

Saxonite Limited
ArtiSan Mailbox Manager
Rules Engine

Author: Keith Young
Reference: ArtiSan/Rules
Date: 17 June 2005
Version: 1.0
Distribution: Company confidential

Abstract

This document describes in detail the ArtiSan Rules Engine

© Saxonite Ltd 2005

Change History

1.0 Initial

Contents

Introduction	2
Preconditions for Rules	2
Preparing for Evaluation	2
One Rule – Many Rules	3
Evaluation Results	3
The Rules System	4
The Rules System Type Libraries	4
The Rules System Objects	4
Global Objects and Global Scripts	5
Testing Rules	5
Rules Maintenance	6
Reloading the Rules	6
Futures	Error! Bookmark not defined.

Introduction

The ArtiSan mailbox manager incorporates a Rules engine that is used to evaluate the actions to be performed on mail. The engine is invoked automatically during the scan process and evaluates rules designed by the user that tell the engine whether the particular mail item should be archived and or stubbed. The engine also provides a mechanism to allow the user to specify a retention date for the item.

It is important to bear in mind that for each item there may be many instances of that item (i.e. the same mail in multiple mailboxes) and because the instances are evaluated separately they are probably being evaluated with different rules. Consequently, you can never go from an item to an instance because it is a one-to-many relationship.

There is no reason why the retention dates cannot be changed after the process of archival, indeed the system does this itself and it is a function we will be using in some future enhancements. However, there is no utility at present to allow you to amend these values. Version 2 extends the tables so that there is now a notion of a disposition as separate from retention. This will allow the system to express concepts like "delete item" and "place item on permanent hold" and to express a reason, e.g. "deleted by the administrator", etc. This works in concert with the orphan management in V2 that allows the system to determine whether an item is an orphan and to respond appropriately (an orphan is an Exchange stub that refers to an item that is no longer available for whatever reason).

Preconditions for Rules

The rules system is only invoked for items that have a potential action associated with it. There are a number of short-circuits in place to reduce the number of evaluations made and to reduce the work of the ArtiSan server.

The first short-circuit is tested when scanning users. If a user does not have any rules associated with them either a rule on the mailbox or in any folder or to any group to which they belong, ArtiSan skips evaluating any mail in the users mailbox and will not invoke the rules system. Equally, if the user is identified as a member of an exclusion list (or is not a member of an inclusion list), then the user is skipped and no rules are evaluated. Exclusion/Inclusion lists are available for Exchange Servers, Mailbox Stores and ADS groups. There is also an exclusion/inclusion group for folders. Rules are never evaluated on items that are in an excluded folder or no in the list of included folders. Rules are also not evaluated for ad-hoc operations such as archives performed using the Outlook add-in.

Once the system has found a valid candidate for testing, it checks whether the item is already archived and restored. If the item has been restored and the restore timeout is exceeded, the item is restubbed. The system then checks if the item is stubbed. Items that are archived and stubbed are skipped because there is no further operation to carry out.

The system then checks if the item is archived and exceeded retention. If so, the item is deleted and rules are also not evaluated.

In effect, the only items passed to rules are those items that may have a potential action associated with them that can be deduced from evaluating rules.

Preparing for Evaluation

Prior to evaluating the rules, ArtiSan prepares the message and associated information for presentation to the rules system. To do this, the system creates an object called MessageInfo. A MessageInfo object contains key information about the particular message being evaluated. Information is copied to a new instance of the object and this object is passed to the rules system along with user, folder and category details of the message. The Rules system then constructs a RuleItem object which contains all these details and this is exposed in the Rules system and forms the basis of the evaluation.

One Rule – Many Rules and Precedence

Once the Rules system has an item to evaluate, it uses the information passed to derive from the configuration to construct a list of scriptlets that might have to be evaluated. This list of scriptlets may be as a result of a particular folder, user, group or message category. In the process of evaluating the list the rules system also aggregates scripts that are valid either as a result of inheritance or through aggregation of templates. The scripts are evaluated until the first one returns an indication that the item should be archived, i.e. it short circuits the rest of the scripts.

The order of evaluation of scripts is to start at a folder rule and if there is a script evaluate it. Then check if there is an aggregated (template) rule and evaluated that. Then check the inherit flag. If true, the system moves to the parent node and repeats the process. This takes into account both folder and user rules. Once folders and user rules are evaluated, the system moves on to evaluate groups which are handled in alphabetic order.

Evaluation Results

Each evaluation results in three main outputs:

Item.Archive	If set true, this indicates that the item should be archived. Setting this value to true also short circuits rule evaluation.
Item.StubMessage	This indicates whether the item should be stubbed. This cannot be true unless Item.Archive is also true.
Item.Retention	This is the retention period (in months) to associate with the item when it is archived. The system uses the maximum retention period across all instances of an item within the user mailboxes for rules that have been evaluated.

No state is maintained after completion of a rule evaluation, so the results for one item never affect another.

The Rules System

The rules system is a standard VBScript engine just as used by ASP files and from the command line using cscript and wscript. Rules can contain all the things that are supported by any VBScript system such as dim variables, set objects, call methods and so on.

Typically, rules are created either using the rules UI either as generated rules or as custom rules. Generated rules allow the user to specify rules without actually typing in the code. A generated rule is simply creates the VBScript code??.

The Rules System Type Libraries

The main ArtiSan type library is the basic type library of the AMM system that defines general purpose objects such as Management (used for managing the system), the Engine interface (the thing used to operate on mail items) and so on. Specifically for rules it defines the MessageInfo object. The rules type library defines things specific to rules such as the script host, the Evaluator and the Rule Item object.

The script engine has some COM modules automatically added to it. These are the "ArtiSan Type Library" and the Microsoft Scripting Run Time. The latter is a set of utility objects such as Dictionaries and some classes that allow you to interact with the operating system such as files and folders.

The Rules System Objects

The main objects in the rules type libraries are:

MessageInfo	This is an abstract of an item in Exchange.
RuleItem	The item is used in the context of a rule evaluation and contains the message, ancilliary information like folder size, user name and so on.
Evaluator	The Evaluator is the rules engine itself. You can use the Evaluator to test rules in situ.

As described above, the rules system is launched through use of the engine or other part of the system and these type libraries are added. As each mail is tested for an action, the system creates an object called a MessageInfo. MessageInfo object contains a read-only abstract of the information in the mail item. The MessageInfo object is actually a specialised version of a general purpose property bag. It is possible to add your own properties to a MessageInfo object and to retrieve those properties. The main purpose of this is to make the data being passed into rules extensible so we can add new features later. Incidentally, the .art files created in the store are the result of serialising the MessageInfo object to a disk file. The MessageInfo object supports standard IPersist... interfaces.

The RuleItem object is created by the rule evaluator prior to processing of scriptlets. It is populated with the MessageInfo and the other properties of the message. The evaluator then evaluates each script in turn passing the RuleItem object into the scripting environment and testing the results. From the perspective of scripting the RuleItem object appears as a global variable called Item. Hence, the usage:

```
item.Archive = True
```

```
if item.Message.Size then ...
```

The Item object has a property called message that is set to the MessageInfo object. Incidentally, the system always forces the "Option Explicit" at the top of each scriptlet so you have to define variable before their use. One method of the RuleItem allows the rules themselves to Log information to the AMM.log. For instance, you can add lines in rules like:

```
Item.Log 1, "Test message has size " & Item.Message.Size
```

This will add a log at trace level 1 to the AMM.log if the "AMM Rules" value (under HKLM\Saxonite\AMM\Trace) is set to log at level 1.

Global Objects and Global Scripts

You can add to the list of preloaded objects using the registry key:

HKLM\Saxonite\AMM\Rules\GlobalObjects

Under this key, you create values whose name is the name of the object as you want it to appear in script and whose value is the ProgId of the object (e.g. "Scripting.FileSystemObject", "ADODB.Connection"). The system will then load these objects on the fly when the script engine is launched. This allows you to cache the object in the engine rather than constantly creating and destroying them, but you should be aware that these objects are shared across evaluations and so will carry state information between separate invocations of the rules. So suppose you decided to preload a database connection in the rules script (because you didn't want to keep logging on and off a database), then you need to be aware that the connection is going to be shared across all evaluations in the rules system itself.

In version 2, you can also add preformed scripts of localized functions. These are specified in the key:

HKLM\Saxonite\AMM\Rules\GlobalScripts

They consists of a set of name values, where the name is an arbitrary descriptive name set by the customer and the value is the full path to the file containing the script. Note that the script is loaded when the rules configuration changes.

Testing Rules

The main rules test software is called RuleChecker and is located in the bin directory of the AMM system.

One important feature of the AMM system is that most of the internal objects are exposed to COM programming environments including the Rules system. Consequently, rules can contain the various objects in the ArtiSan type library like the Management object. Management allows the customer to:

- To interact with the configuration system to both read, write and delete values
- To access to key system information like the name of the domain server
- To access details of licencing
- To access stats from the engine
- To access to version information

Indeed, the management UI is largely built from functions provided by Management (in association with ADSI and ADO). Management also allows you access to the logging system. Since management objects can be loaded from within the customer rules, trace logs can be generated from those rules. To do this, create a management object, set the LogName property of the object (this will be reflected in the registry under trace, where the trace level is set) and then use the Log method of the management object to pass the relevant log level and trace string. The trace information will turn up in AMM.log. This is different from using the Log method of the RuleItem object, because management allows the rule to have its own log type and level that will be listed separately in the Trace key and therefore is controllable separately. The Log method on the RuleItem object shares the trace context with the rule system, so increases to the log level for the rules system will affect both.

Rules Maintenance

The rules system keeps an XML tree of rule definitions that are loaded and cached inside the rule engine. When a test on a mail is to be performed, the system uses the rules defined in the XML to create a list of scriptlets that need to be run based on properties such as user, groups and folders. These are the scripts used to evaluate the status of the message (should it be archived/stubbed and what is the retention).

Rules are edited through the management UI.

Rules may be placed remotely using a web request. Configuring a remote rules file disables the editing of rules from the local management UI. Remote rules can only be edited from the Management UI local to those rules.

Reloading the Rules

In most circumstances, whenever the configuration of the rules system changes, the rules automatically reload themselves. If you need to reload the rules separately, then you must stop all ArtiSan services and the IIS service.