Automation Security V12.1

ca technologies | Automic™ White Paper

TABLE OF CONTENTS

TABLE OF CONTENTS

Introduction

With the increasing importance of IT systems and electronically stored information, security issues are now a vital concern for IT managers, and application security is one of the top priorities at Automic.

The automation engine controls key data processing across all platforms within an enterprise. As the automation engine is a key application, effective security is an essential component of the product.

This document discusses this topic, covering all major issues concerning security, the automation engine and our offerings.

Product Architecture

Multi-Tenancy

Automic is a native multi-tenant architecture and one single installation can support the whole enterprise. Multi-tenant architecture is an approach in which a single instance of the software centrally serves multiple client enterprises (tenants), highly segregated within a single instance for security and compliance.
Because IT today is generally viewed as a service, being able to logically segregate client enterprises is necessary. Simplifying the reporting and charge-back of delivered services to the appropriate department value is realized by consolidating enterprises onto one centralized product. Such separation also enhances security. Many products on the market do not support multi-tenancy, resulting in increased costs to:

- Purchase additional hardware and software for each end customer
- Purchase additional hardware to separate development, quality assurance and production environments
- Maintain and administer the additional hardware, databases and scheduling systems

Therefore, there is no need to implement as many Automic Systems as the company has environments. One unique Automic infrastructure will manage environments (called clients in Automic) such as QA/DEV/PROD (up to 9999) and/or business units in a secure way (every environment has its own security, resources and scheduling design).

Lifecycle management is made easier with the help of embedded tools enabling the import of new scheduling object/design from QA to PROD for instance. This unique concept is a fantastic added-value for growing companies who have to integrate new IT environments through acquisitions in a normalized way.

With its multi-tenant architecture, the automation engine partitions each tenant data and configuration, so that each enterprise works with a customized segregated instance of the application. It simplifies and reduces administration costs and increases flexibility in management and performance.

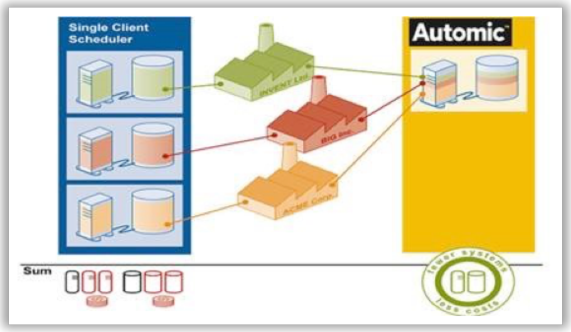


Figure 1: High-level view of the multi-tenant concept

System Design

Our system design provides the basis for its excellent security. From the very beginning, the Automic platform has been designed with a security focus. Secure systems are quality systems, because when designed for security, systems automatically become very robust.

Our architects designed the platform as a collection of self-contained objects. Small self-contained software units with limited functionality are the basis for large and complex systems. Such small units are simple in design, development and maintenance, thus reducing the number of possible errors.

Centralized Architecture

The platform has been designed according to the latest developments in software technology. Based on the advantages of the centralized architecture it has been extended with some essential features.

For data storage a centralized relational database management system is used. A clear separation of management and data storage provides full database independence. Unlike similar applications the automation engine does not store any job scheduling information in flat files, thus avoiding the security risks most other job schedulers have.

All scheduling data including the JCL is always centrally available, providing a comprehensive overview over past, present and future processing for the best possible operational safety.

Through its special server design the automation engine is available non-stop and is fully scalable without increasing the system’s complexity.

System stability is achieved through the server layout. The centralized control unit can be distributed across any number of CPUs and physical servers. Furthermore, third-party tools are not required and resources are only used on dedicated servers without affecting the production systems in any way.

Automation Engine

The Automation Engine is designed as a collection of self-contained objects. Therefore, it's not a monolithic block of software, but consists of any number of work processes and communication processes. With their very specific and limited functionality these modules are easy to develop and maintain, one of the reasons for the automation engine's stability and reliability.

Furthermore, this architecture provides fault-tolerance and scalability.

Network

Network infrastructure plays a key role for availability. The automation engine uses the TCP/IP protocol family for data transmission. These protocols have been developed for fail-safe peripheral communication and are therefore very well suited for safe data transfer. TCP/IP needs relatively low effort for the design of redundant networks, thus making the design of highly available and fail-safe networks easy.

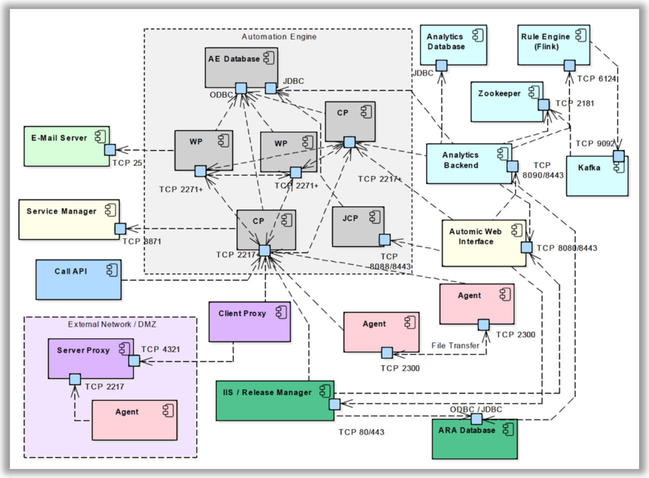


Figure 2: Network Architecture

Development and Quality Assurance

Our engineering teams work according to the highest quality and security standards. Based on thorough design, object-oriented code is created and each encapsulated object is thoroughly reviewed and tested. Special development training is held to increase awareness of security risks.

Development Process

In a nutshell, we have undergone a cultural and technological change in engineering. This was not easy and has taken a good year to achieve; we have an R&D department of over 150 staff, dispersed across six countries, working in five very different time-zones.

We operate multiple scrum teams and encourage them to use whatever open source tooling they want; that's the world of development, which many of you tell us you operate in. Indeed, that's why we have adopted this approach – to mirror your goals as closely as possible. These teams have a lot of autonomy and work in a loosely coupled environment. We previously worked on a cycle of build, deploy, test and release, while taking advantage of some automation to shorten timescales. However, we wanted to make a major step-change, so we adopted new agile processes and a DevOps approach.

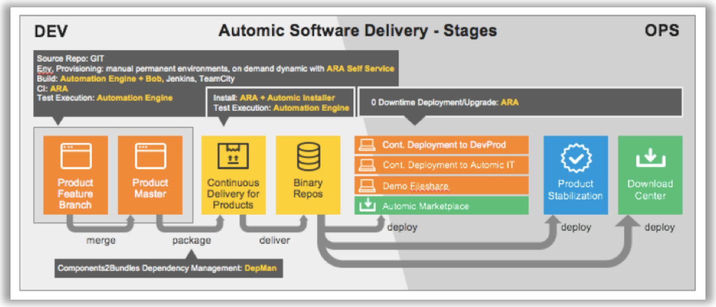

DEV
Automic Software Delivery - Stages
OPS
Source Repo: GIT
Env. Provisioning: manual permanent environments, on demand dynamic with ARA Self Service
Build: Automation Engine + Bob, Jenkins, TeamCity
CI: ARA
Test Execution: Automation Engine
Install: ARA + Automic Installer
Test Execution: Automation Engine
0 Downtime Deployment/Upgrade: ARA
Product Feature Branch
Product Master
Continuous Delivery for Products
Binary Repos
Cont. Deployment to DevProd
Cont. Deployment to Automic IT
Demo Fileshare
Automic Marketplace
Product Stabilization
Download Center
merge
package
deliver
deploy
deploy
deploy
Components2Bundles Dependency Management: DepMan

Figure 3: Automic Software Delivery Stages

The diagram above shows our process flow for all updates to Automic software code. Failure at any stage in this process results in the DevOps team being alerted to errors, with corrections made. Successful completion of this exercise results in the product feature branch updates being deployed into Automic's production environments.

Quality Assurance

We are building quality in and securing it thanks to embedded test experts and team members have relevant certifications (development related or quality related). Business praised tools are used for test management, reporting and tracking changes, with both static and dynamic code analysis tools used daily on every feature branch. Continuous integration is implemented in order to detect code breaking changes early during development and automated tests on different system platforms are triggered to ensure quality. Close partnership with selected customers allows for improved production ready products . Additionally we continuously release product updates to internal users in two-week sprints. It is vital these new features are communicated and used immediately to get feedback as quickly as possible .

Security

We have several stages and processes in place to ensure a secure product for our customers:

- During the design process of security critical features our in-house security department is involved as soon as possible to achieve a secure software design
- First static code analysis (SonarQube) is deployed to run on every code change and we monitor our used library for known vulnerabilities (Nexus IQ Server)
- Every code change is implemented as a pull request, and each pull request gets reviewed by at least one different developer using several checklists (i.e. OWASP secure coding practices checklist) and expert knowledge
- Our developers receive various external and in-house training courses at least once a year on secure coding practices and security testing, which helps during development and testing to avoid common mistakes that can lead to security vulnerabilities
- After each sprint all new features are tested separately by our in-house security team using manual and automated security tests, with the centralized sprint security reviews adding an additional layer of quality assurance to our product
- Before release we are doing external security audits on our products

Education and Security Training

We think that it is important for our engineers that they stay ahead of the latest security threats and vulnerabilities. Therefore we offer external and in-house trainings on various security topics such as Web Application Security (OWASP Top 10), secure coding best practices, penetration testing, security testing tools, etc. All our engineering staff receive at least one security related training course suitable for his/her position once a year.

Maintenance

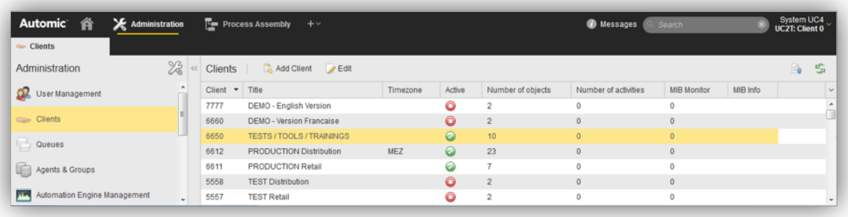
We supply our customers with security patches according to our support policy during our maintenance period. Updates are available as hotfixes or regular service releases.

Security Concept and Authorization

Users, Groups and Roles

The powerful access control concept of the automation engine allows administrators to grant user/user groups only access to features, objects and views that they need and/or are allowed to see. A very granular set of authorization methods and privileges empowers the admin of each client to properly set up the access control to the system.

View of several Automic clients (= environments), each of them covering the Job scheduling of a distinct business unit or environment:



Client	Title	Timezone	Active	Number of objects	Number of activities	MIB Monitor	MIB Info
7777	DEMO - English Version		🔴	2	0	0	
6660	DEMO - Version Francaise		🔴	2	0	0	
6650	TESTS / TOOLS / TRAININGS		✓	10	0	0	
6612	PRODUCTION Distribution	MEZ	✓	23	0	0	
6611	PRODUCTION Retail		✓	7	0	0	
5558	TEST Distribution		🔴	2	0	0	
5557	TEST Retail		🔴	2	0	0	

Figure 4: View of several Automic Clients (= environments)

Automic provides a robust internal security mechanism and users of Automic will access via a user ID provided by the administrators, which includes a department for additional security. Automic provides for advanced password management that includes the ability to require specific password formats such as:

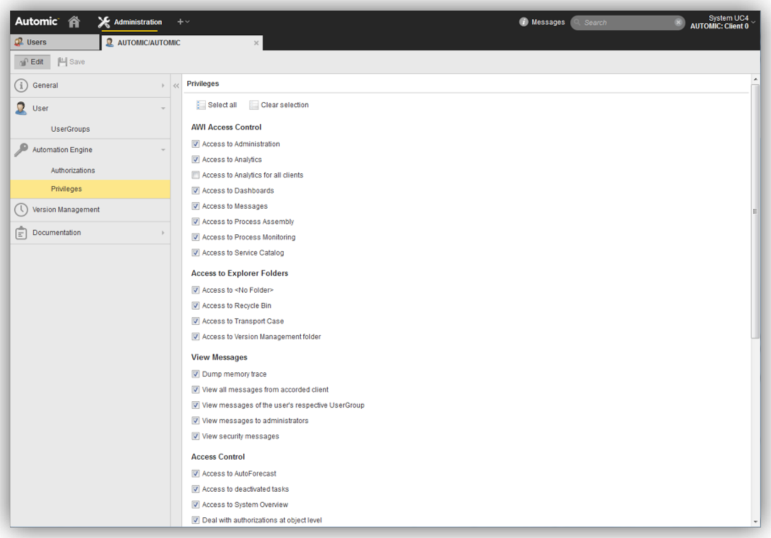
- Minimum password length
- Special character requirements
- Capital letter requirements
- Numeric requirements

Automic also allows for integration with Active Directory as well as LDAP compatible user authentication. This integration is for the sole purpose of user/password validation. Once a user is connected to Automic, all of their activity is tracked for the purpose of audit reporting.

Permissions and Privileges

The automation engine comes with a secure setup out-of-the-box. The administrator must explicitly enable the automation engine privileges for users and the same procedure should apply for operating systems throughout the landscape. The automation engine requires very few privileges and all unused rights should be turned off. It is designed for ultimate security, but nevertheless also depends on operating system security features – we recommend using the latest versions of operating systems and database management systems. To avoid exploits of the automation engine system, it is critical to apply security fixes provided by the individual systems' vendors.

Some of the privileges assigned to the Automic users and user groups



Privileges

Select all Clear selection

AWI Access Control

- [✓] Access to Administration
- [✓] Access to Analytics
- [ ] Access to Analytics for all clients
- [✓] Access to Dashboards
- [✓] Access to Messages
- [✓] Access to Process Assembly
- [✓] Access to Process Monitoring
- [✓] Access to Service Catalog

Access to Explorer Folders

- [✓] Access to <No Folder>
- [✓] Access to Recycle Bin
- [✓] Access to Transport Case
- [✓] Access to Version Management folder

View Messages

- [✓] Dump memory trace
- [✓] View all messages from accorded client
- [✓] View messages of the user's respective UserGroup
- [✓] View messages to administrators
- [✓] View security messages

Access Control

- [✓] Access to AutoForecast
- [✓] Access to deactivated tasks
- [✓] Access to System Overview
- [✓] Deal with authorizations at object level

Figure 5: Overview of privileges

In addition to the privileges that give users access to functions in the Automic user interface, authorizations for users and user groups can be defined on an object level within every environment. By doing so, users and user groups obtain exclusive access rights to a particular object. Several levels of rights can be applied such as read, delete, execute, cancel, access to statistics etc. and rights can be applied simultaneously to the folder/sub folder structure. Therefore, the Automic security system is fully granular and will adapt to security policies and organization.
Security can be as broad as administrator access to a role, or as granular as, "Read access to one unique job in one folder during the hours of 8:00am to 4:00pm on Tuesdays." Any runtime actions taken against jobs or workflows are logged both in the report of the object and also within our audit report for easy viewing and reporting.

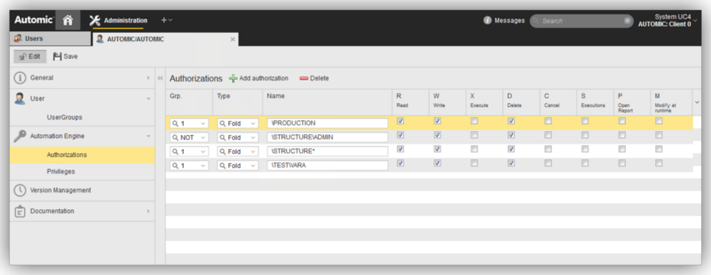
Authorization and Access

Access to folders, statistics, reports and objects is subject to authorization. (Note that servers and agents are also objects.) The automation engine offers a variety of access controls for the objects managed. This involves:

- Function level authorization (the ability to grant/revoke permission for certain functions of the platform)
- Object level authorization (the ability to create access control lists (ACLs) at the level of single objects)
- ACL aggregation (the ability to group object level ACLs together through intelligent filter criteria in order to reduce security management effort, e.g. by naming conventions)
- user grouping (the ability to aggregate users to groups also for the purpose of decreased management effort)

Although folders are objects and rights can be defined for them, specifying folder rights does not prevent access to objects stored in them. A user who is not allowed to access a particular folder could still access an object in this folder (such as if it is used in a workflow. The command “edit” is available from almost anywhere including workflows). Objects that should not be accessed by particular users should also be protected.

The following example refers to the above explorer structure and shows how rights can be assigned and explains the different effects:



Grp.	Type	Name	R Read	W Write	X Execute	D Delete	C Cancel	S Executions	P Open Report	M Modify at runtime
1	Fold	\PRODUCTION	✓	✓	☐	✓	☐	☐	☐	☐
NOT	Fold	\STRUCTURE\ADMIN	✓	✓	☐	✓	☐	☐	☐	☐
1	Fold	\STRUCTURE*	✓	✓	☐	✓	☐	☐	☐	☐
1	Fold	\TEST\VARA	✓	✓	☐	✓	☐	☐	☐	☐

Figure 6: Authorization by naming convention

- Line 1: Users can access the folder “PRODUCTION” but not its corresponding subfolders
- Line 2: Users can - regardless of other authorizations that have been granted - not access the folder “ADMIN” which is a subfolder of “STRUCTURE”
- Line 3: Users can access the folder “STRUCTURE” including its subfolders, except for the subfolder defined in line 2
- Line 4: Users have access rights to the folder “VARA” but not to its subfolders or to the folder “TEST”

Agents

For security reasons, a newly installed agent does not have any rights. Furthermore, it cannot execute tasks, nor can it be selected in the objects of a client of the automation engine system.

The agent logs on to the automation engine system with an agent object created in the folder HOST of the system client 0000. In each agent you can define the clients that should be given access rights.

The access rights read, write and execute are available:

- “Read” - The agent can send files (file transfer).
- “Write” - The agent can receive files (file transfer).
- “Execute” - The agent can execute jobs.

Separation of Concern

Finally, in order to allow for a role model implementation supporting typical IT organization structures, the need for separating execution from design and implementation is a given. Object usage must not automatically imply permission for object definition (e.g. a login object for a target system gets defined by a person knowing the password but can be used by a lot of different automation specialists implementing workflows and thereby using the respective login object). The authorization of the automation engine supports the separation of design and implementation (e.g. developers can use the logins without knowing the passwords, adding an additional layer of security).

Example for Authentication, Authorization and Auditing

Customers can decide on which options they want to use and implement.

Security within the solution is preferably role-driven, and in order to keep maintenance of security rules manageable, role assignment is driven through group membership. As such, we can grant or revoke access in the solution by simply making the necessary changes to group membership. Membership of a user in multiple groups (and the underlying role assignment) should be supported and the access controls sufficiently flexible. It must be possible to grant a user full workload execution rights on one server, read-only rights on another and make sure this user has no access whatsoever towards definitions not on these servers. It is preferred that users can also be assigned a subset of capabilities on a resource. For instance, the ability to run workload as non-root on some systems, or only trigger workload execution with a certain priority. Permissions and access controls should be enforceable immediately (and not as part of internal batch runs that are only scheduled daily, or caches that expire after a day).

Logins to Hosts, Databases and Applications

Furthermore, host, database and application logins are objects in the automation engine. Login objects usually are defined by DBAs and system personnel. Passwords for the logins are encrypted and never displayed in clear type. Developers can use the logins without knowing the passwords, adding an additional layer of security.

Sarbanes-Oxley Act

For many years, global businesses have been using the automation engine to achieve the kind of IQ levels required by SOX. While the automation engine is not a SOX specific solution, its functionality is perfectly aligned to the requirements of achieving SOX compliance. It ensures the secure creation, automation and control of all business process cycles, as well as the safe and reliable transfer of data between diverse applications, both within the enterprise and with external parties. The system also offers automated control of all reporting processes. As standard components of the automation engine's proven functionality, these features help companies to meet SOX-related challenges. In the context of critical business applications such as SAP/mySAP, the automation engine can automate, control and securely manage all the SAP financial reporting processes.

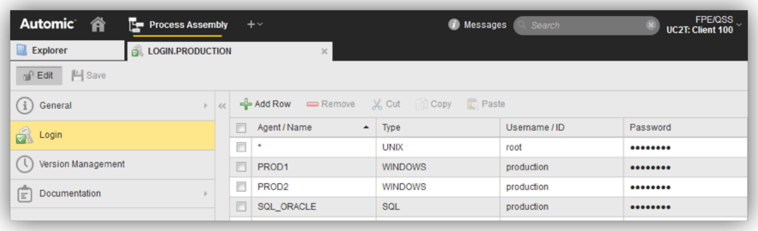
Password Management

Managing enterprise wide passwords is a great challenge for IT managers. As such, Automic provides comprehensive support for the secure central storage of enterprise passwords.

The Automic platform manages user IDs and the passwords of host systems by means of login objects. Login objects store the information about now to access host systems. They can be used for activating executable objects on host systems.

The automation engine's object oriented design allows for flexible and granular access privileges to login objects. The use of login objects can be limited to individual users or user groups, while read, write and execute rights for login objects can be defined separately. System programmers, for example may create jobs that use a certain login object, but have no access to the actual login details and password. Once a password is stored in a login object it is not displayed to the user at any time.

Login objects are – like all information – stored in the database. To prevent unauthorized access to critical login information outside of the automation engine's authorization regime all stored passwords are encrypted.



Agent / Name	Type	Username / ID	Password
*	UNIX	root	••••••••
PROD1	WINDOWS	production	••••••••
PROD2	WINDOWS	production	••••••••
SQL_ORACLE	SQL	production	••••••••

Figure 7: Sample Login object

Enhanced Security Concept for Release Automation

The security concept of ARA is based on the concept of folder permissions, create type permissions, approvals and login objects.

Create Type Permission

The create type permissions controls the creation of new entities that are a certain type, for instance, a Production Environment. However, even if you do not have the create permissions, you may still have the permission to work with the type's entities.

Folder Permission

The folder permissions controls what you are allowed to do with entities in a folder. Whenever you create a new entity, you store it in a folder (e.g. RELEASES). Depending on the folder's permission, you can do different things with entities in that folder. There are five different permissions which can be set per folder and user/group by an administrator:

- read: You can access the folder and view all entities within it
- use: You are allowed to reference entities in this folder from other entities
- (Example: If you want to assign an environment to a deployment profile, you require use permissions on the folder in which the environment is stored)
- write: You are allowed to change entities in this folder
- (Example: When you want to assign an environment to a deployment profile, you require write permissions on the folder in which the deployment profile is stored)

- delete: You are allowed to delete entities in this folder
- execute: You are allowed to use the entity when executing a workflow

The following entities are stored in dedicated folders: Activity, Activity Template, Application, Component, Deployment Profile, Deployment Target, Environment, Environment Reservation, Login, Package, Queue, Release, Workflow (Application and General).

Approvals

Approvals add another security layer to workflow executions and activities. An administrator can configure via approval rules, who needs to approve a workflow execution based on the context of the execution (e.g. to which environment a package gets deployed). Workflows will only start if all approvers give their ok.

Network Communication

No external encryption solutions are required within an Automic environment. All necessary encryption is done natively via an AES key level of your choice (128, 192 or 256). This encryption is used for communication between the automation engines and agents, including API calls.

In addition to the core components of the automation engine all user-facing components and APIs support TLS v1.2:

- Automic Web Interface (AWI)
- Release Manager UI
- API Endpoints (JCP, ARA REST/SOAP API, Analytics REST API)
- Proxy Client / Server
- Analytics Backend (Kafka, Zookeeper, Rule Engine)

Confidentiality and Encryption

All network communication is per default encrypted using AES-256. For authentication of two connection partners a pre-shared key is used. The key exchange can happen in various ways depending on the configured authentication level.

Authentication Methods

The automation engine supports different authentication methods, which define how the communication key is initially distributed. Each authentication method offers various advantages and disadvantages depending on the required security level. As default the easiest authentication mode (“None”) is used, because it offers a simple and automatic setup of new agents.

Authentication method	Description
None	An agent that starts for the first time can immediately log on to the automation engine system. The “company key” (a term used in each automation engine system) is automatically derived from the automation engine system’s name. It prevents an agent from logging on to an automation engine system with a different company key afterwards.
Server	The company key must be determined during the automation engine installation. Subsequently, it can be exported to a file and used during agent installation. The agents can log on to the automation engine system when they start the first time but they cannot automatically be used. The administrator must release them in the system overview of client 0000. By doing so, the automation engine automatically transfers the authentication package via the line to the relevant agent. Only then is the agent authenticated and ready to use.

Authentication method	Description
Server and Agent	The company key must be determined during the automation engine installation. Some preparatory work is required to make sure that the agents can log on to the automation engine system. Create an agent object for each agent in system client 0000. Subsequently, export an authentication package and store it on the agent's computer for the installation. Now the agent is ready to use. In order guarantee a top secure installation, Automic recommends transferring the authentication package to the agent either manually or via a secure line. Doing so ensures that potential hackers never obtain access to the authentication package via the network.

It is also possible to withdraw an authentication of an agent by highlighting the relevant agent in the system overview of client 0000 and selecting the corresponding context menu command. This prevents the agent from being used until it has been re-authenticated.

Type of Keys

The automation engine uses three different types of keys:

- Company key (the system's public key)
- Transfer key (shared secret between two communication partners)
- Session key (AES key used for encryption of data of a connection)

The company key is the systems' public key and is defined once on installation of the system. It is used to encrypt the transfer keys. If the company key gets changed or deleted all other keys will also become invalid.

In addition to the company key, there are transfer and session keys. The transfer key is the shared secret of two connection partners (e.g. automation engine and agent). It applies for a connection and is defined before or on first connection (depending on the chosen authentication mode). It is used for authentication and to derive the session key.

The session key is used for data encryption and applies for a communication session. After reconnect a new session key is used.

Key exchange and Communication

Depending on the chosen authentication method a new agent can receive a transfer key on various ways. The default authentication is "none". During the first setup the agent connects to the automation engine without verifying the identity of the automation engine using the company key (gray). Afterwards a Diffie-Hellman protocol is used to exchange the transfer key (yellow) between the new agent and the automation engine. During communication a session key (green) is generated and used for the communication. The transfer key is not transmitted over the wire any more.

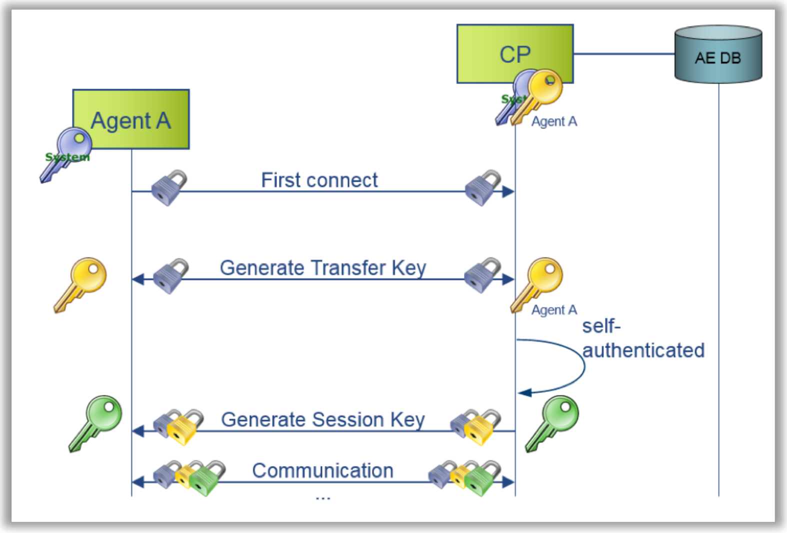


Figure 8: Initial key exchange during first connection (authentication method: None)

A regular start is using basically the same mechanism without exchanging the keys (because both parties already know the transfer key). Again a session key is used for communication. The transfer keys are stored in the database on the automation engine side and in the key store on agent side.

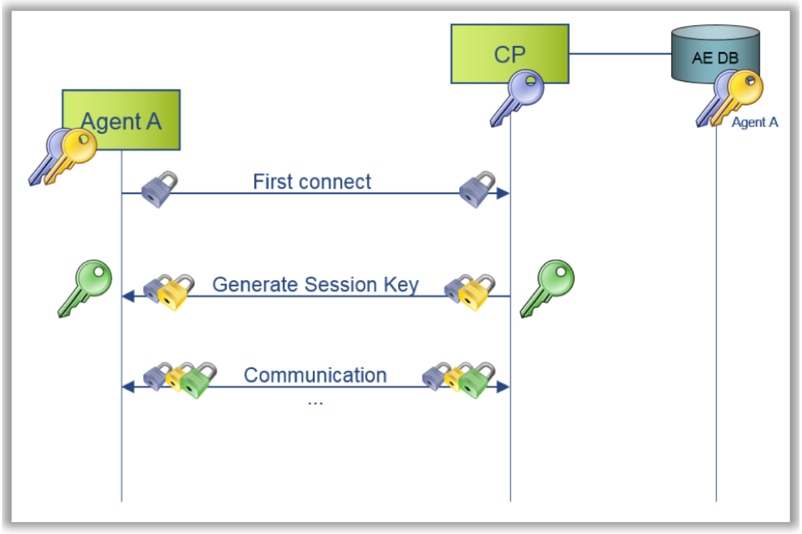


Figure 9: Regular connection initialization after initial setup

Using the authentication mode “agent only” the agent verifies the identity of the CP by using the company key. Therefore the installation of agents requires the user to extract the company key from client 0 and to manually transfer it to the agent. When first starting, the agent imports this company key, connects and verifies the identity of the CP by using the supplied company key. Afterwards the same mechanism as before is used to exchange the transfer key using Diffie-Hellman.

Advantages

- Quick and easy setup
- Transfer key gets distributed automatically using a secure key exchange protocol
- Does not involve any manual steps
- New agents get registered to the system automatically

Disadvantages

- Key gets somehow transferred over the wire (even using an additional secured connection)
- No authentication of the two connection partners is done

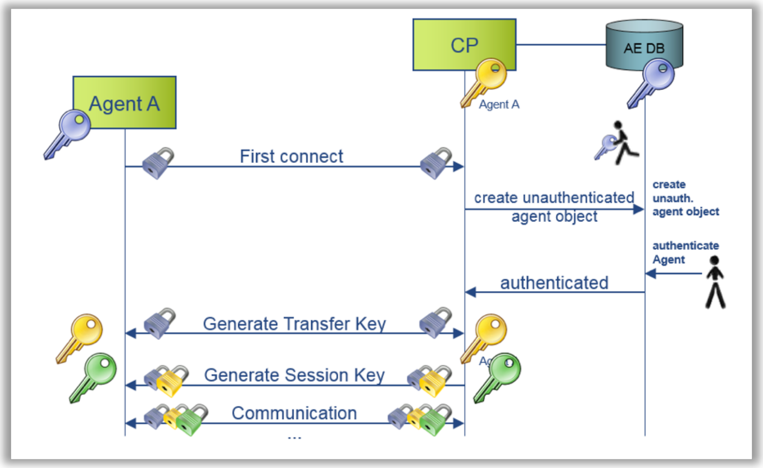


Figure 11: Initial key exchange during first connection (authentication method: Server)

The most secure authentication mode is to use "agent and server". In this case the installation of agents requires the user to manually create an agent object. Afterwards he has to extract the authentication package from client 0 for that particular agent object in the system overview and manually transfer it to the agent (e.g. using a second secure channel like an USB stick). On first start, the agent imports the company key and the transfer key. In this case, the transfer key is not transmitted over the wire at any time.

Advantages

- Semi-automatic setup of new agents
- Authenticity of the automation engine is ensured on agent side
- New agents have to be authenticated manually by an administrator

Disadvantages

- Involves manual distribution of the company key
- Transfer key transferred over the wire (using a secured connection)

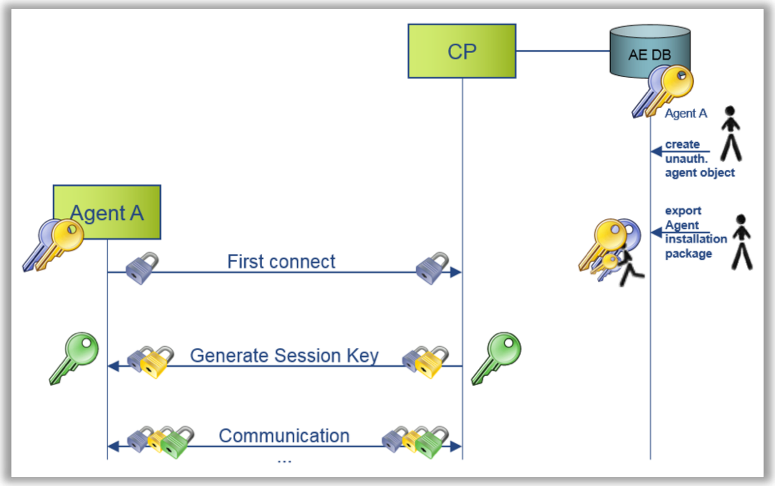


Figure 12: Initial key exchange during first connection (authentication method: Agent and Server)

After keys have been exchanged, a regular start is the same for all three methods. All messages send over the wire are encrypted using the established session key.

The authentication of the transfer partners is done by the transfer keys. Both know a common secret and therefore both know the identity of the other one.

Advantages

- Authenticity of both connection partners is ensured
- It is not able for unauthorized agents to connect to the automation engine
- No key is transferred over the wire at any time

Disadvantages

- cv

File Transfers

The automation engine provides an optimized and improved file transfer procedure. It sends the complete file transfer request (including wildcard specifications in partially qualified file transfers) to the source agent. The sending agent is responsible for resolving the request (determining the files). In order to ensure a secure connection the agent receives a session key from the automation engine. This mechanism ensures that nobody can start a file transfer without prior authorization from the automation engine.

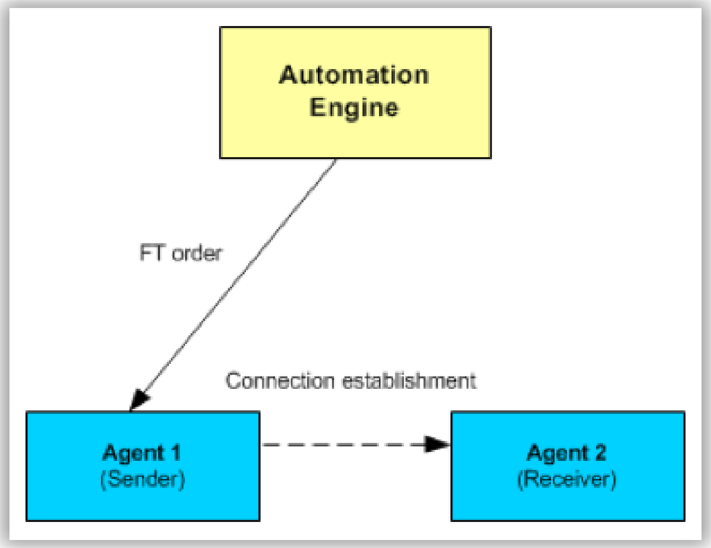


Figure 13: File Transfer Workflow

Connection Establishment

The sending agent tries to establish a connection to the receiving agent. If this attempt fails (for example, because of the firewall settings), it notifies the automation engine. The file transfer request is then sent to the receiver, which now tries to establish a connection to the sender. After the connection has been established, the receiving agent transfers the file transfer request to the sender.

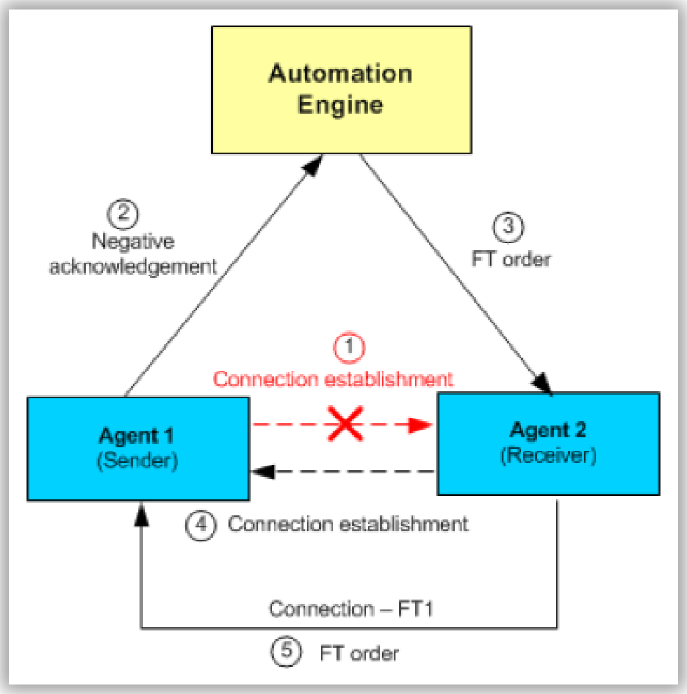


Figure 14: Connection establishment of a file transfer

Transmission Security

The accuracy of transferred data is verified with a checksum that is embedded in the data stream.

Frontend Communication and APIs

All our web frontends, (i.e. Automic Web Interface and Release Manager), and SOAP/REST-APIs support HTTPS including TLS 1.2. After installation, all communications are unencrypted over HTTP. We offer comprehensive guides on how to setup HTTPS in our documentation. It is recommended to run our web applications and APIs in a HTTPS-only mode, as this ensures the validation of the two connection partners, i.e. it protects against man-in-the-middle attacks by using valid and signed certificates. Furthermore, it encrypts the communication from prying eyes.

Integrity

Due to our key distribution every agent possesses different transfer keys, thus the automation engine can verify the identity of the connection partner. Furthermore, it is not possible for an attacker to alter the message without knowing the transfer key. This will ensure the integrity of the message. In addition, every messages contains a checksum, which ensures that no byte was altered during the network transfer.

Availability

A thorough discussion of security issues must include availability of all systems and components.
Careful design, continuous object-oriented development and a well-structured quality assurance process ensure best possible functionality of all Automic components.
Realistically software cannot be 100 percent failure proof. Additionally, the operating system or hardware can fail. Therefore several concepts provide the high availability of the Automic platform (See Section 1.2).

Authenticity

Securing the identity of a communication partner is an important requirement for secure network communication. An attack may try to intercept the connection between different components (man-in-the-middle attack) in order to penetrate the system or break confidentiality (eavesdropping). Therefore, it is important to ensure that the communication partner is who they claim to be.

Man-in-the-Middle Attacks

The architecture and the key management protects the automation engine and its communication against man-in-the-middle attacks. After setup the connection between agent and automation engine starts instantly without any key exchanged required. Therefore, it is not possible to capture the key during initialization of the connection because it has not to be transferred any more. Depending on the chosen authentication method the transfer key is never transmitted over the wire. If an attacker wants to intercept or read a connection, he/she has to compromise an agent to be able to read the sent messages. In that case he/she would have been able to do so anyway, because he/she already has access to the machine. However, older messages cannot be decrypted, because of the different session key used to encrypt the messages. Furthermore, he/she is not able to read the messages of different connections, because they use different transfer and session keys to the compromised agent.

Unauthorized Access to the Automation Engine by Compromising an Agent

Assuming that an attacker manages to compromise an agent, the system design of the automation engine prevents that this could lead to the compromising of other systems.

The agent connects to the automation engine from where it receives all commands. It is not possible for the agent to send commands back to either the automation engine or other agents (except file transfers). The only thing the agent does is connect and wait for commands.

For file transfers the agent receives a session key from the automation engine in order to connect to the other agent. Therefore, it is not possible for the agent to start a file transfer without prior request, because it does not know the session key from the other agent.

Assuming that an agent gets compromised, an attacker may be able to listen to the communication between the automation engine and the compromised agent, thus it is may be possible to intercept credentials for the compromised agent during the start of a job execution (assuming that different credentials are used on different machines). In this scenario the attacker wouldn't need the credentials anyway, because the system is already compromised and he is able to execute code on the machine.

Audit and Compliance

The automation platform offers enterprise compliance support to allow logging capabilities that keep track of any user activity within the system. Enabling such a feature must ensure tracking (logging) of all modifications to the system's repository, including:

- The user that made the modification
- A time and date stamp
- Before and after image copies of the changed data

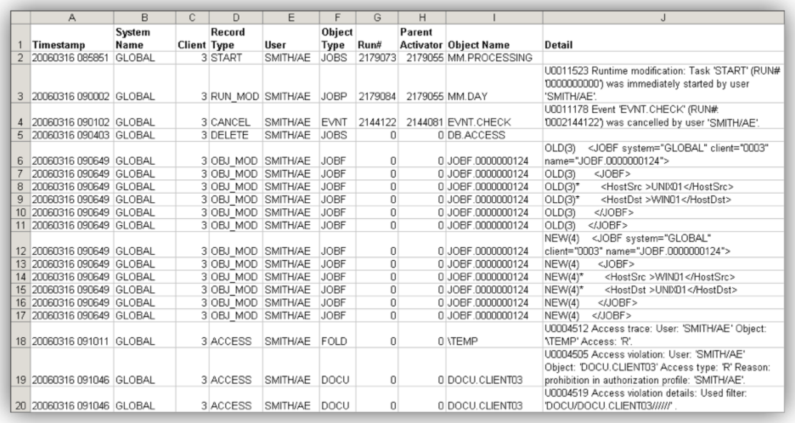
Such a feature improves systems integrity and ensures compliance with various regulatory statutes.

Job Execution

All job executions are audited and held within the Automic database. For each job, the following metrics are retained:

- Title
- Object Name
- Object Version
- Execution Queue
- Automic Run ID
- Automic User
- Passed Parameters
- Target Agent
- Target User Context

- Target Process ID
- Activation Timestamp
- Start Timestamp
- End Timestamp
- Runtime
- Return Status
- Return Code
- Processor Utilization



	A	B	C	D	E	F	G	H	I	J
1	**Timestamp**	**System Name**	**Client**	**Record Type**	**User**	**Object Type**	**Run#**	**Parent Activator**	**Object Name**	**Detail**
2	20060316 085851	GLOBAL	3	START	SMITH/AE	JOBS	2179073	2179055	MM.PROCESSING	
3	20060316 090002	GLOBAL	3	RUN_MOD	SMITH/AE	JOBP	2179084	2179055	MM.DAY	U0011523 Runtime modification: Task 'START' (RUN# '0000000000') was immediately started by user 'SMITH/AE'.
4	20060316 090102	GLOBAL	3	CANCEL	SMITH/AE	EVNT	2144122	2144081	EVNT.CHECK	U0011178 Event 'EVNT.CHECK' (RUN# '0002144122') was cancelled by user 'SMITH/AE'.
5	20060316 090403	GLOBAL	3	DELETE	SMITH/AE	JOBS	0	0	DB.ACCESS	
6	20060316 090649	GLOBAL	3	OBJ_MOD	SMITH/AE	JOBF	0	0	JOBF.0000000124	OLD(3) <JOBF system="GLOBAL" client="0003" name="JOBF.0000000124">
7	20060316 090649	GLOBAL	3	OBJ_MOD	SMITH/AE	JOBF	0	0	JOBF.0000000124	OLD(3) <JOBF>
8	20060316 090649	GLOBAL	3	OBJ_MOD	SMITH/AE	JOBF	0	0	JOBF.0000000124	OLD(3)* <HostSrc >UNIX01</HostSrc>
9	20060316 090649	GLOBAL	3	OBJ_MOD	SMITH/AE	JOBF	0	0	JOBF.0000000124	OLD(3)* <HostDst >WIN01</HostDst>
10	20060316 090649	GLOBAL	3	OBJ_MOD	SMITH/AE	JOBF	0	0	JOBF.0000000124	OLD(3) </JOBF>
11	20060316 090649	GLOBAL	3	OBJ_MOD	SMITH/AE	JOBF	0	0	JOBF.0000000124	OLD(3) </JOBF>
12	20060316 090649	GLOBAL	3	OBJ_MOD	SMITH/AE	JOBF	0	0	JOBF.0000000124	NEW(4) <JOBF system="GLOBAL" client="0003" name="JOBF.0000000124">
13	20060316 090649	GLOBAL	3	OBJ_MOD	SMITH/AE	JOBF	0	0	JOBF.0000000124	NEW(4) <JOBF>
14	20060316 090649	GLOBAL	3	OBJ_MOD	SMITH/AE	JOBF	0	0	JOBF.0000000124	NEW(4)* <HostSrc >WIN01</HostSrc>
15	20060316 090649	GLOBAL	3	OBJ_MOD	SMITH/AE	JOBF	0	0	JOBF.0000000124	NEW(4)* <HostDst >UNIX01</HostDst>
16	20060316 090649	GLOBAL	3	OBJ_MOD	SMITH/AE	JOBF	0	0	JOBF.0000000124	NEW(4) </JOBF>
17	20060316 090649	GLOBAL	3	OBJ_MOD	SMITH/AE	JOBF	0	0	JOBF.0000000124	NEW(4) </JOBF>
18	20060316 091011	GLOBAL	3	ACCESS	SMITH/AE	FOLD	0	0	\TEMP	U0004512 Access trace: User: 'SMITH/AE' Object: '\TEMP' Access: 'R'.
19	20060316 091046	GLOBAL	3	ACCESS	SMITH/AE	DOCU	0	0	DOCU.CLIENT03	U0004505 Access violation: User: 'SMITH/AE' Object: 'DOCU.CLIENT03' Access type: 'R' Reason: prohibition in authorization profile: 'SMITH/AE'.
20	20060316 091046	GLOBAL	3	ACCESS	SMITH/AE	DOCU	0	0	DOCU.CLIENT03	U0004519 Access violation details: Used filter: 'DOCU/DOCU.CLIENT03//////' .

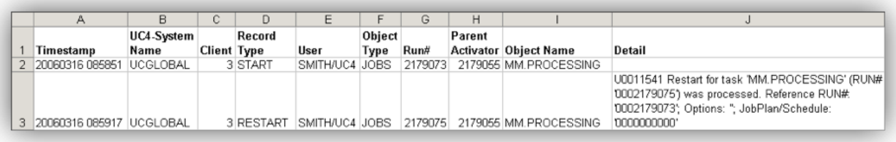
Figure 15: Sample revision report

Furthermore, the commands issued to and the responses returned from the target system, along with any post-processing, are audited.

Therefore, it is possible to determine who executed each workflow and when and for each constituent tasks the commands executed and the security context used to execute the command. The system response is captured, plus runtime metrics such as start time, end time and return code.

Task Starts and Restarts

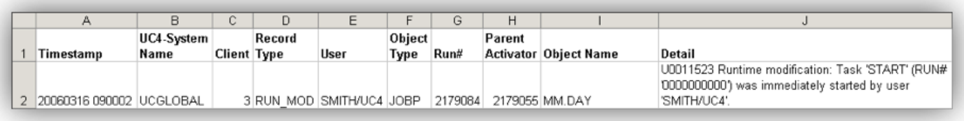
The start time (i.e. the activation time) is stored.



	A	B	C	D	E	F	G	H	I	J
1	**Timestamp**	**UC4-System Name**	**Client**	**Record Type**	**User**	**Object Type**	**Run#**	**Parent Activator**	**Object Name**	**Detail**
2	20060316 085851	UCGLOBAL	3	START	SMITH/UC4	JOBS	2179073	2179055	MM.PROCESSING	
3	20060316 085917	UCGLOBAL	3	RESTART	SMITH/UC4	JOBS	2179075	2179055	MM.PROCESSING	U0011541 Restart for task 'MM.PROCESSING' (RUN# '0002179075') was processed. Reference RUN# '0002179073'; Options: ''; JobPlan/Schedule: '0000000000'

Modifications at Runtime

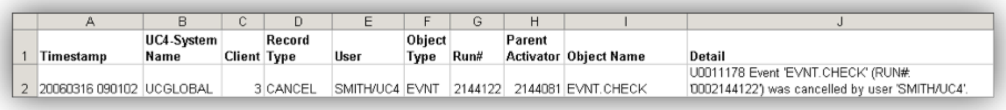
Modifications at runtime are logged. This includes modifications made via monitors or concerning states. In the case of JCL modifications, the JCL is not written to the revision report. It can be viewed in the object report.



	A	B	C	D	E	F	G	H	I	J
1	Timestamp	UC4-System Name	Client	Record Type	User	Object Type	Run#	Parent Activator	Object Name	Detail
2	20060316 090002	UCGLOBAL	3	RUN_MOD	SMITH/UC4	JOBP	2179084	2179055	MM.DAY	U0011523 Runtime modification: Task 'START' (RUN# '0000000000') was immediately started by user 'SMITH/UC4'.

Task Abortion

Aborted tasks are registered.



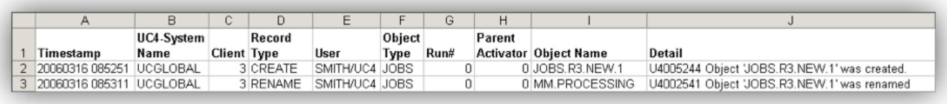
	A	B	C	D	E	F	G	H	I	J
1	Timestamp	UC4-System Name	Client	Record Type	User	Object Type	Run#	Parent Activator	Object Name	Detail
2	20060316 090102	UCGLOBAL	3	CANCEL	SMITH/UC4	EVNT	2144122	2144081	EVNT.CHECK	U0011178 Event 'EVNT.CHECK' (RUN# '0002144122') was cancelled by user 'SMITH/UC4'.

System Changes

All changes to objects are recorded (user, time, summary of change). Previous versions of objects are also stored so that they can be reverted to in the event of an erroneous change. The retention period of statistics and audit logs is configurable and may be exported to external systems for longer term storage.

Creating and Renaming Objects

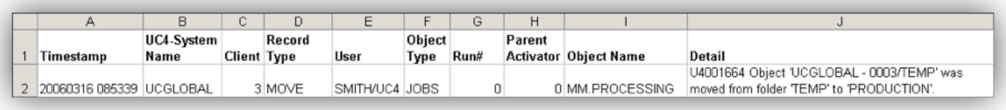
The creation and renaming of objects are logged.



	A	B	C	D	E	F	G	H	I	J
1	Timestamp	UC4-System Name	Client	Record Type	User	Object Type	Run#	Parent Activator	Object Name	Detail
2	20060316 085251	UCGLOBAL	3	CREATE	SMITH/UC4	JOBS	0	0	JOBS.R3.NEW.1	U4005244 Object 'JOBS.R3.NEW.1' was created.
3	20060316 085311	UCGLOBAL	3	RENAME	SMITH/UC4	JOBS	0	0	MM.PROCESSING	U4002541 Object 'JOBS.R3.NEW.1' was renamed

Moving Objects

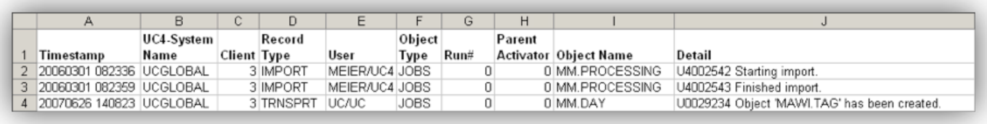
Source and target folders are recorded when objects are moved.



	A	B	C	D	E	F	G	H	I	J
1	Timestamp	UC4-System Name	Client	Record Type	User	Object Type	Run#	Parent Activator	Object Name	Detail
2	20060316 085339	UCGLOBAL	3	MOVE	SMITH/UC4	JOBS	0	0	MM.PROCESSING	U4001664 Object 'UCGLOBAL - 0003/TEMP' was moved from folder 'TEMP' to 'PRODUCTION'.

Imported and Transported Objects

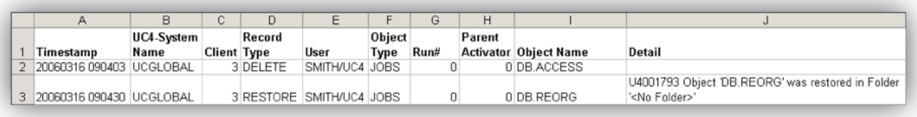
Import time and transportation time are stored. Contents of the XML and transport files are not written to the revision report.



	A	B	C	D	E	F	G	H	I	J
1	Timestamp	UC4-System Name	Client	Record Type	User	Object Type	Run#	Parent Activator	Object Name	Detail
2	20060301 082336	UCGLOBAL	3	IMPORT	MEIER/UC4	JOBS	0	0	MM.PROCESSING	U4002542 Starting import.
3	20060301 082359	UCGLOBAL	3	IMPORT	MEIER/UC4	JOBS	0	0	MM.PROCESSING	U4002543 Finished import.
4	20070626 140823	UCGLOBAL	3	TRNSPRT	UC/UC	JOBS	0	0	MM.DAY	U0029234 Object 'MAWI.TAG' has been created.

Deleted or Restored Objects

Deletion and restoring processes of objects are recorded.



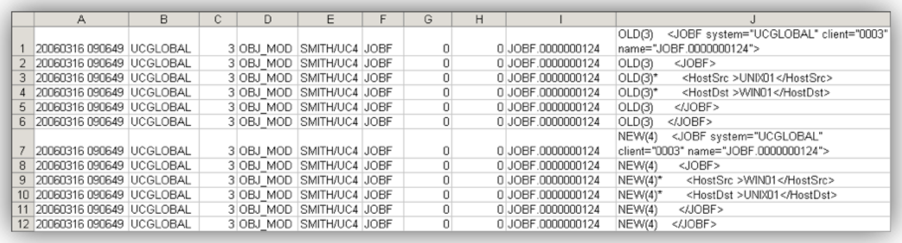
	A	B	C	D	E	F	G	H	I	J
1	Timestamp	UC4-System Name	Client	Record Type	User	Object Type	Run#	Parent Activator	Object Name	Detail
2	20060316 090403	UCGLOBAL	3	DELETE	SMITH/UC4	JOBS	0	0	DB.ACCESS	
3	20060316 090430	UCGLOBAL	3	RESTORE	SMITH/UC4	JOBS	0	0	DB.REORG	U4001793 Object 'DB.REORG' was restored in Folder '<No Folder>'

Object Modifications

Changes of object definitions are logged (e.g. modifications of priority, start type etc.).
Exceptions:

- Modifications made using an automation engine script
- Status modifications of sync objects,
- Contents of variable objects,
- Modifications of calendar objects.

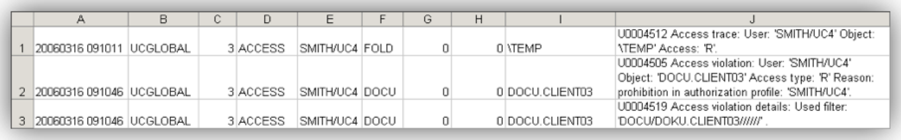
The revision report informs of changes. You can also view the new and previous values. The particular part of the object's XML structure which contains the modified attribute is output for this purpose.



	A	B	C	D	E	F	G	H	I	J
1	20060316 090649	UCGLOBAL	3	OBJ_MOD	SMITH/UC4	JOBF	0	0	JOBF.0000000124	OLD(3) <JOBF system="UCGLOBAL" client="0003" name="JOBF.0000000124">
2	20060316 090649	UCGLOBAL	3	OBJ_MOD	SMITH/UC4	JOBF	0	0	JOBF.0000000124	OLD(3) <JOBF>
3	20060316 090649	UCGLOBAL	3	OBJ_MOD	SMITH/UC4	JOBF	0	0	JOBF.0000000124	OLD(3)* <HostSrc >UNIX01</HostSrc>
4	20060316 090649	UCGLOBAL	3	OBJ_MOD	SMITH/UC4	JOBF	0	0	JOBF.0000000124	OLD(3)* <HostDst >WIN01</HostDst>
5	20060316 090649	UCGLOBAL	3	OBJ_MOD	SMITH/UC4	JOBF	0	0	JOBF.0000000124	OLD(3) </JOBF>
6	20060316 090649	UCGLOBAL	3	OBJ_MOD	SMITH/UC4	JOBF	0	0	JOBF.0000000124	OLD(3) </JOBF>
7	20060316 090649	UCGLOBAL	3	OBJ_MOD	SMITH/UC4	JOBF	0	0	JOBF.0000000124	NEW(4) <JOBF system="UCGLOBAL" client="0003" name="JOBF.0000000124">
8	20060316 090649	UCGLOBAL	3	OBJ_MOD	SMITH/UC4	JOBF	0	0	JOBF.0000000124	NEW(4) <JOBF>
9	20060316 090649	UCGLOBAL	3	OBJ_MOD	SMITH/UC4	JOBF	0	0	JOBF.0000000124	NEW(4)* <HostSrc >WIN01</HostSrc>
10	20060316 090649	UCGLOBAL	3	OBJ_MOD	SMITH/UC4	JOBF	0	0	JOBF.0000000124	NEW(4)* <HostDst >UNIX01</HostDst>
11	20060316 090649	UCGLOBAL	3	OBJ_MOD	SMITH/UC4	JOBF	0	0	JOBF.0000000124	NEW(4) </JOBF>
12	20060316 090649	UCGLOBAL	3	OBJ_MOD	SMITH/UC4	JOBF	0	0	JOBF.0000000124	NEW(4) </JOBF>

Accesses of Any Kind

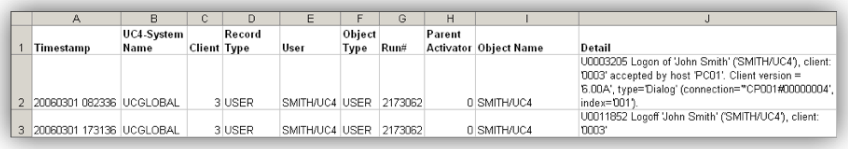
Accesses to objects and folders are recorded. This includes successful accesses and access violations which occurred due to restricted automation engine authorizations.



	A	B	C	D	E	F	G	H	I	J
1	20060316 091011	UCGLOBAL	3	ACCESS	SMITH/UC4	FOLD	0	0	\TEMP	U0004512 Access trace: User: 'SMITH/UC4' Object: '\TEMP' Access: 'R'.
2	20060316 091046	UCGLOBAL	3	ACCESS	SMITH/UC4	DOCU	0	0	DOCU.CLIENT03	U0004505 Access violation: User: 'SMITH/UC4' Object: 'DOCU.CLIENT03' Access type: 'R' Reason: prohibition in authorization profile: 'SMITH/UC4'.
3	20060316 091046	UCGLOBAL	3	ACCESS	SMITH/UC4	DOCU	0	0	DOCU.CLIENT03	U0004519 Access violation details: Used filter: 'DOCU/DOKU.CLIENT03//////' .

User Login/Logoff

Times of individual user logins and logoffs are also recorded.



	A	B	C	D	E	F	G	H	I	J
1	Timestamp	UC4-System Name	Client	Record Type	User	Object Type	Run#	Parent Activator	Object Name	Detail
2	20060301 082336	UCGLOBAL	3	USER	SMITH/UC4	USER	2173062	0	SMITH/UC4	U0003205 Logon of 'John Smith' ('SMITH/UC4'), client: '0003' accepted by host 'PC01'. Client version = '6.00A', type='Dialog' (connection='*CP001#00000004', index='001').
3	20060301 173136	UCGLOBAL	3	USER	SMITH/UC4	USER	2173062	0	SMITH/UC4	U0011852 Logoff 'John Smith' ('SMITH/UC4'), client: '0003'

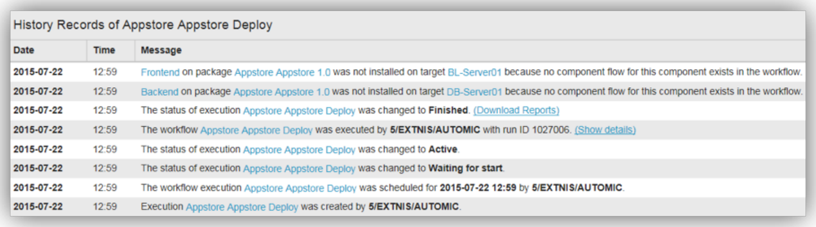
Release Automation

Each time entities change in ARA, log entries are written so that everything is documented. One log entry may be related to one or more entities and also displayed in the history of several entities. Viewing the history of any entity will therefore provide a lot of relevant information pertaining to its owner.

As an example, the following screenshot shows the history records of a deployment profile. The view presents the following columns:

- Date: Date when the event occurred
- Time: Time when the event occurred
- Message: Describes what happened to the object

The messages are grouped by date and sorted from the newest to the latest event.



History Records of Appstore Appstore Deploy

Date	Time	Message
2015-07-22	12:59	Frontend on package Appstore Appstore 1.0 was not installed on target BL-Server01 because no component flow for this component exists in the workflow.
2015-07-22	12:59	Backend on package Appstore Appstore 1.0 was not installed on target DB-Server01 because no component flow for this component exists in the workflow.
2015-07-22	12:59	The status of execution Appstore Appstore Deploy was changed to **Finished**. (Download Reports)
2015-07-22	12:59	The workflow Appstore Appstore Deploy was executed by **5/EXTNIS/AUTOMIC** with run ID 1027006. (Show details)
2015-07-22	12:59	The status of execution Appstore Appstore Deploy was changed to **Active**.
2015-07-22	12:59	The status of execution Appstore Appstore Deploy was changed to **Waiting for start**.
2015-07-22	12:59	The workflow execution Appstore Appstore Deploy was scheduled for **2015-07-22 12:59** by **5/EXTNIS/AUTOMIC**.
2015-07-22	12:59	Execution Appstore Appstore Deploy was created by **5/EXTNIS/AUTOMIC**.

Summary

At Automic security is a very important topic. We perform external and internal security audits on a regular basis. During the whole development lifecycle various checks are in place to ensure the best security of our products. Secure coding guidelines following industry standards (i.e. OWASP and SEI CERT coding standards) have been established by our application security team and are enforced throughout the software lifecycle. Furthermore, our product supports different functionality to support a secure operation such as:

- User, Groups and Privileges: The powerful access control concept of the automation engine allows administrators to grant users/user groups selective access to features, objects and views that they need and/or are allowed to see. A very granular set of authorization methods and privileges empowers the admin of each client to set up the access control to the system properly.
- Logins to Hosts, Databases and Applications: Furthermore, host, database, and application logins are objects in the automation engine. Login objects usually are defined by DBAs and system personnel. Passwords for the logins are encrypted and never displayed in clear type. Developers can use the logins without knowing the passwords, adding an additional layer of security.
- Multi-Tenancy: Automic is a native multi-tenant architecture and one single installation supports the enterprise. Multi-tenancy refers to and approach in which a single instance of the software centrally serves multiple client enterprises (tenants), highly segregated within a single instance for security and compliance.
- End-To-End Transparency, Audit Trails: The automation engine has a very fine granular authorization and permission system to allow actions just for privileged users. This ensures that the automation engine is completely revision secure and there exists the possibility to get audit reports for all activities within the system.
- Encryption: No external encryption solutions are required within an Automic environment. All necessary encryption is done natively via an AES key level of your choice (128, 192 or 256). This encryption is used for the following:
  - Password Storage within the Automic database repository
  - Database Password Reference within the automation engine configuration file
  - Communication between the automation engines and agents
  - User Interfaces and API Calls

For more information or a product demonstration please visit www.automic.com