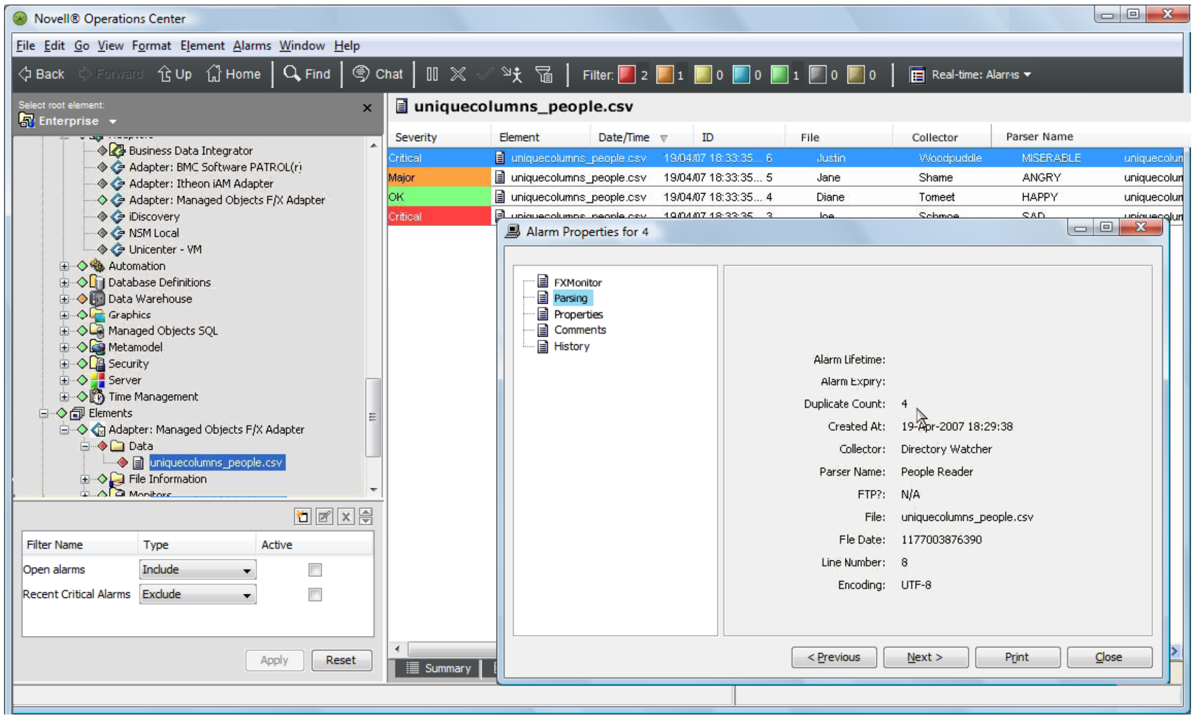
In the example above, the CLEAR severity used in the parser definition denotes that the alarm should be closed (deleted).

When an alarm is received by the F/X adapter, where unique columns have been specified, the adapter calculates a unique numeric key based on the content of the unique column fields. This key determines if the F/X alarm is a new alarm in the adapter or if it is an update to an existing alarm. Where an update is encountered, the alarm properties are compared and changes are applied. If new columns appear in the new alarm, they are added to the alarm on the adapter side.

For any alarm update, the duplicate count and last update times are always updated to reflect the new alarm. If the F/X alarm is a CLOSE alarm, then the alarm is deleted on the adapter side.

Figure 6-1 shows some duplicated alarms. Note the duplicate count value in the Alarm Properties dialog box, indicating that four alarms so far have been deduplicated:

***Figure 6-1** Operations Center Console*



uniquecolumns_people.csv

| Severity | Element | Date/Time | ID | File | Collector | Parser Name | |
|---|---|---|---|---|---|---|---|
| Critical | uniquecolumns_people.csv | 19/04/07 18:33:35... | 6 | Justin | Woodpuddle | MISERABLE | uniquecolun |
| Major | uniquecolumns_people.csv | 19/04/07 18:33:35... | 5 | Jane | Shame | ANGRY | uniquecolun |
| OK | uniquecolumns_people.csv | 19/04/07 18:33:35... | 4 | Diane | Tomeet | HAPPY | uniquecolun |
| Critical | uniquecolumns_people.csv | 19/04/07 18:33:35... | 3 | Joe | Schmoe | SAD | uniquecolun |

| Filter Name | Type | Active |
|---|---|---|
| Open alarms | Include | ☐ |
| Recent Critical Alarms | Exclude | ☐ |

| | |
|---|---|
| Alarm Lifetime: | |
| Alarm Expiry: | |
| Duplicate Count: | 4 |
| Created At: | 19-Apr-2007 18:29:38 |
| Collector: | Directory Watcher |
| Parser Name: | People Reader |
| FTP?: | N/A |
| File: | uniquecolumns_people.csv |
| Fle Date: | 1177003876390 |
| Line Number: | 8 |
| Encoding: | UTF-8 |

# A F/X Monitor Return Codes

The F/X Monitor returns a full set of return codes, returned as an integer. Table A-1 lists the close-related return codes.

***Table A-1*** *Monitor Return Codes*

| Code | Meaning |
|---|---|
| 0 | OK (success). |
| -1 | Cannot create signal file. |
| -2 | No monitor name provided. |
| -3 | Monitor not running. |
| -4 | Unable to signal. |
| -10 | Insufficient create arguments. |
| -12 | Configuration is empty. |
| -13 | Logging configuration is empty. |
| -14 | Monitor name already exists (running). |
| -15 | Cannot start Monitor. |

# B F/X Adapter Alarm Properties

The F/X Adapter generates alarms in Operations Center that have some standard properties in addition to the properties generated as a result of the parsing configuration and data content. These standard properties are listed here to enable clarification over what alarm properties are available as standard for use within the MODL™ Hierarchy File and/or script processing.

Following are the alarm properties:

| Alarm Property | Meaning |
|---|---|
| ALARM_TYPE | Type of alarm. Possible values:<br>**DATA:** The most common type, representing real data from a source.<br>**STATUS:** F/X Monitor generated status alarm reporting the health of the monitor or reader polls.<br>**FILEINFO:** File information alarm, details files processed, size, date stamp, and so on. These alarms are generated based on the `<sendfile>` setting of the corresponding Collector. |
| AlarmExpiry | Calculated expiry time for the alarm shown in the form:<br>*N DATETYPE CREATE*\|*DATE*<br>where:<br>◆ *N* is the number of units<br>◆ *DATETYPE* is one of minutes, hours, days, weeks, months<br>◆ *CREATE* is relative to alarm create date<br>◆ *DATE* is relative to the alarm date stamp |
| AlarmLifetime | Defined lifetime for the alarm shown as a formatted date string using the locale of the Operations Center server. |
| CollectorType | The type of Collector that captured the source data. |
| CreateDate | Time at which the alarm was created in Operations Center, in java Date format. |
| Duplicates | The number of alarms deduplicated by this alarm. Initially 0 for an alarm, indicating that no additional alarms have been deduplicated. |
| Encoding | Encoding of the source data (where applicable). |
| FileDate | Date stamp of the source file (where applicable). |
| FileDirectory | Directory of the source file (where applicable). |
| FileLine | The line number of the data that created the alarm (where applicable). |
| FilePath | File name of the source data (where applicable). For URICollector, this value is the URI accessed. |
| FileSize | Size of the source file from which the alarm was generated (where applicable). |

| Alarm Property | Meaning |
|---|---|
| FTPInfo | Details of the FTP Server data was collected from (where applicable). |
| MONITOR_PROCESSDATE | The time stamp at which the F/X Monitor processed the alarm. This is represented in ‘long’ date format, such as the number of milliseconds since 1 January 1970. |
| MonitorHost | The hostname of the F/X Monitor from which the alarm was generated. |
| MonitorIP | The IP address of the F/X Monitor from which the alarm was generated. |
| ParserName | The name of the Reader that defined the collector and parser. |
| UniqueColumns | The list of unique alarm columns identified by configuration. |

# C Logging Configuration File

F/X Monitor logging uses the Apache log4j project as standard and expects the logging configuration file to be defined using an XML format.

Log4j configuration consists of appenders (sources to receive logging) and configuration of the appenders to use.

The supplied example is shown in the following example. It configures a log file and `stdout` in the `<appender>` tags and references their use in the `<root>` tag. For example, STDOUT logging can be suppressed by commenting out or removing the `<appender-ref ref="STDOUT"/>` tag.

Further detail on log4j logging configuration can be found at http://logging.apache.org/log4j/docs/manual.html:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">

<log4j:configuration>
  <appender name="A1" class="org.apache.log4j.FileAppender">
    <!-- change this to be your log filename -->
    <param name="File"   value="fxmonitor.log" />
    <!-- if you want a new file each run or not... -->
    <param name="Append" value="true" />
    <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern" value="%t %-5p %c{2} - %m%n"/>
    </layout>
  </appender>

  <appender name="STDOUT" class="org.apache.log4j.ConsoleAppender">
    <layout class="org.apache.log4j.PatternLayout">
           <param name="ConversionPattern" value="%d{DATE} %-5p %c - %m%n"/>
    </layout>
  </appender>

  <root>
    <priority value="info"/>
    <appender-ref ref="STDOUT"/>
    <appender-ref ref="A1"/>
  </root>
</log4j:configuration>
```

# D Sample Monitor Configuration Files

In many cases, it is helpful to view sample F/X Monitor configuration files before attempting to create one. The files corresponding to the example configurations documented in this section can be found in `/`*`OperationsCenter_install_path`*`/database/examples/example_configuration`.

- Section D.1, "Basic Configuration File," on page 51
- Section D.2, "FTP Collector with Clear Password," on page 52
- Section D.3, "FTP Collector with Encrypted Password," on page 53
- Section D.4, "XML Parser," on page 54
- Section D.5, "Severity Rules," on page 55

## D.1 Basic Configuration File

The following is a basic configuration file, `/`*`OperationsCenter_install_path`*`/database/examples/example_configuration/Demo.FX.Config.XML`:

```
<?xml version="1.0" ?>
- <fxmonitor>
- <adapters>
- <adapter>
  <name>FXAdapter</name>
  <type>PRIMARY</type>
  <group>Tail</group>
  <alarmformat>XML</alarmformat>
  <adapterport>50001</adapterport>
  <adapterhost>localhost</adapterhost>
  <adapterretry>60</adapterretry>
  </adapter>
  </adapters>
- <settings>
  <filepolicy>none</filepolicy>
- <!-- filestore>cachefiles</filestore  -->
  </settings>
- <readers>
- <reader>
  <name>Demo Reader</name>
  <class>com.mosol.integration.fx.collectors.DirectoryCollector</class>
  <parser>Demo Parser</parser>
- <settings>
- <!--  filemaxage type="mins">10</filemaxage   -->
  <directory>D:\\FX_Issues/SR19673/data</directory>
  <filematch>DemoData.csv</filematch>
  <poll>10</poll>
  <delay>0</delay>
  <retry>5</retry>
  </settings>
  </reader>
  </readers>
- <parsers>
- <parser>
  <name>Demo Parser</name>
  <class>com.mosol.integration.fx.parsers.DelimitedFileParser</class>
- <settings>
  <columns>Business_Risk,Business_Justification,Estimated_Down_Time</columns>
```

```
- <!--  lifetime type="mins" relative="create">5</lifetime   -->
  <headerline>0</headerline>
  <startline>1</startline>
  <delimiter>;</delimiter>
- <severities>
- <!--  default>UNKNOWN</default   -->
  <severity column="Business_Risk" condition="MINOR">PRODUCTION</severity>
  <severity column="Business_Risk" condition="CRITICAL">UAT</severity>
  <severity column="Business_Risk" condition="INFO">Medium</severity>
  <severity column="Business_Risk" condition="MAJOR">PRODUCTION &amp; DISASTER
RECOVERY</severity>
  <severity column="Business_Risk" condition="OK">Low</severity>
  <severity column="Business_Risk" condition="UNKNOWN" />
  </severities>

  <datecolumns>Planned_Start_Date,Planned_End_Date</datecolumns>
<dateformats>dd.MM.yyyy HH:mm,dd.MM.yyyy HH:mm</dateformats>
  </settings>
  </parser>
  </parsers>
  </fxmonitor>
```

## D.2 FTP Collector with Clear Password

The following file, `/OperationsCenter_install_path/database/examples/example_configuration/Demo.FX.Config_FTP.XML`, configures a Monitor to read data from a FTP collector using a clear password:

```
<?xml version="1.0" ?>
- <fxmonitor>
- <adapters>
- <adapter>
  <name>FXAdapter</name>
  <type>PRIMARY</type>
  <group>Tail</group>
  <alarmformat>XML</alarmformat>
  <adapterport>50001</adapterport>
  <adapterhost>localhost</adapterhost>
  <adapterretry>60</adapterretry>
  </adapter>
  </adapters>
- <settings>
  <filepolicy>none</filepolicy>
- <!-- filestore>cachefiles</filestore  -->
  </settings>
- <readers>
- <reader>
  <name>Demo FTP Reader</name>
  <class>com.mosol.integration.fx.collectors.FTPCollector</class>
  <parser>Demo Parser</parser>
- <settings>
  <directory>FTP DATA DIRECTORY</directory>
  <encoding>UTF-8</encoding>
  <filepolicy>LEAVE</filepolicy>
  <filematch>DemoData.csv</filematch>
- <!--  filemaxage type="mins">10</filemaxage   -->
  <persist>yes</persist>
  <port>21</port>
  <sendfile>ALL</sendfile>
  <server>FTP HOST NAME</server>
  <password>FTP HOST PASSWORD</password>
  <user>FTP HOST USERNAME</user>
  <poll>10</poll>
  <delay>0</delay>
  <retry>5</retry>
  </settings>
  </reader>
  </readers>
- <parsers>
- <parser>
  <name>Demo Parser</name>
  <class>com.mosol.integration.fx.parsers.DelimitedFileParser</class>
- <settings>
  <lifetime type="mins" relative="create">5</lifetime>
  <headerline>0</headerline>
  <startline>1</startline>
  <delimiter>;</delimiter>
  <datecolumns>Planned_Start_Date,Planned_End_Date</datecolumns>
  <dateformats>dd.MM.yyyy HH:mm,dd.MM.yyyy HH:mm</dateformats>
  </settings>
  </parser>
  </parsers>
  </fxmonitor>
```

For more information on using a clear password, see "FTP Collector" on page 24.

## D.3 FTP Collector with Encrypted Password

The following example, `/`*`OperationsCenter_install_path`*`/database/examples/example_configuration/Demo.FX.Config_FTP_ENCRYPTED.XML`, configures a Monitor to read data from a FTP collector with an encrypted password:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <fxmonitor>
- <adapters>
- <adapter>
  <name>FXAdapter</name>
  <type>PRIMARY</type>
  <group>Tail</group>
  <alarmformat>XML</alarmformat>
  <adapterport>50001</adapterport>
  <adapterhost>localhost</adapterhost>
  <adapterretry>60</adapterretry>
  </adapter>
  </adapters>
- <settings>
  <filepolicy>none</filepolicy>
- <!-- filestore>cachefiles</filestore  -->
  </settings>
- <readers>
- <reader>
  <name>Demo FTP Reader</name>
  <class>com.mosol.integration.fx.collectors.FTPCollector</class>
  <parser>Demo Parser</parser>
- <settings>
  <directory>FTP DATA DIRECTORY</directory>
  <encoding>UTF-8</encoding>
  <filepolicy>LEAVE</filepolicy>
  <filematch>DemoData.csv</filematch>
- <!--  filemaxage type="mins">10</filemaxage   -->
  <persist>yes</persist>
  <port>21</port>
  <sendfile>ALL</sendfile>
  <server>FTP HOST NAME</server>
  <password cipher="TEA">X-TEAV:91F483F41C4A3C7F43E157C68D318903357870B10F55FEB5</
password>
  <user>FTP HOST USERNAME</user>
  <poll>10</poll>
  <delay>0</delay>
  <retry>5</retry>
  </settings>
  </reader>
  </readers>
- <parsers>
- <parser>
  <name>Demo Parser</name>
  <class>com.mosol.integration.fx.parsers.DelimitedFileParser</class>
- <settings>
  <lifetime relative="create" type="mins">5</lifetime>
  <headerline>0</headerline>
  <startline>1</startline>
  <delimiter>;</delimiter>
  <datecolumns>Planned_Start_Date,Planned_End_Date</datecolumns>
  <dateformats>dd.MM.yyyy HH:mm,dd.MM.yyyy HH:mm</dateformats>
  </settings>
  </parser>
  </parsers>
  </fxmonitor>
```

For more information on using a encrypted password, see "FTP Collector" on page 24.

## D.4 XML Parser

The following example, `/`*`OperationsCenter_install_path`*`/database/examples/example_configuration/Demo.FX.Config_XML.XML`, shows a configuration that uses an XML parser:

```
<?xml version="1.0" ?>
- <fxmonitor>
- <adapters>
- <adapter>
  <name>FXAdapter</name>
  <type>PRIMARY</type>
  <group>Tail</group>
  <alarmformat>XML</alarmformat>
  <adapterport>50001</adapterport>
  <adapterhost>localhost</adapterhost>
  <adapterretry>60</adapterretry>
  </adapter>
  </adapters>
- <settings>
  <filepolicy>none</filepolicy>
- <!-- filestore>cachefiles</filestore  -->
  </settings>
- <readers>
- <reader>
  <name>Demo Reader</name>
  <class>com.mosol.integration.fx.collectors.DirectoryCollector</class>
- <!--  parser>Demo Parser</parser   -->
  <parser>XML TEST Parser</parser>
- <settings>
- <!--  filemaxage type="mins">10</filemaxage   -->
  <directory>D:\\FX_Issues/SR19673/data</directory>
- <!--  filematch>DemoData.csv</filematch   -->
  <filematch>XMLData.xml</filematch>
  <poll>10</poll>
  <delay>0</delay>
  <retry>5</retry>
  </settings>
  </reader>
  </readers>
- <!--  parsers>
    <parser>
      <name>Demo Parser</name>
      <class>com.mosol.integration.fx.parsers.DelimitedFileParser</class>
      <settings>
        <lifetime type="mins" relative="create">5</lifetime>
        <headerline>0</headerline>
        <startline>1</startline>
        <delimiter>;</delimiter>
        <datecolumns>Planned_Start_Date,Planned_End_Date</datecolumns>
        <dateformats>dd.MM.yyyy HH:mm,dd.MM.yyyy HH:mm</dateformats>
      </settings>
    </parser>
  </parsers   -->
- <parsers>
- <parser>
  <name>XML TEST Parser</name>
  <class>com.mosol.integration.fx.parsers.XMLParser</class>
- <settings>
- <filters>
- <filter name="Big XML Test" path="result">
  <parentattribute node="result" attribute="test-name" tofield="YahooCooCoo" />
- <!--
 convert field="time" type="DATE" inputformat="long" outputformat="ddMMyyyy
hh:mm:ss" tofield="recdate"/
  --> - <!--  parentname levelsup="1" tofield="r" /   -->
- <!--  parentname levelsup="2" tofield="MetricGroup" /   -->
- <!--  parentattribute node="Server" attribute="name" tofield="ServerName" /
  -->
- <!--
 parentattribute node="Node" attribute="name" tofield="NodeName" /   -->
- <!--  convert field="time" type="DATE" inputformat="long" outputformat="ddMMyyyy
hh:mm:ss" tofield="recdate"/   -->
- <!--  convert field="val" type="NUMERIC" /   -->
  </filter>
  </filters>
  </settings>
  </parser>
  </parsers>
  </fxmonitor>
```

For more information on using an XML parser, see “XML Parser” on page 34.

## D.5 Severity Rules

The following example, `/`*`OperationsCenter_install_path`*`/database/examples/example_configuration/Demo.FX.Config_SevRules.XML`, uses severity rules in the parser:

```
<?xml version="1.0" ?>
- <fxmonitor>
- <adapters>
- <adapter>
  <name>FXAdapter</name>
  <type>PRIMARY</type>
  <group>Tail</group>
  <alarmformat>XML</alarmformat>
  <adapterport>50001</adapterport>
  <adapterhost>localhost</adapterhost>
  <adapterretry>60</adapterretry>
  </adapter>
  </adapters>
- <settings>
  <filepolicy>none</filepolicy>
- <!-- filestore>cachefiles</filestore  -->
  </settings>
- <readers>
- <reader>
  <name>Demo Reader</name>
  <class>com.mosol.integration.fx.collectors.DirectoryCollector</class>
  <parser>Demo Parser</parser>
- <settings>
- <!--  filemaxage type="mins">10</filemaxage   -->
  <directory>D:\\FX_Issues/SR19673/data</directory>
  <filematch>DemoData.csv</filematch>
  <poll>10</poll>
  <delay>0</delay>
  <retry>5</retry>
  </settings>
  </reader>
  </readers>
- <parsers>
- <parser>
  <name>Demo Parser</name>
  <class>com.mosol.integration.fx.parsers.DelimitedFileParser</class>
- <settings>
  <columns>Business_Risk,Business_Justification,Estimated_Down_Time</columns>

- <!--  lifetime type="mins" relative="create">5</lifetime   -->
  <headerline>0</headerline>
  <startline>1</startline>
  <delimiter>;</delimiter>
- <severities>
- <!--  default>UNKNOWN</default   -->
- <severity column="Business_Risk" condition="MINOR">
  PRODUCTION
  <sevRule column="Estimated_Down_Time" condition="MINOR" equation="gt">0</
sevRule>
  <sevRule column="Estimated_Down_Time" condition="MAJOR" equation="gt">2</
sevRule>
  </severity>
  <severity column="Business_Risk" condition="CRITICAL">UAT</severity>
- <severity column="Business_Risk" condition="INFO">
  Medium
  <sevRule equation="script">fxAdapter.setSeverity("CRITICAL");</sevRule>
  </severity>
  <severity column="Business_Risk" condition="MAJOR">PRODUCTION & DISASTER
RECOVERY</severity>
- <severity column="Business_Risk" condition="OK">
  Low
  <sevRule column="Estimated_Down_Time" condition="MINOR" equation="lt">1</
sevRule>
  <sevRule column="Estimated_Down_Time" condition="CRITICAL" equation="gt">2</
sevRule>
  </severity>
  <severity column="Business_Risk" condition="UNKNOWN" />
  </severities>
  <datecolumns>Planned_Start_Date,Planned_End_Date</datecolumns>
  <dateformats>dd.MM.yyyy HH:mm,dd.MM.yyyy HH:mm</dateformats>
  </settings>
  </parser>
  </parsers>
  </fxmonitor>
```

For more information on using severity rules, see "More about Severity Rule Tags" on page 32.

# E Java SimpleDateFormat Syntax

Date and time formats are specified by date and time pattern strings. Within date and time pattern strings, unquoted letters from A to Z (including a to z) are interpreted as pattern letters representing the components of a date or time string. Text can be quoted using single quotes (') to avoid interpretation. "''" represents a single quote. All other characters are not interpreted; they're simply copied into the output string during formatting or matched against the input string during parsing.

- Section E.1, "Pattern Letters," on page 57
- Section E.2, "Regular Expression Constructs," on page 58

## E.1 Pattern Letters

Pattern letters are usually repeated, as their number determines the exact presentation:

- **Text:** For formatting, if the number of pattern letters is four or more, the full form is used; otherwise, a short or abbreviated form is used, if available.

  For parsing, both forms are accepted, independent of the number of pattern letters.
- **Number:** For formatting, the number of pattern letters is the minimum number of digits, and shorter numbers are zero-padded to this amount.

  For parsing, the number of pattern letters is ignored, unless it is needed to separate two adjacent fields.
- **Year:** For formatting, if the number of pattern letters is four, the year is truncated to two digits; otherwise, it is interpreted as a number.

Table E-1 lists the pattern letters that are defined. All other characters from A to Z and from a to z are reserved.

***Table E-1*** *SimpleDateFormat Syntax*

| Letter | Date or Time Components | Examples |
|---|---|---|
| `a` | Am/pm marker | `PM` |
| `D` | Day in year | `189` |
| `d` | Day in month | `10` |
| `E` | Day in week | `Tuesday`; `Tue` |
| `F` | Day of week in month | `2` |
| `G` | Era designator | `AD` |
| `H` | Hour in day (0–23) | `0` |
| `h` | Hour in am/pm (1–12) | `12` |
| `K` | Hour in am/pm (0–11) | `0` |

| Lette r | Date or Time Components | Examples |
|---|---|---|
| k | Hour in day (1–24) | 24 |
| M | Month in year | July; Jul; 07 |
| m | Minute in hour | 30 |
| S | Millisecond | 978 |
| s | Second in minute | 55 |
| W | Week in month | 2 |
| w | Week in year | 27 |
| y | Year | 1996; 96 |
| Z | Time zone | -0800 |
| z | Time zone | Pacific Standard Time; PST; GMT-08:00 |

## E.2 Regular Expression Constructs

The following tables provide the constructs for regular expressions:

- Table E-2, "Characters," on page 58
- Table E-3, "Character Classes," on page 59
- Table E-4, "Predefined Character Classes," on page 59
- Table E-5, "POSIX character classes (US-ASCII only)," on page 60
- Table E-6, "Classes for Unicode Blocks and Categories," on page 60
- Table E-7, "Boundary Matchers," on page 60
- Table E-8, "Greedy quantifiers," on page 61
- Table E-9, "Reluctant quantifiers," on page 61
- Table E-10, "Possessive quantifiers," on page 61
- Table E-11, "Logical operators," on page 61
- Table E-12, "Back references," on page 62
- Table E-13, "Quotation," on page 62

***Table E-2*** *Characters*

| Construct | Matches |
|---|---|
| x | The character x. |
| \\ | The backslash character. |
| \0n | The character with octal value 0n (0 <= n <= 7). |
| \0nn | The character with octal value 0nn (0 <= n <= 7). |
| \0mnn | The character with octal value 0mnn (0 <= m <= 3, 0 <= n <= 7). |

| Construct | Matches |
|---|---|
| `\x`hh | The character with hexadecimal value `0x`hh. |
| `\u`hhhh | The character with hexadecimal value `0x`hhhh. |
| `\t` | The tab character (`'\u0009'`). |
| `\n` | The newline (line feed) character (`'\u000A'`). |
| `\r` | The carriage-return character (`'\u000D'`). |
| `\f` | The form-feed character (`'\u000C'`). |
| `\a` | The alert (bell) character (`'\u0007'`). |
| `\e` | The escape character (`'\u001B'`). |
| `\c`x | The control character corresponding to x. |

***Table E-3*** *Character Classes*

| Construct | Matches |
|---|---|
| [abc] | a, b, or c (simple class). |
| [^abc] | Any character, except a, b, or c (negation). |
| [a-zA-Z] | a through z, or A through Z, inclusive (range). |
| [a-d[m-p]] | a through d, or m through p: [a-dm-p] (union). |
| [a-z&&[def]] | d, e, or f (intersection). |
| [a-z&&[^bc]] | a through z, except for b and c: [ad-z] (subtraction). |
| [a-z&&[^m-p]] | a through z, and not m through p: [a-lq-z](subtraction). |

***Table E-4*** *Predefined Character Classes*

| Construct | Matches |
|---|---|
| Any character (could match line terminators). | |
| \d | A digit: [0–9] |
| \D | A nondigit: [^0–9] |
| \s | A whitespace character: [ \t\n\x0B\f\r] |
| \S | A nonwhitespace character: [^\s] |
| \w | A word character: [a-zA-Z_0-9] |
| \W | A nonword character: [^\w] |

***Table E-5*** *POSIX character classes (US-ASCII only)*

| Construct | Matches |
|---|---|
| \p{Lower} | A lowercase alphabetic character: [a-z] |
| \p{Upper} | An uppercase alphabetic character: [A-Z] |
| \p{ASCII} | All ASCII: [\x00-\x7F] |
| \p{Alpha} | An alphabetic character: [\p{Lower}\p{Upper}] |
| \p{Digit} | A decimal digit: [0-9] |
| \p{Alnum} | An alphanumeric character: [\p{Alpha}\p{Digit}] |
| \p{Punct} | Punctuation: One of: !"#$%&'()*+,-./:;<=>?@[\]^_`{\|}~ |
| \p{Graph} | A visible character: [\p{Alnum}\p{Punct}] |
| \p{Print} | A printable character: [\p{Graph}\x20] |
| \p{Blank} | A space or a tab: [ \t] |
| \p{Cntrl} | A control character: [\x00-\x1F\x7F] |
| \p{XDigit} | A hexadecimal digit: [0-9a-fA-F] |
| \p{Space} | A whitespace character: [ \t\n\x0B\f\r] |

***Table E-6*** *Classes for Unicode Blocks and Categories*

| Construct | Matches |
|---|---|
| \p{InGreek} | A character in the Greek block (simple block). |
| \p{Lu} | An uppercase letter (simple category). |
| \p{Sc} | A currency symbol. |
| \P{InGreek} | Any character, except one in the Greek block (negation). |
| [\p{L}&&[^\p{Lu}]] | Any letter, except an uppercase letter (subtraction). |

***Table E-7*** *Boundary Matchers*

| Construct | Matches |
|---|---|
| `^` | The beginning of a line. |
| `$` | The end of a line. |
| `\b` | A word boundary. |
| `\B` | A nonword boundary. |
| `\A` | The beginning of the input. |
| `\G` | The end of the previous match. |
| `\Z` | The end of the input, but for the final terminator, if any. |
| `\z` | The end of the input. |

***Table E-8*** *Greedy quantifiers*

| **Construct** | **Matches** |
|---|---|
| X? | X, once or not at all. |
| X* | X, zero or more times. |
| X+ | X, one or more times. |
| X{*n*} | X, exactly *n* times. |
| X{*n*,} | X, at least *n* times. |
| X{*n*,*m*} | X, at least *n*, but not more than *m* times. |

***Table E-9*** *Reluctant quantifiers*

| **Construct** | **Matches** |
|---|---|
| X?? | X, once or not at all. |
| X*? | X, zero or more times. |
| X+? | X, one or more times. |
| X{*n*}? | X, exactly *n* times. |
| X{*n*,}? | X, at least *n* times. |
| X{*n*,*m*}? | X, at least *n*, but not more than *m* times. |

***Table E-10*** *Possessive quantifiers*

| **Construct** | **Matches** |
|---|---|
| X?+ | X, once or not at all. |
| X*+ | X, zero or more times. |
| X++ | X, one or more times. |
| X{*n*}+ | X, exactly *n* times. |
| X{*n*,}+ | X, at least *n* times. |
| X{*n*,*m*}+ | X, at least *n*, but not more than *m* times. |

***Table E-11*** *Logical operators*

| **Construct** | **Matches** |
|---|---|
| XY | X followed by Y. |
| X\|Y | Either X or Y. |
| (X) | X, as a capturing group. |

***Table E-12*** *Back references*

| Construct | Matches |
|---|---|
| \*n* | Whatever the *n*th capturing group matched. |

***Table E-13*** *Quotation*

| Construct | Matches |
|---|---|
| \\ | Nothing, but quotes the following character. |
| \\Q | Nothing, but quotes all characters until \\E. |
| \\E | Nothing, but ends quoting started by \\Q. |

# F XPath Reference Resources

Some useful XPath examples and papers include:

- http://www.w3.org/TR/xpath (http://www.w3.org/TR/xpath)
- http://www.w3schools.com/xpath/xpath_syntax.asp (http://www.w3schools.com/xpath/xpath_syntax.asp)
- http://www.w3schools.com/xpath/xpath_examples.asp (http://www.w3schools.com/xpath/xpath_examples.asp)