b4A White Paper

Package Management – The Path to a Deployment Process

best-blu consulting with energy GmbH

Your contact
Andreas Büsching

E-mail a.buesching@best-blu.de
Phone +49 421 491 811 80
Date: May 27, 2025

Content

Motivation

The demand for automation has grown enormously in recent years. Not only has the number of automated processes increased, but the requirements have also become more demanding and complex.

Many of the automation environments based on Automic Automation have been in use in companies for many years and are trying to meet growing needs and requirements. With today's complexity, many of these environments are reaching their limits. There are many reasons for this. One factor that is usually involved is the growing importance and significance of automation systems and the processes they run. Errors in the processes or other instabilities in these environments now often have fatal consequences for a wide variety of areas within a company.

Major changes are needed to build an environment that meets these new requirements from old, established automation infrastructures and the associated development and testing procedures. These changes must be applied in various areas to create a modern, stable, and therefore high-quality automation platform.

This white paper addresses one of these topics and presents a solution. It describes a concept that establishes a basis for structured development with well-defined processes and also presents a method that enables a fully automated deployment process.

Current situation

Many environments have been part of the infrastructure for many years and are trying to adapt to constant changes and growing requirements. In this context, the further development of standards in automation has often fallen by the wayside, making it increasingly difficult to keep pace.

In order to bring high-quality and increasingly complex automation processes into production as quickly as possible, new concepts are needed for development, testing, and deployment. These must address the challenges and problems that exist in many Automic Automation environments today. The following are a few examples to illustrate the intentions behind these concepts.

Naming conventions

Rules for object names are primarily important in an Automic Automation environment in order to set up an authorization system. Furthermore, object names are unique per client and should provide information about the function and logical affiliation of an object. For these reasons, errors in object names are not just a cosmetic flaw, but can have more critical consequences. It is therefore important that object names are correct and that incorrectly named objects never reach production.

Typos when creating or renaming objects are a common cause of such errors. Since there is usually no review at this level, such errors are rarely detected.

Development guidelines (compliance)

There are usually many different ways to implement requirements in an Automic Automation environment. In many environments, you will therefore find a wide variety of solutions for the same problem. For developers, this means that they often have to familiarize themselves with solutions

developed by others more often than necessary. However, such a "free" form of development not only leads to increased effort in development, but can also make it more difficult to solve critical production problems.

Configuration

In environments where production and development are already separated, i.e., where there is more than one Automic Automation environment, the difficulty of dealing with different configurations will be familiar. Classic cases that frequently occur here are different paths on the file system, different names for agents, and different email addresses. However, there are many more variations. Examples include host names or URLs to external systems that are controlled by Automic Automation. Most solutions to this challenge are probably based on changing the exported objects in some way. This is done using specially written scripts or by making the changes manually each time after import. Another variant is the use of more complex include objects that adapt the configuration at runtime based on the environment in which they are running.

In practice, all these solutions pose risks that can compromise production stability due to incorrect changes or require additional resources at runtime, making development more difficult.

Deployment

All of the points mentioned above can affect the stable deployment of objects. However, there are other issues that can make deployment a dangerous operation. When deploying objects from Automic Automation, it is crucial to select the right objects. If some of the objects are also used by other processes, it must be ensured that they do not break compatibility with other objects in production. Since today's automation environments have tens of thousands or even hundreds of thousands of objects, it is almost impossible to maintain an overview. It is quite common for too few, too many, or even the wrong objects to be selected for deployment. The frequency depends on several factors. These include the complexity of the automation processes and the development processes themselves.

Reproducibility

This keyword is likely to be heard in every IT department these days and is a fundamental principle for every update. In many cases, it is also important to restore the original state as quickly as possible. With the tools provided by Automic Automation and integrated version management, it is entirely possible to restore old versions. However, there are limits to what can be done. Each object has its own version number, and some object types are frequently modified through manual reworking (e.g., variables and schedules). This means that restoration is feasible but can become a more complex task.

Summary

There is a clear commonality between the problems described above. All of these problems are related to manual work that is not subject to any or insufficient control.

Solution

The best4Automic Package Management defines a concept that organizes objects into logical groups and versions these groups. Such groups are called *b4A packages*. On this basis, functionalities are offered that address all of the problems described above and also offer additional capabilities.

For example, b4A provides a module that can be used to check both the naming convention and the development guidelines. The integrated configuration mechanism allows developers to define configurations for all environments and, if desired, for individual clients. The correct configuration is automatically imported into the client during deployment. For deployment, only *b4A packages* need to be selected, not individual objects. Dependencies between the *b4A packages* ensure that the installation in a target client does not cause any damage and that all necessary objects are available.

With best4Automic Package Management, it is possible to revert to an old version at any time and thus restore the exact state that existed before an update.

The configuration mechanism and additional capabilities cover some typical post-deployment tasks. For example, schedule objects can be updated and the reloading of definitions can be triggered. Outdated objects are automatically identified, renamed, and moved. Access data for login and connection objects can be restored during installation and can therefore also be transported. Together with other functions, this procedure offers solutions for many challenges in the area of automation process deployment.

Migration

Since every environment is different and the characteristics of automation also vary, there is no single path to package management. The following section outlines basic strategies that can be pursued during migration. All of these approaches have already been implemented in customer projects.

From the beginning

The simplest migration is no migration at all. When setting up an automation environment from scratch, the most important thing is to clearly and thoroughly define the fundamentals and configurations for package management. This includes development guidelines and processes that begin with the requirements and end with deployment in production. Clear guidelines with as few exceptions as possible are the most important factor for optimal and efficient implementation.

Side by side

This is one of the most commonly used strategies. Automation environments that have accumulated many problems over the years need a fresh start, if possible. This ensures that legacy issues do not continue to exist and thus do not stand in the way of rebuilding. There are two versions of this strategy, and both methods usually require very few additional resources.

New environment

Often in combination with an upgrade project, the opportunity is taken not only to set up the Automation Engine from scratch, but also to build development on completely new standards. From a certain point in time, all new developments are carried out based on b4A Package Management, and only the maintenance of old processes is still carried out in the old environment using old procedures.

The old processes are then gradually moved to the new environment and converted into b4A Packages.

New client

This technique is very similar to the above variant. However, in this case, instead of using a completely new environment, only an additional development client is set up that works according to the b4A Package Management concept. The migration process remains the same. This procedure facilitates synchronization between processes in the old and new environments.

Step by step

With this strategy, the new concepts and procedures are only slowly introduced in the same client where the old techniques are also used. However, the old objects are transferred to the new folder structures all at once. To do this, the objects are transferred to the new package folder structures in a clone client. An environment-specific solution must be found here that can be implemented with the support of b4A. The new structure can then be replicated in the main development client using a b4A module. After these steps, the objects have been transferred to the new folder structures, but the naming conventions are not yet correct, and some other package management features cannot yet be used. The remaining adjustments are made step by step and can be tackled whenever the objects of a package need to be changed due to requirements.

Conclusion

best4Automic Package Management provides a stable, well-defined basis for modern Automic automation environments, addressing common problems in automation environments and providing solutions. These are highly configurable and offer the flexibility needed to adapt to any environment.