Exspans
Systems Inc.

AutoMan V3.2

Features Description

May 2014

Exspans Systems Inc.
Perth Mews
RPO 20082
Perth
ON K7H 3M6
Canada

www.exspans.com

Available Features

AutoMan is divided into optional feature sets so that the user only needs to have the procedures that are of value to them. By selecting a set of components that address the specific needs of an organization, AutoMan can supply all the functionality needed for smooth trouble free operations and disaster recovery. All of the interfaces that AutoMan provides execute GAL fragments to perform services on behalf of the user.

AutoMan provides a choice of interfaces to use and processors to actuate.

Function and feature list

Function	Interface Choices

System Message Automation	STAGE=0
	STAGE=1
	STAGE=2

Retained Message Automation	

Operator Command Automation	STAGE=0
	STAGE=1

Conditions Automation	
SYSOUT Scanning service	
Schedule Automation	Basic Scheduling Advanced Work Scheduling Definitions Jobs Scheduler
JES Command Support	
Cross Systems support	SYSPLEX Intercommunications Shared Device Communications Support
Synchronization services	Intra system Cross systems
System Services Automation & Config.Objects	
Command Lists	
Generalized Automation Language Compiler	
Backup Copy Support	

System Message Automation

AutoMan intercepts and processes system messages. When a message is intercepted a user written GAL fragment is executed and decisions can be made about how to proceed.

There are three control points, or Stages, defined in the system at which messages can be intercepted. The choice of stage depends on the immediacy with which the messages are to be handled, and the visual effects desired on the console. The message monitors can make decisions about whether to process messages based on message contents and it source. During message processing all systems support variables that relate to a message, such as route codes, tokens, descriptor, and console may be used and altered. The same command sets will execute the same in all environments, except as noted for display command in Stage 2.

Stage 2

Stage 2 conceptually corresponds to an MCS console. At this stage AutoMan processes messages after they have been displayed on the console and the color, highlighting and alteration of text cannot be performed. Messages received at this stage contain any alterations that may have been made to the text by AutoMan Stage 0/1 processing, or by subsystems and other automation products. At this stage accounting for all automation activity accrues to the AutoMan address space.

Stage 1

Stage 1 corresponds to the subsystem interface and allows AutoMan to run alongside other automation software, which uses the MPF interface. Messages are processed before they appear on the console display. At this stage accounting for some activity may accrue to the address space of the program on whose behalf work is being done. This is affected by parameters that specify how accounting is to be performed. At this stage the color, text and highlighting of messages may be altered by programmed commands. Messages may also be suppressed from display.

Stage 0

Stage 0 corresponds to the MPF message interface. This is the first available message processing point in the system and is the most efficient interface at which to stop messages from being propagated, if that is required. If a message is to be deleted, so that it does not appear on the console, it is best to delete it at this interface. Messages, which are deleted at this stage, are not propagated further. At this interface messages may be altered, before being propagated to later stages. This interface receives messages and performs activity related to those messages before any other system activity related to those activities continues. At this stage the color and highlighting of messages can be altered. Messages can be suppressed from the console display or deleted from the log. Accounting for automated messages is applied to the address space, which issued the message. If a separate MPF exit is written for any specific message, that message will not be processed by AutoMan at Stage 0. If you want AutoMan to process a message at this stage, make sure that you do not have an MPF exit for that message.

Scheduler Automation

AutoMan schedules jobs, tasks and other work. Scheduling is performed on the basis of immediacy, time of day, and date and complex date relationships. Simple statements specify when an activity is to take place, when it is to be repeated and when repetition is to stop. Scheduled activity may be simple or complex. Scheduling can submit jobs and tasks, and can run GAL programs that track the state of processes. The scheduler supports clock synchronization and will respond to time changes, such as Daylight Savings Time.

Basic Scheduler

The basic scheduler works on time, day, date and repetition. The basic scheduler recognizes day of week names and month names in standard English. This provides sufficient support to perform all routine systems management oriented scheduling.

Advanced Work Scheduling Definitions

The AWS allows the definition of date names in English and other languages. The date names are Calendar Objects. A Calendar Object is a hierarchy of definition. The user defines and uses their own terms, based on their own conventions, and in words that are meaningful to the work being done. The names chosen should immediately become meaningful when used on a schedule dispatch statement. A Calendar Object can contain any kind of date information. It can contain floating dates which have uncertainty in their definitions. Any date relationship of any complexity can be resolved by AWS support, and used to schedule an event. This is used when special days, or intercultural date translation, are needed.

Jobs Scheduler

The jobs scheduler tracks jobs as they progress through the system and allows follow on activity to be performed depending on the success or failure of previous jobs. It also enables the tracking and interception of started tasks and TSO users. It works with the Basic Scheduler and AWS to schedule and track jobs through execution.

Retained Message Automation

AMRF can leave sticky messages on the console before AutoMan is started. The Retained Message scripts can be used to respond to messages that are held in place by AMRF. The same Retained Message scripts can also be used to respond to WTOR's that have been placed on the console before AutoMan starts up. The Retained Message script facility is identical to the normal message script facility and all of the same commands and actions, which are available for normal message processing, are available to the Retained Message facility. The only difference is, is that the contents, color and highlighting of messages cannot be changed, because they are already on the console, so commands that do this are ignored. AMRF definitions may be divided into sections and the sections activated when needed. AMRF definitions, which are not in a section are executed once as soon as AutoMan starts up, or when the script is loaded. The retained message processor enables decisions to be made about messages based on their age.

Operator Command Automation

The Operator Command Processor intercepts, processes and can potentially alter, system commands before the system or subsystem gets them. This facility can be used to create completely new commands. This facility can also be used to disallow some commands, so that they will not be processed on the system. It can be used to check command parameters and perform different activity depending on installation defined standards.

The Operator Command processor can run at STAGE 0, which corresponds to the first MPF exit, at STAGE 1, which corresponds to the sub-system interface and in addition can also run as a JES2 command exit, for specialist legacy support.

Conditions Automation

The Conditions Processor enables programmed activity to take place when specific conditions are detected in the system. It can be used to monitor all or a specific set of jobs for CPU time, CPU utilization, paging or storage page usage, and various other potential fault indicators.

SYSOUT Processor

The SYSOUT Processor intercepts and interprets lines of text being written to SYSOUT files. The text may be scanned and activity performed based on values being written. The text may be altered by this processor. Typical usage is to monitor applications logs and send emails to responsible individuals when signs of potential trouble are found.

Cross Systems

AutoMan can work cooperatively with other copies running on different images of the operating system. These may be in different LPARs or on completely different machines. The GAL language contains easy to use constructs that enable direct user programmable cross systems communication of information and commands.

GAL contains constructs that perform cross systems communications in background to support the real time execution of GAL programs.

Cross systems support is provided by:

SYSPLEX Intercommunications

Sysplex cross systems communications services. The functionality to interconnect is built in to GAL and is invoked by simple statements, when this support is enabled.

Shared device communications Support

If sysplex services are not available or multiple separate non-communicating sysplexes are to work cooperatively, AutoMan can use a shared device channel.

Synchronization

AutoMan provides queuing and waiting facilities to synchronize all processes. GAL provides all the statements needed to invoke full systems and applications synchronization, in simple easy to use statements. Synchronization across systems is invoked simply, and requires no more than identifying the partner system in GAL statements. So the user can program their own synchronicity across systems, if their scenario requires it.

GAL contains event and queue objects that extend their reach to all tasks, programs and systems. This means that user written command lists has the potential to synchronize complex work across single and multi-image sysplex environments.

Some objects, like SSID, have the potential for cross systems status query capabilities built in to their definitions, and perform their own synchronization..

System Services Automation

System Services are the subsystems and programs needed to provide the operational support that is needed for all the work done by the system. These include such things as JES, VTAM, OMVS, CICS etc. The SSID processor enables these to be started, stopped and controlled by simple statements, without any human intervention. SSID objects contain the capabilities of cross systems synchronization.

Configurations

The Configuration Manager manages groups of SSIDs in such a way that they can be brought up and down in context of each other, bearing in mind prerequisites, dependencies and successors. This facility enables the operations management to view groups of dependent subsystems as manageable units of system definition. Configurations control and track the state of every subsystem to ensure it is up when needed and down when not. Configurations make sure that, what you want up is up, and what you want down, is down. Configurations can automatically restart failing subsystems and track their progress.

Once defined, a Configuration becomes an object that can be used in GAL statements.

Command Lists

Commands lists are groups of GAL statements and commands, which are stored as members of a partitioned dataset.

Command lists group together logic, commands and orders, queries and decisions, which relate to a specific activity, so that, when that activity is required, a single command can be issued to do all of it.

Command lists may be run in response to console traffic, as a result of scheduled activity or by operator console commands. Command lists may contain any AutoMan command or GAL statement, except those specific to setting the appearance of messages on the console, and may invoke other command lists. Command lists may be run from a library, or they may be preloaded.

Command lists may be dispatched synchronously, and run as if they were a subroutine of another script, or they may be dispatched asynchronously, and create a parallel task.

A command list can be run any time there is a piece of work to be done.

There is no pre-specified maximum number of command lists, or recursive iterations of a command list that can be active at any time. This is limited only by system resources.

Generalized Automation Language

GAL is a simple language, a little like BASIC. It has the same fundamental structures of variables, assignment and query statements. Its simplicity of form surrounds very powerful concepts. GAL is mostly written in short phrases, attached to an event or process that needs to be controlled.

The objects that GAL works with represent complex concepts and control points. Cross systems query and synchronization is assumed when support is activated. From the GAL point of view it is only one parameter to a statement whether other systems should be consulted. GAL works with individual numbers and with assemblages of systems components, reduced to the form of objects. GAL enables simple queries and decisions to manipulate what runs, what waits, and what checks.

Variables

The user defines variables to contain the pieces of information they are working with to affect a result. Variables are used in GAL for calculations, testing and communications between different AutoMan control processors.

Variables may be global or local. Global variables may be used to communicate information between processors and threads. Local variables are only known in the thread in which they are defined. Variables are used to replace text in commands and orders. Variables allow the user great flexibility in generating what will be used at execution time.

REXX

GAL provides a REXX interface. REXX scripts may be executed by command from any GAL procedure.

EMAIL

EMAIL is a fundamental object in GAL and is defined by the GAL EMAIL command, so that Email messages and pager alerts may be sent from any AutoMan processor. Email control is provided by GAL statements.

JOB output to JES may also be intercepted and directed to EMAIL if the recipient wants their job output.

GAL Compiler

GAL is both an interpreted, and a compiled, language.
The compiler enables the definitions for automation and scheduling to be compiled into an object format.

These automation objects can be transported to new systems, to bring up standard z/OS images with their workloads defined, and in operation.

Compiled GAL helps to improve the system responsiveness. A system, starting up under AutoMan with a compiled GAL initialization program, will start faster and be fully operative quicker than any other system start up sequence method.

Compiled code is also less susceptible to accidental or malicious alteration, and can provide backup security by being a known object, from which you can start a known system.

The GAL compiler provides a quick and easy way to test compile GAL programs during development, to discover syntax errors. Even if the end result will be the source text being used to run with AutoMan, the compiler can speed up development.

JES

AutoMan can be started as the first thing the system does at IPL and can control the start and stopping of JES and all other subsystems. AutoMan can intercept JES output and redirect it. AutoMan can also be started when the system is fully initialized and JES is already active, when it is not being used for systems control. AutoMan submits jobs through JES and retrieves sysout from it. AutoMan provide JES command services for specialist legacy functions. AutoMan is fully integrated into either, an environment where JES is and one where it is absent.

External API

AutoMan supports an external API service, so that user applications programs can communicate and request services. A demonstration utility called AutoBat is available to demonstrate this functionality and provide utility services.

Backup Copy Support

This enables licensed code to be used on different machines for testing, backup or emergency recovery.