Best-Practice Document

Elements of a Software Change Management Strategy

Version 1.1

Dietmar-Hopp-Allee 16
D-69190 Walldorf
December 2017

SAP PRODUCT(S) PRODUCT VERSION(S)
INDEPENDANT INDEPENDANT
OPERATING SYSTEM(S) DATABASE(S)
All All
ALM PHASE(S)
PLAN
SAP SOLUTION MANAGER 7.1 SP SAP SOLUTION MANAGER WORK CENTER(S)
All Change Management
Table of Contents

1 Introduction 3
2 Software Change Management Process 4
   2.1 Motivation 4
   2.2 Organizational Elements 4
   2.3 Release Management 6
      2.3.1 Overview and Benefits of Release Management 6
      2.3.2 Release Categories 8
      2.3.3 Quality Gates for Minor and Major Releases 9
      2.3.4 Release Calendar (per year) 10
      2.3.5 Considerations for Requirements Gathering 10
   2.4 Change Request Management 11
      2.4.1 Definition of Change Categories and Priorities 12
   2.5 Portfolio Management 14
3 SWCM Transport Landscapes 15
   3.1 Motivation 15
   3.2 Variant: 3-System Transport Landscape 16
   3.3 Variant: 4-System Transport Landscape 18
   3.4 Dual-Track Transport Landscape 19
      3.4.1 Variant: 5-System Dual Track Transport Landscape 20
      3.4.2 Variant: 6-System Dual-Track Transport Landscape (standard 3+3) 22
      3.4.3 Variant: 6-System Dual-Track Transport Landscape (4+2 with shared PRE) 23
      3.4.4 Dual-Track Cutover Process 25
      3.4.5 Retrofit in a Dual-Track Transport Landscape 26
   3.5 Variant: 4-System Transport Landscape with Multiple Production Systems 30
   3.6 Suggested Size of the Development and Test Systems 31
4 SAP Tools and Services for Software Change Management 33
   4.1 Motivation 33
   4.2 Overview of SAP Tools for Change Control Management 33
   4.3 Overview of SAP Services for Software Change Management 35
5 Abbreviations 36
1 Introduction

Enterprise change is continuous. Go to market strategies, regulatory compliance, manufacturing innovations and technology upgrades are a few of the drivers that are fueling continual change. Enterprises need to be prepared for, and plan for, continual change to remain competitive and compliant. Software change management (SWCM) must enable the enterprise to adapt effectively and efficiently. Essential for stability, and high quality in application solutions, are effective SWCM processes consisting of strong procedures, processes and tools, which plan for, manage and govern change.

SAP Support has worked with customers across the world in various industries to understand their IT objectives with respect to their SWCM strategy. The following is a list of the most common themes:

- Ease of use for business users: Meeting business requirements and empowering people to do their job efficiently and get the information they need
- Speed of deployment: Reducing the time to implement support for a new business requirement or a fix into production systems
- Quality of service: Minimizing failure of changes, service degradation, and unplanned outages
- Landscape manageability: Reducing landscape complexity to ensure solution availability and scalability while alleviating administrative burdens on IT
- Reasonable IT cost: Minimizing IT cost without compromising quality of service
- Keeping technology up-to-date: Ensuring that infrastructure and software are up-to-date enough to enable innovation and guarantee support from the vendors

While these IT objectives may not be new, changes in the business requirements or the solution landscape—along with suboptimal practices that persist over time—can make realizing them difficult at best.

Typical questions from customers are:

- Are the designed software change processes and the transport landscape effective? Do they support the desired release / maintenance strategy? Are they in line with industry best practices? How could the processes be improved for higher quality or higher agility for changes?
- Can the number of non-productive systems be reduced without impacting the quality of service? What is the “right” number of supporting systems? What is the best practice for developing multiple projects in parallel?

This document is a collection of observations with respect to software change management. It contains considerations for how to design and improve the software change management processes, the transport landscape, and how SAP Solution Manager can improve automating tasks and controlling the process.

It was built on the experiences the authors obtained from working with large enterprises as part of their SAP MaxAttention engagements. For feedback on this document, please contact monika.ahrens@sap.com.

The target audiences for this document are IT architects at SAP and at SAP customers who seek answers to the above mentioned questions about software change management.
2 Software Change Management Process

2.1 Motivation

The SWCM process is a key operations process for an SAP system. When implemented correctly, SWCM will mitigate the risks to production while enabling innovation to support the needs of the business.

The following seven elements are the fundamental backbone of a solid and stable SWCM:

- Organizational roles and responsibilities
- Portfolio management
- Requirements management
- Change Request Management
- Release management
- Test management
- Transport management

To achieve a high level of application quality and to provide agility and stability to production, these processes need to be defined well. When designing the processes and the supporting transport landscape, the requirements need to be analyzed, at least with respect to:

- Scope of changes/projects
- Expected time/speed and flexibility to ship a change/project
- Required quality of a change/project
- Cost for the landscape (hardware, labor)
- Testing requirements for change/project

Key questions are: What are the best practices for software change management? What are the recommended minimum processes to enable high quality, agility, and stability?

The following section gives an overview about the key software change management processes and Best Practices. The software change landscape is discussed in the “SWCM Transport Landscapes” chapter.

2.2 Organizational Elements

The starting point must be organizational elements. Senior management engagement, in the form of visible executive sponsors, and the definition and creation of the proper roles and responsibilities within the organization, with the correct level of authorization, is crucial to successfully implementing and managing SWCM. It is therefore incumbent on senior management to provide the leadership and vision to enable a strong SWCM organization and culture. As previously mentioned, holistically this will include requirements management, release management, maintenance management, test management, and landscape management.
2.2.1.1 Visible Executive Sponsorship

The **executive sponsor** is an individual or group of individuals who is responsible for making decisions related to the organization’s direction, strategy, and financial commitments, project expenditure, human resource commitments, and business processes. The executive sponsor must be available to promote as well as intervene on projects and releases. For example, an executive sponsor may need to intervene if an individual project’s scope creep is endangering the milestones of an entire release and, therefore, potentially impacting the business detrimentally.

Some responsibilities of the executive sponsor include:

- Create awareness and communicate within the organization the importance of change.
- Be the ultimate authority and responsibility deciding and communicating decisions.
- Ensure the IT teams and business units are aligned.
- Enlist the support of all management teams as change advocate.
- Ensure all necessary resources are available for successful delivery.
- Remove roadblocks.

2.2.1.2 Roles and Responsibilities

Roles and responsibilities are important, and need to be clearly defined and supported by the executive sponsor and the entire organization. Clarity of roles, as well as the relevant responsibilities, will have a big impact on how effective and successful SWCM will be.

Examples for key roles are:

- **Change manager**: Provides a single point of contact and is responsible for managing change and the impact on the environment. Responsibilities include:
  - Preside over change advisory board meetings.
  - Ensure received requests for changes (RFC) are documented and addressed.
  - Monitor the progress of change through production.
  - Validate and close with change advisory board.

- **Change advisory board** (CAB): Consists of individual stakeholders who have decision authority on the implementation of changes. CAB members should have a clear understanding of the business and technology requirements.
  - The CAB is led by the change manager.
  - The CAB will meet regularly to approve/authorize all major and medium change requests, review the closure of completed change requests, and review the performance indicators for the change process.
  - Participants need to be representatives from across the different divisions, understanding the business needs as well as dependencies and the impact and risk of a change.
• **Emergency change advisory board:**
  - For critical changes, an emergency CAB should be defined that will be able to convene and make expedient decisions about emergency change requests.

• **Release manager:** Oversees the major milestones of a release, such as development, testing, and deployment, and is accountable for the end-to-end life cycle and quality of a release. The tasks of a release manager include:
  - Release plan scope, execution, compliance, and delivery, in particular deliverables from multiple projects that must come together.
  - Resolve/escalate issues that will impact the release or release milestone.
  - Engage project teams and executive sponsor.

### 2.3 Release Management

#### 2.3.1 Overview and Benefits of Release Management

Release management is the process of planning, building, transporting, testing, and deploying software. Its purpose is to ensure that a consistent and cyclic reoccurring methodology of deployment is in place, and that it reduces the likelihood of errors in production through formal checks and procedures. A release is the collection of software, processes, and documentation that is required to implement one or multiple approved changes. Such a release is then treated as a single cohesive entity when it comes to development, testing, transporting, deployment, and ultimately the support effort.

The key objective of release management is to protect the live productive environment while enabling business as usual (BAU) corrections and enhancements to be deployed in a controlled manner. On-going, the releases will include:

- Major and minor business enhancements
- SAP Maintenance: support packages, enhancement packages, and upgrades
- Problem resolution development
- Emergency fixes

Figure 1 outlines the principles of release management, release planning, and a release calendar. In a release with multiple projects, as illustrated in Figure 1, all projects must align when the release enters or exits a quality gate or major milestone.

The uppermost arrow depicts a timeline for which there will be three major releases. Depicted in blue are four independent projects. The business owners, requirements analysis, change category, and prioritization, as well as the SWCM organization, will determine in which release a project will be deployed. In this example, projects 2, 3, and 4 will be deployed in major release 2. Notice how all three projects (2, 3, and 4), must align and commit to “Synchronized Transition to Testing” at the same point in time. All three projects must align when the release enters or exits a quality gate or major milestone (for example, end of development, testing cycles, and go live).
Note: For larger projects, which need to deliver functionality quickly and at different periods of time, and where possible, the scope of the project may need to be split by functions and managed within multiple releases, such as shown here for project 4.

Incident-related changes are often implemented by customers individually, even if the deployment is not urgent. Releases should be utilized for all change types. It is a best practice to differentiate between standard changes and minor and major releases to accommodate the requirements from production support and project development.

There are clear benefits of managing all changes in releases:

- A repeatable and reliable processes for releasing software
- A commitment to the business as to what goes live when
- The stability of the production system is increased because the frequency of transports to production is reduced, and a solid testing of the release components as one cohesive work package is ensured.
- Quality gates as checkpoints to sign off on milestones
- The overall costs for testing can be reduced by using common regression and integration testing for several projects/changes as one cohesive work package.
- Risk of inconsistencies is reduced, for example, due to "forgotten transports" or sequence violations.
- Administration efforts for transport management are reduced as all projects move from one phase to another at a single point in time.
2.3.2 Release Categories

Changes need to be allocated/bundled into a release category. Essentially, there are generally four release categories:

- **Major release**: Has a significant release window intended for the introduction of major new functionality consisting of a larger number of new developments. The proper amount of time and resources need to be allocated to complete the required regression, user acceptance, and performance testing.

- **Minor release**: Has a much shorter release cycle and is intended for minor enhancements and problem resolution fixes. Due to the much shorter release cycle of a minor release, there will be insufficient time for full regression and user acceptance testing as conducted in a major release. A limited but targeted testing strategy is needed.

- **Standard changes**: Is low risk and the impact is clearly understood, such as certain configuration changes. Standard changes are preapproved and don’t need to be assessed in Change Request Management.

- **Emergency changes**: Should contain only priority-one changes that will be transported immediately to solve critical production issues.

Figure 2 illustrates a best practice example for release categories.

<table>
<thead>
<tr>
<th>Transport Cycle every</th>
<th>Emergency Changes</th>
<th>Standard Changes</th>
<th>Minor Release</th>
<th>Major Release</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transport Cycle every</td>
<td>Emergency only</td>
<td>Standard changes only (non invasive, low risk and impact well known)</td>
<td>Non-invasive changes (+ Re-Import of Emergency Changes)</td>
<td>All types of changes including invasive changes</td>
</tr>
<tr>
<td>Change Categories</td>
<td>Emergency only</td>
<td>Standard changes only (non invasive, low risk and impact well known)</td>
<td>Non-invasive changes (+ Re-Import of Emergency Changes)</td>
<td>All types of changes including invasive changes</td>
</tr>
<tr>
<td>Priorities</td>
<td>Emergency only</td>
<td>Normal Priority</td>
<td>Normal Priority</td>
<td>Normal Priority</td>
</tr>
<tr>
<td>Test focus</td>
<td>Regressions for core processes (automated tests)</td>
<td>Test identified in pre-approval. Regressions for core processes (automated tests)</td>
<td>New features and corrections. Regressions for core processes (automated tests)</td>
<td>Complete test scope</td>
</tr>
<tr>
<td>Examples</td>
<td>Any changes to solve high priority incidents</td>
<td>Already approved changes e.g. storage locations, MRP controller.</td>
<td>Non-critical configuration, medium priority incidents</td>
<td>New (major) functions, Support / Enhancement Packages, Upgrades, medium/low priority incidents</td>
</tr>
</tbody>
</table>

Figure 2: Release Categories - Best Practice Example

The test strategies (scope, effort and duration) are different per release category according to the level of changes. Figure 3 illustrates the focus and prioritization of test activities.
Deciding which release to assign a change to has to be made during Change Request Management. Criteria include criticality of the change and change priority; high-risk changes need more testing and thus are natural candidates for a major release cycle.

One specific aspect in the release planning process is the integration of SAP maintenance activities, such as SAP support packages, enhancement packages, or upgrades. These can either be treated as part of a major release, together with other projects, or as a separate major release.

- **Handling SAP maintenance events in dedicated major releases automatically leads to the minimal import time and system downtime.** Further downtime optimization methods can be applied very effectively if needed. If minimization of import time of a single release is the most important criterion, then this is the recommended approach.

- **Combining SAP maintenance events with other projects minimizes the overall testing effort (one test for both projects) and reduces risk (the maintenance event is already tested extensively during the release development).** The most common procedure is to implement the support package/enhancement package at the beginning of the development cycle of the next major release.

### 2.3.3 Quality Gates for Minor and Major Releases

There should be at least three quality gates for a minor or major release:

- **Scope to build.** After the definition of the requirement and before the build is started, all requests for changes need to be approved, with the exception of standard changes (preapproved category of changes).
• **Build to test**: When development is completed (including developer/unit test) and before the release is tested, it is recommended to assess the status of the release and make sure that only transports with sufficient quality are moved to the testing system. This quality gate is optional for smaller changes.

• **Test to deploy**: Based on the test results before the deployment, the final go/no-go decision for deployment is made. This is recommended for all changes.

• Large-scale changes and projects can have additional quality gates.

### 2.3.4 Release Calendar (per year)

A release calendar is a planning tool that announces to the enterprise what to expect and when. Ideally, a release calendar consisting of all hardware and software activity will be maintained for the enterprise.

Once release categories have been defined, a release calendar inclusive of major, minor, and maintenance events needs to be completed and communicated to the enterprise.

It is a best practice to align the release go-live dates across all SAP applications within the environment, for example, one go-live weekend for both SAP ERP and SAP APO within a region in case of regional implementations.

The release calendar needs to be aligned with the business on freeze periods, downtime scheduling, and frequency of shipment of changes. It should be defined for at least one year and reviewed regularly.

The release calendar should also include cutoff days, CAB meetings, and testing periods.

![Illustrative Calendar Year](image)

**Figure 4: Schematic Illustration of a Release Calendar**

### 2.3.5 Considerations for Requirements Gathering

When defining the release strategy and the supporting test strategy and landscape, the first step is to analyze the requirements for SWCM. The more accurate the analysis, the better the determination will be for impact analysis, development longevity, costing, and delivery of the solution. Typical questions, listed here, help you to get a better understanding:

• What is the level of new project developments?
2.4 Change Request Management

Change requests can occur daily and have various input sources, for example, a problem to be solved because of an incident (break-fix), new business enhancement, or a change to an existing project.

All requests for change should be documented in a standardized way and include the necessary information for the change manager to make proper decisions.

- Often, a request for change has three sections: general information to be entered by the submitter, evaluation on the impact by the change manager and respective testing scope/effort done by a senior person, and the decision about the request.

One essential step in the Change Request Management process is the assessment and authorization step early in the process. This step includes:

- The definition of the required test scope for a change request
- The central categorization and prioritization of the change requests according to the defined change categories (defining the change impact) and change priorities
- The assignment of the change request to a release category, based on the change category and priority
- Agreement on a specific go-live date for the change according to the release category and calendar
2.4.1 Definition of Change Categories and Priorities

In the previous section, it was already mentioned that only changes of a certain category and priority can go into a certain release category. It is very important that the change categories and priorities are defined explicitly (“positive lists”) up front, avoiding any discussion in the day-to-day Change Request Management process.

Change categories:

- Change categories reflect the impact and the risk associated with a certain change. Examples:
  - Critical (invasive): Any configuration change or development that will or has the potential – from either business or technical side – to interrupt business-critical services (for example, changes to core transactions, adaptations to user interfaces, changes in common elements of business processes, and so on).
  - Non-invasive: Any change that is not likely to affect the availability of one or more entire business-critical services and that can easily be backed out/recovered in the event of an issue.
  - Standard: Repetitive non-invasive changes with known outcomes and defined procedures to implement (for example, new master data type configuration, like storage location).
- The criteria for how to classify a change request should be explicitly defined, for example, based on a number of end users/countries/business units affected, need for training of end users, invasiveness.
of the change (new, enhanced, changed process), magnitude of the change (single module, single application, multiple applications), expected effort to build.

- For each change category, levels of decision makers should be defined.
- It is recommended to define the category “standard change” as this helps to reduce the assessment and approval load.
  - The definition of a new standard change type includes the definition of the allowed objects and required test and validation steps.
  - Create a standard change list, including owners of standard changes and a list with preapproved who can create and/or release transports containing standard changes.
  - Once a standard change type is approved, a request of this type does not need to be assessed and approved. It will follow the predetermined, documented workflow including a well-defined test scope.

Change priorities:

- The change priority indicates how urgently the change needs to be applied. The best practice is to start with two levels of priority to keep the process simple.
  - Emergency: Changes addressing priority-one incidents. In case of a priority-one incident, a severe outage is affecting multiple business processes, or an entire business unit or one of the mission-critical business processes is not available or impacted.
  - Normal: Changes of normal priority are those that can be processed in line with the change category definitions and associated target approval/authorization/implementation schedules. Critical business processes are not impacted or a workaround is available.

- A fast-track process for urgent requests should be defined and used. A typical shortcut is that the change manager directly approves the request and later asks the CAB for approval in a regular CAB review meeting. Alternatively, an emergency CAB would have to approve an urgent request to be built. In both cases, at least a basic assessment of effort and impact is necessary.

The criteria defined in this chapter need to be used to decide to which release category a certain change is assigned.

<table>
<thead>
<tr>
<th>Mapping of change categories and priorities to release categories</th>
<th>Release Category</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Emergency Changes</td>
</tr>
<tr>
<td>Invasive</td>
<td>(X)</td>
</tr>
<tr>
<td>Non-invasive</td>
<td>X</td>
</tr>
<tr>
<td>Standard</td>
<td>-</td>
</tr>
<tr>
<td>Emergency</td>
<td>X</td>
</tr>
<tr>
<td>Normal</td>
<td>-</td>
</tr>
</tbody>
</table>

Figure 7: Mapping of change categories / priorities to release categories
2.5 Portfolio Management

The scope of a major release is defined by a portfolio management process (not by handling each single request for new functionalities as a request for change). The objective of portfolio management is to identify the new development projects, prioritize them with the business according to business benefits and with IT according to IT benefits, and manage the related risks.

Once a program or project is approved, the related projects are handed over to the project management organization for detailed planning and realization.

Portfolio or multi-project management takes care of high-level project management tasks like planning, controlling, and reporting of multiple development projects.

Besides changes that are implemented by development projects, there are updates necessary for production support. Operations, which are usually managed by following ITIL processes, regularly create bug fixes, continuous improvements, or standard changes.

Project developments and production support might “fight” for the same resource, such as development/test systems or people. An isolated view on production support vs. project work will never cover the complete picture and has some risks, such as:

- Projects might be delayed because key resources are blocked to solve problems critical for production.
- The same object is processed multiple times within a time frame, which not only leads to the risk of blocking it, but also to the risk of transport sequence violation and object downgrade. Thus additional work is needed to govern multiple transport requests on the same object. Additionally, testing such changes either can become more complex or even irrelevant (for example, a scenario is tested that will not be imported later into production).

To overcome such conflicts, an integrated portfolio management process is recommended, which can harmonize the incident and change management processes based on common criteria for the company.

Aspects to be covered here are an assessment based on unified criteria and weighting, a regular synchronization between operational requirements and project management, and holistically defined processes for planning, development, and deployment.

Having a strict release management process in place will ease the coordination of requests.
3  SWCM Transport Landscapes

3.1  Motivation

Experience shows that without a strong SWCM strategy in place, the transport landscape can become overly complex and increasingly more difficult to maintain as the number of instances may fluctuate and the actual instance roles change. Individual projects, for example, that are allowed to affect the transport landscape to support specific project phases such as development or testing will often lead to such complexity and fluctuation. This type of landscape manipulation/juggling introduces risk, puts a strain on IT and its skills (continually standing up – and down instances) and, as the number of projects and their demands grow, is not sustainable.

What is the best transport landscape to support the company’s software change management requirements?

Figure 8 is an overview of some of the transport landscape strategies that we will discuss in more detail.

<table>
<thead>
<tr>
<th>3-System Landscape</th>
<th>Supports enterprises during initial implementation and also production support where change volume is low.</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-System Landscape</td>
<td>To reduce risk and increase flexibility, an additional testing instance is added. PRE is a pristine testing environment where release changes can be imported and tested in isolation.</td>
</tr>
<tr>
<td>5-System Landscape</td>
<td>To further mitigate risk, a second development track can be introduced to separate production support from project development. Production in PSD, projects in DEV track.</td>
</tr>
<tr>
<td>6-System Landscape</td>
<td>Similar to the 4-tier landscape, an additional testing instance can be added to the project landscape (and potentially for the production support track) to isolate release changes and improve testing capabilities.</td>
</tr>
</tbody>
</table>

Figure 8: Overview of transport landscapes

Some key factors for deciding a transport landscape strategy:

- **Risk mitigation**: For certain industries, but most especially regulated industries, the customer will add additional instances to extend development and testing capabilities in an endeavor to deliver flawless change.

- **Flexibility**: How fast can the SWCM solution implement change to satisfy the business go-to-market needs and allow IT to fulfill its responsibility of ensuring continuous quality with the appropriate resources.

- **Demand**: Scope and duration of active projects, including maintenance. Development projects that have extended implementation periods, such as functional implementations and maintenance upgrades, have different demands and requirements than ongoing, stable solutions.

- **Overall cost**
The weighting of these factors is different from company to company – and even from solution to solution within a company – depending on the individual requirements. So there is no “one size fits all”. Examples:

- The requirements for software change management are different based on the status of a given solution. A solution may be in an early phase with heavy developments or there may even be heavy developments in parallel to production support; other solutions may be very mature with focus on continuous improvements to a productive system.
- The requirements for software change management may be different based on the business criticality of a given solution (and required stability and availability of the solution).
- The requirements may be different based on the size of the systems, for example, production systems processing high volumes may require a dedicated load test system.

The following is a more detailed discussion of the most common SWCM transport landscapes and the high-level pros and cons:

- Classical 3-system transport landscape
- Enhanced 4-system transport landscape
- Dual-track (N+1) transport landscapes with 5 or 6 systems

### 3.2 Variant: 3-System Transport Landscape

![Figure 9: Classical 3-System Transport Landscape](image)

Concept of the 3-system transport landscape:

- All changes are created in DEV (development system).
- All tests are completed in QAS (quality assurance system) before transports are imported into PRD (production system).
- The SBX (sandbox) is intended to be temporary and used for prototyping new functionality, sensitive updates such as OS/DB upgrades, and SAP maintenance. There is no transport path between the sandbox and the DEV system.

After the initial implementation of the SAP solution, the 3-tier landscape is best suited to those customers that do not require additional major developments, that is, customers that have entered into a mature production support state and need only to update production periodically with minor enhancements and corrections.

This is due to some limitations a single testing instance poses. For example, it is not suited for testing multiple concurrent changes/projects with different go-live dates. In such a scenario, changes belonging to different releases might be tested in an incorrect state, leading to potential errors in production (see also Figure 12). Therefore, strict SWCM controls need to be in place to govern transports and testing.
SAP maintenance events, support packages, and enhancement packages must also be considered. In a 3-tier environment, production support and the necessary freeze periods will differ than in, for example, an N+1 environment. Once maintenance is moved to the DEV environment, there needs to be a production freeze, as DEV will be at a different maintenance level than QAS and PRD until the maintenance has been transported throughout the landscape. Customers circumvent these deficiencies by introducing temporary instances and then deleting them upon completion of the project. Each solution has its nuances and it is necessary to consider these while planning.

The 3-tier landscape is the least complex in terms of infrastructure, but will require strong SWCM processes to manage change successfully.

Recommended risk mitigation for this landscape:

- Set up a strict release management process, that is, all changes/projects should go live at the same release date and, ideally, projects for the next release will not have started or, in the event of conflict, will be staggered in the SBX.

- Implement a strict change management process for production support and project development. Ideally, the same development team is responsible for both types of changes.

- Put in place a strong change control. For example, do not allow any auto-import into QAS, constantly monitor for over taker and parked transports or abandoned transports.

- Figure 10 list the advantages and disadvantages of a 3-system transport landscape.

<table>
<thead>
<tr>
<th>PRO</th>
<th>CON</th>
</tr>
</thead>
</table>
| Quality | • Testing for projects and production support in parallel in QAS, i.e. test environment for production support “polluted” with project development objects.  
  ➔ Test results are not reliable: risk that production support change works in QAS, but not in PRD  
  ➔ Higher risk of downgrades (sequencing problems)  
  ➔ Risk that QAS is not “stable” enough for fast testing of e.g. urgent corrections.  
  ➔ Final project testing ideally needs a code freeze for production support.  
  • Overlapping projects with different release dates may be tested in parallel in QAS.  
  ➔ Testing environment does not represent PRD. Risk that release works in QAS, but not in PRD. |
| Time | • No need to cut over between transport tiers nor many systems have to be passed for the transports  
  ➔ Changes can be deployed fast |
| Cost | • Minimum amount of systems  
  ➔ Lowest cost for hardware and system maintenance (operate, refresh, …)  
  • Development for projects and production support in single DEV. In case of conflicts, project changes can be blocked or have to be backed-out temporarily  
  ➔ Risk of low speed for individual corrections. |

Figure 10: Classical 3-System Transport Landscape - Advantages and Disadvantages
3.3 Variant: 4-System Transport Landscape

Figure 11: 4-System Transport Landscape

Concept of the 4-system transport landscape:

- All changes are created in DEV (development system).
- The QAS (quality assurance) system is used as a testing environment for integration testing and data conversion testing. This system can, depending on project requirements, contain the changes of many in-flight developments.
- The PRE (pre-production) system is a pristine environment to isolate and test changes belonging only to the next release. Ideally, a recent copy from production, to which only the next release of changes are be imported, will enable test management to run final test scenarios such as integration, regression, user acceptance, and performance testing.
- The SBX (sandbox) system is intended to be temporary and used for prototyping new functionality, sensitive updates such as OS/DB upgrades, and SAP maintenance. There is no transport path between the sandbox and the DEV system.

The 4-tier transport landscape adds additional flexibility to the solution by introducing an additional testing environment PRE, which is intended to improve the SWCM release processes needed to manage multiple concurrent projects. Where enterprises have multiple concurrent projects in development at the same time, and with different release schedules, it is necessary to isolate and test only those changes moving to production with the next release cycle. In the following example, only the RED project is moving to production with the next release cycle. Any potential dependencies/handshaking defects the RED project may have with any of the other developments can only assuredly be detected during qualified and isolated testing in the PRE instance.

Figure 12: Illustration of Staggering Releases in a 4-System Transport Landscape

Figure 12 illustrates PRE and isolated testing:

1) In DEV there are four separate independent projects. Only the RED project is to move to production with the next release cycle.

2) At various stages, three of the four projects are transported to QAS to start various testing scenarios.

3) Prior to the next release window, PRD should be copied into PRE.
4) You then transport the RED project into PRE and conduct the necessary final regression and user acceptance testing. Any potential dependencies/handshaking defects the RED project may have with any of the other developments can only assuredly be detected during qualified and isolated testing in PRE.

5) After successful testing in PRE, you can transport the RED project to production.

The 4-tier landscape is still relatively low in cost and reduces the risks when project activity/scope is high. The main benefit of the landscape is that it allows for staged testing, as it isolates changes being prepared for import to production. Consequently, this landscape is the best practice landscape for solutions with production support and development projects of a medium scope (even parallel releases).

As with the 3-system transport landscape, the same risk mitigation measures apply with respect to governance and control of transports:

- Implement a strict change management process for production support and project development. Ideally, the same development team is responsible for both types of changes.
- Put in place a strong change control process in place, in particular, a strict control of when and which transports move into PRE and define a PRE system refresh strategy.

Figure 13 lists the advantages and disadvantages of a 4-system transport landscape.

<table>
<thead>
<tr>
<th>PRO</th>
<th>CON</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Quality</strong></td>
<td>PRE represents the software status of PRO and is a pristine environment to isolate and test changes belonging to the next import to production (change or release). → Reliable results for regressing testing for production support</td>
</tr>
<tr>
<td>Time</td>
<td>No need to cut over between transport tiers nor many systems have to be passed for the transports → Changes can be deployed fast</td>
</tr>
<tr>
<td>Cost</td>
<td>Low amount of systems → Low cost for hardware and system maintenance (operate, refresh, …)</td>
</tr>
</tbody>
</table>

Figure 13: 4-System Transport Landscape - Advantages and Disadvantages

3.4 Dual-Track Transport Landscape

The dual-track transport landscape, also known as N+1, was designed for customers who need to continually release a significant number of software-updates on a regular basis and provide a secure and stable production environment for the end users.

The advantage of dual-track transport landscape is the addition of a parallel track to production, which contains a second development and testing instance(s) that will be utilized continuously for developing,
testing, and releasing software to the production track, in release waves, according to the release calendar. This design has a salient and distinctive advantage over the single track in that it enables the customer to move the entire project development scope and all project development teams away from production – a clear segregation of development tasks from production tasks and its personnel. By minimizing the change activity in the production track and the personnel that can make those changes, it better empowers the production support team with its risk management.

The dual track allows the project teams much more flexibility, as they are not constrained by strict production controls. For example, the path to production will not be blocked by project testing requirements or maintenance updates. Testing instances can be refreshed as often as the projects demand. More realistic quality gates can be enforced and not circumvented by production priorities; for example, testing duration and its iterations can be driven by the needs of the projects and not by a production-imposed freeze period.

The dual-track strategy puts in place a permanent set of instances that have a consistent and permanent role, ensuring the SWCM teams can rely on knowing which tasks are performed when and where. This is an advantage over the single track. A single track that handles large developments and maintenance usually requires the Basis team to intervene manually, and put temporary instances in place. This intervention introduces more risk, as these changes require the SWCM processes, such as transport paths, to also change.

In essence, the dual track becomes a permanent software factory, continuously releasing (cutover) new functionality to production according to the release calendar.

Ensuring the dual-track development instances are synchronized is an important task. The dual track landscape should not be put in place without a clear concept for a) retrofit of changes from production support and b) cutover to production.

The above reasoning does not, however, imply that a dual-track transport landscape is a must-have in case of large developments for a landscape. There are a number of SAP customers that run all of their production support and project developments very successfully in a single-track transport landscape; however, this requires very mature organizations and processes. A dual-track transport landscape can help mitigate risks for production from project development, for example, if the solution is still in full rollout mode, there are different organizations with different skills involved, a lot of invasive transports are expected, or there is high risk sensitivity in the company, the industry, or the particular solution.

3.4.1 Variant: 5-System Dual Track Transport Landscape

![Dual-Track Transport Landscape (with 5 Systems)](image)

Figure 14: Dual-Track Transport Landscape (with 5 Systems)
Concept of the 5-system dual-track transport landscape:

- PRD (production instance)
- There are two development and two quality assurance instances:
  - PSD (production support development)
  - PSQ (production support quality assurance)
  - DEV (project development)
  - QAS (project quality assurance)
- The SBX (sandbox) is intended to be temporary and used for prototyping new functionality, sensitive updates such as OS/DB upgrades, and SAP maintenance. There is no transport path between the sandbox and the DEV system.

As a permanent instance strategy, the 5-system dual track is more costly than a single track in terms of the number of instances and the administration effort. Arguably, however, the single track to manage the same workload may cost more and will introduce more risk, as manual intervention is required to put temporarily solutions in place to manage special events such as maintenance.

This instance strategy puts in place a more secure production support track that is shielded from major development activities taking place in the project development track.

The limitation of this instance strategy, unlike the 6-system dual-track transport landscape, is it lacks testing capacity to manage more than one release at a time (see also Figure 17 for how the 6-system landscape can be used to stagger releases).

The advantages and disadvantages of this landscape are summarized in the following table:

<table>
<thead>
<tr>
<th></th>
<th>PRO</th>
<th>CON</th>
</tr>
</thead>
</table>
| Quality | • Good separation between projects and production support, both for development and testing  
  ➔ Reliable results for regressing testing for production support in PSQ  
  ➔ Lower risk of downgrades  
  ➔ Parallel tests of projects in QAS | • In case of overlapping projects with different release dates and these projects tested in parallel in QAS.  
  ➔ Testing environment does not represent PRD. Risk that release works in QAS, but not in PRD.  
  • Depending on cut-over variant code freeze for production support still necessary. |
| Time   | • No conflicts in PSD for corrections that touch objects in project development  
  ➔ Production support changes can be deployed fast | • In case of overlapping projects with different release dates and only next release being tested in QAS.  
  ➔ Projects in subsequent releases cannot start user and integration testing before go live of the previous release. |
| Cost   | • Higher amount of systems  
  ➔ Higher cost for hardware and system maintenance (operate, refresh, …)  
  • Effort for retrofit, i.e. conflict resolution in DEV  
  ➔ Effort might be higher compared to single track (only shift of time when conflict needs to be solved) |
3.4.2 Variant: 6-System Dual-Track Transport Landscape (standard 3+3)

![Diagram](attachment:image.png)

Figure 16: Dual-Track Transport Landscape (with 6 Systems, standard 3+3)

Concept of the 6-system dual-track transport landscape:

- PRD (production instance)
- There are two development and three quality assurance instances
  - PSD (production support development)
  - PSQ (production support quality assurance)
  - DEV (project development)
  - QAS (project quality assurance)
  - PRE (project quality assurance)

The 6-system dual-track transport landscape differs from its 5-system dual-track variant in that it adds an additional testing instance (PRE) to the project development track to better enable staggering, testing, and managing multiple projects with different go-live dates. Figure 17 illustrates this:

![Diagram](attachment:image.png)

Figure 17: 6-System Dual-Track Transport Landscape (standard 3+3) - Staggering Releases

Explanation of how to work with multiple releases in parallel:

1. Red release: In agreement with the SWCM release calendar, the red release is scheduled to move to PRE for final testing. At this stage, and as part of the project quality gates, sufficient testing cycles have been executed in QAS to correct all major defects a project may have. Otherwise the project is pulled from the release, or the Executive Sponsor is engaged. Prior to transporting all projects, developments, configuration etc. belonging to the red release, you copy production PRE into the project development instance PRE. The red release can now enter its final regression, user acceptance and performance testing scenarios before being cutover to production.
2. Blue release: Has completed the bulk of development and is in scenario testing in QAS correcting defects DEV

3. Pink release: Primarily in DEV development phase, initial QAS testing for some developments

4. Orange release: Often, development teams are constrained by project conflicts belonging to a prior release. Customers often utilize the SBX so development/configuration teams can be productive with initial proof of concepts, including coding designs. Note that the SBX is not in the transport path and no transports will be moved from SBX forward. There are other tools to capture work done.

The 6-system dual track transport landscape and the proper usage of SWCM processes should manage most customers SWCM requirements. For this reason we generally do not recommend another track, that is, a third development environment (N+2).

The advantages and disadvantages of this landscape are summarized in the following table:

<table>
<thead>
<tr>
<th>PRO</th>
<th>CON</th>
</tr>
</thead>
</table>
| **Quality** | • Good separation between projects and production support, both for development and testing.  
→ Reliable results for regressing testing for production support in PSQ  
→ Lower risk of downgrades  
→ Parallel tests of projects in QAS  
• PRE represents the software status of PRO and is a pristine environment to isolate and test changes belonging to the next release.  
→ Reliable results for final release testing in PRE in case of overlapping projects  
→ Lower risk of downgrades  
→ Parallel tests of projects for go live in PRE and projects in development in QAS | • Depending on cut-over variant code freeze for production support still necessary. |
| **Time** | • No conflicts in PSD for corrections that touch objects in project development  
→ Production support changes can be deployed fast |
| **Cost** | • Higher amount of systems  
→ Higher cost for hardware and system maintenance (operate, refresh, …)  
• Effort for retrofit, i.e. conflict resolution in DEV  
→ Effort might be higher compared to single track (only shift of time when conflict needs to be solved) |

Figure 18: 6-System Dual-Track Landscape (standard 3+3) - Advantages and Disadvantages

### 3.4.3 Variant: 6-System Dual-Track Transport Landscape (4+2 with shared PRE)

![Diagram](image_url)

Figure 19: Dual-Track Transport Landscape (with 6 Systems, 4+2 w. shared PRE – Normal Transport Path)
Figure 20: Dual Track Transport Landscape (with 6 systems, 4+2 w. shared PRE – transport path during major release testing)

This variant of the standard 6-system dual-track transport landscape is a hybrid scenario of the 4-system transport landscape and the dual-track transport landscape:

- Key idea of this variant: the PRE-production system is used for regression/release testing for both production support changes as well as for the project developments.

- There is a 4-system transport landscape used for production support (and minor releases). All changes are regression tested in the PRE system, which is part of the production support landscape in the “normal” transport path.

- Major projects are developed in the project development landscape. During release and regression testing of a major release, the PRE system in the production support track is used for this task exclusively.
  - Before release testing, the transport path for production support is changed (see orange line in Figure 20): changes from PSD go via PSQ to PRD and bypass PRE.
  - The release scope of the major project is imported into PRE and regression tested in PRE.
  - After successful testing of the major release, the release is imported into production, and then PSD and PSQ are updated. Finally, the transport path is switched back to “normal” mode.

Prerequisites for this landscape:

- Very mature change management processes: As this option allows you to do minor releases in the production support landscape, there needs to be a very good categorization between which changes go into minor vs. major releases. Often it is simpler from a governance perspective, and roles and responsibilities for development, to have all project developments in the project development track.

- During regressing/release testing of a major release in PRE with the reduced 3-system landscape in production support, there should only be production support changes and no minor releases.

This type of dual role of the PRE system is—from a cost perspective—much preferred to the alternative option of a 7-system landscape (with 4+3 systems).

The advantages and disadvantages of this landscape are described in the following table:
### 3.4.4 Dual-Track Cutover Process

Cutover is the process needed in a dual-track landscape to move the next release from the project development track to the production support track for go live. The goal of cutover is to update the production track expeditiously, limiting the impact to production and its supporting instances. Testing, for example, is not foreseen as a function of cutover because this would considerably increase the time needed to cutover and would block or complicate production emergency fix capability. Some considerations for cutover are planned downtime requirements, how to manage production emergencies during cutover, and how long the complete process takes.

A requirement of the cutover plan is to have all instances in the production track be updated from the project track. The dilemma is when the downtime window is insufficient to complete the updates to all instances, necessitating that the cutover to be staggered. SAP recommends two options when staggering the cutover.

Option 1 as illustrated in Figure 22 is to cut over from the project landscape (from PRE in case of a standard 6-system landscape; from QAS in case of a 5-system landscape) into production development PSD, and from there to PSQ and finally PRD.

---

<table>
<thead>
<tr>
<th>PRO</th>
<th>CON</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Quality</strong></td>
<td>• PRE represents the software status of PRD and is a pristine environment to isolate and test changes for the next import to production.</td>
</tr>
<tr>
<td></td>
<td>→ Reliable results for regression/release testing in PRE in case of overlapping projects</td>
</tr>
<tr>
<td></td>
<td>→ Lower risk of downgrades</td>
</tr>
<tr>
<td></td>
<td>→ Parallel tests of projects for go live in PRE and projects in development in QAS</td>
</tr>
<tr>
<td></td>
<td>• During major release testing still 3 system landscape for production support</td>
</tr>
<tr>
<td></td>
<td>→ No code freeze for production support</td>
</tr>
</tbody>
</table>

| **Time** | • No conflicts in PSD for corrections that touch objects in project development |
| | → Production support changes can be deployed fast |
| **Cost** | • One PRE shared for production support and project development |
| | → Significantly lower costs compared to a 7-system landscape (4+3) with two pre-production systems. | • Higher amount of systems |
| | | → Higher cost for hardware and system maintenance (operate, refresh, …) |
| | | • Highest effort for retrofit (with minor releases in PSD), i.e. conflict resolution in DEV |
| | | → Effort might be higher compared to single track (only shift of time when conflict needs to be solved) |

---

![Figure 21: 6-System Dual-Track Landscape (with 6 Systems, 4+2 w. shared PRE) - Advantages and Disadvantages](image)

**Figure 21:** 6-System Dual-Track Landscape (with 6 Systems, 4+2 w. shared PRE) - Advantages and Disadvantages

---

![Figure 22: Cutover in a Dual-Track Transport Landscape – Option-1](image)

**Figure 22:** Cutover in a Dual-Track Transport Landscape – Option-1
Option 1 allows the upgrade team to update PSD and PSQ early and outside of the PRD upgrade window. This option also allows the transport team to evaluate the transport package for completeness and the ability to re-package to improve import time to PSQ and PRD if necessary. The disadvantage of this option is that between upgrading PSD and PRD, there is not a development environment to support production in the need of an emergency correction.

A staggered cutover, option 2, as illustrated in Figure 23, is to cut over directly from the project landscape (from PRE in case of a standard 6-system landscape; from QAS in case of a 5-system landscape) into production PRD. As a second phase production support PSD and PSQ are updated.

![Figure 23: Cutover in a Dual-Track Transport Landscape - Option 2](image)

The advantage of option 2 is the ability to cut over directly into PRD, which can be done in a period of least activity, such as a weekend or holiday. Once production has been successfully updated, the supporting instances PSD and PSQ can be updated as a second phase, which may extend into normal operations.

In case of a 6-system dual-track landscape with 4+2 and shared PRE, the cutover is done through the PRE system as described above.

### 3.4.5 Retrofit in a Dual-Track Transport Landscape

One of the key challenges with the dual-track transport landscape is how to synchronize (retrofit) changes between the two tracks. This is necessary to ensure the two landscapes do not diverge. The need arises because changes made to production, to support business as usual, must be incorporated into the project track.

![Figure 24: Dual-Track Transport Landscape and Retrofit](image)

This section provides an overview of how to utilize SAP Solution Manager to support the retrofit process.
3.4.5.1 SAP Solution Manager ChaRM Retrofit

SAP Solution Manager offers support for the retrofit process within Change Request Management (ChaRM). By using the retrofit functionality in SAP Solution Manager, companies achieve significant improvements in synchronizing the two transport landscapes. The major advantages are:

- Conflicts are detected automatically.
- A high degree of changes performed in the production support landscape can typically be retrofitted automatically, without any manual effort.
- A complete worklist of all transports to be synchronized (down to object level) is created, and all changes are logged.

The retrofit process with SAP Solution Manager includes the following functionality:

- Developments in either transport landscape are recorded in SAP Solution Manager. As a result, changed objects are known centrally.
- Conflicts between the production support and project development landscape are identified via cross-system object lock (CSOL). A conflict exists when the same object is changed in both transport landscapes.
- Depending on the type of conflict and type of object, ChaRM offers specific synchronization methods to align changes from the production support landscape into the project landscape:
  - All objects without conflict (that is, changed only in PSD) are transferred automatically. For this, a transport of copies is created and sent to the project development system.
  - For objects with conflicts (changed in PSD and DEV), the synchronization is supported with semiautomatic or manual handling.
    - A conflicting workbench object can be adjusted semiautomatically in a split-screen editor. For this, the Correction Workbench is used. In rare cases, when the change was made within the same area of context, the adjustment has to be done manually.
    - In case of a conflicting customizing setting, the version in the production support system is recorded in a BC Set. During import into the project development system, it is compared with the entry in that system and can be adjusted accordingly.
  - Objects that have no tool support within the Correction Workbench have to be adjusted manually.
  - Retrofit is performed on object level; if one object of a transport request must be retrofitted manually, the other objects can still be transferred automatically.
- If an object was changed several times, the changes are automatically retrofitted in the correct sequence.
- A retrofit of a change triggered from PSD into the project development system DEV is considered in DEV like any other project-related development change. Therefore the adjustment in system DEV is
recorded in a new transport request. Having a separate CTS project as a container for all retrofit changes is usually the best option, but the retrofit changes could also be put into a CTS project together with other project changes.

- Putting the retrofit change into the CTS project that is scheduled to go live next—together with other developments—has the following disadvantage: If multiple projects with different duration and go-live dates are developed in parallel in DEV, strong governance and release management has to be in place to support this procedure. If the project without the retrofit changes goes live first, a downgrade of object versions might happen.

- Putting the retrofit change into a CTS project in DEV that is dedicated to collect all retrofitted objects (with cutover together with any next project that will go live) has the advantage that it is ensured that the retrofitted objects are always cut over—indeed, independent of single project delays.

- All retrofit activities are logged and can be reviewed at any time.

Retrofit requires a minimum setup of Change Request Management in SAP Solution Manager, but it is not necessary to use ChaRM workflows to leverage the retrofit functionality (see section 3.4.5.3.).

3.4.5.2 Considerations for When and How Often to Do the Retrofit

As soon as a transport request is released from the production support system (PSD), an entry for the retrofit worklist is created.

Some considerations as to when to best complete retrofit activities:

- On the one hand, the retrofit activity should be done as soon as possible after the transport request is released to make sure the worklist is manageable.

  - In a long lasting development cycle in DEV, the number of objects to retrofit can become quite large. The time needed for retrofit increases correspondingly and needs to be done on a continual basis.

- On the other hand, only tested changes should be retrofitted to avoid multiple transports to retrofit. So the timing of the retrofit activity also depends on the testing strategy.

  - Usually there is not sufficient test data in PSD and changes are imported into PSQ to be tested. After the change is tested successfully, the retrofit for the change can start. If an error is found in PSQ and a correction transport is needed, ChaRM will ensure that the correct sequence of transports is kept during retrofit.

  - The retrofit activity can start immediately after the release of the transport in PSD if the release of the transport means an “okay” from a testing perspective. (This can be the case if sufficient testing is done in PSD or if the change is sent to PSQ by means of transport of copies and tested in PSQ.)
• The retrofit must be complete before the cutover from the project landscape into the production support landscape, considering the final release testing. ChaRM will check if the retrofit work list has been processed before cutover can start.

• In case of managing the production support transports in releases, the best option would be to do the retrofit after/with the weekly/bi-weekly minor release import from production support into production.

3.4.5.3 Prerequisites for Using Retrofit in SAP Solution Manager

The following prerequisites need to be fulfilled so that the retrofit process can be implemented in SAP Solution Manager:

1. SAP Solution Manager 7.1.
   - Retrofit functionality already exists in SAP Solution Manager 7.0; however, the processing logic is different and the level of automation is lower than 7.1. For fresh ChaRM and retrofit implementations, it is highly recommended to start with SAP Solution Manager 7.1.

2. SAP Solution Manager ChaRM controls the transports in the production support system, and either SAP Solution Manager ChaRM or QGM is used in the project development landscape.
   - ChaRM can be implemented in different variants. The tighter the integration, the more re-use of information is possible.
   - The minimum for retrofit is to use ChaRM for creating/releasing the transport orders instead of doing this directly in the related development systems. Then a change document in Solution Manager ChaRM is created and released, which automatically creates and releases the transport orders in the corresponding development systems. This “central transport management” usually can be implemented within a few days and is a viable option even if a non-SAP Change Request Management tool or IT service management tool is used.
   - With SAP Solution Manager 7.1, Quality Gate Management (QGM) can create CSOL entries too. This means a mix of ChaRM in production support and QGM in project landscape is also an option for retrofit.

3. Cross-system object lock (CSOL) is activated:
   - CSOL helps to avoid inconsistencies based on wrong transport sequence, even in a single transport track.
   - With SAP Solution Manager 7.1, the CSOL information is used to identify if an object in the project landscape has been modified. This is used to calculate the status for ChaRM retrofit. That is, if there is no CSOL entry, an auto-import is possible by ChaRM retrofit. If there is a CSOL entry, the system performs some additional checks, for example, whether the retrofit can be processed using the split-screen editor or not.
3.5 Variant: 4-System Transport Landscape with Multiple Production Systems

Motivation: Business process harmonization across multiple entities such as sites, regions, or business units is a clear trend. An example of a transport landscape that supports a 100% share of all configuration and code is the 4-system transport landscape with multiple production systems.

This section refers only to the specifics of having multiple production systems in this landscape. For general considerations on the 4-system transport landscape, see section 3.3.

Specific considerations:

- This type of landscape is best if the goal is to have all (involved) production systems share the same configuration and code. This model is sometimes called a “virtual single instance”.
  - There is a central development system with a single client. All developments are done in the central development system DEV, including all localizations. A project development system could be added (comparable to that explained in section 3.4).
  - A single DEV system is the easiest way to control process harmonization. Nevertheless, a single DEV system does not guarantee process harmonization. A strong governance model is needed to protect the “template” (needs to be owned by the business); otherwise proliferation of process variants can happen in this option as well.
  - If the objective is to only share a “partial template” in all production systems. A cascading architecture of a template development system and localization development systems might be better, but of course also more complex with respect to transport execution and governance (including template protection).

- The pre-production systems are required to do proper release and regression testing. The “virtual single instance” is not recommended with a classical 3-system transport landscape. The default recommendation from SAP is to have a single pre-production system per production system because this is the only way to achieve proper regression testing for all changes, including cherry-picked transports or emergency corrections.

- The above picture shows a single QAS system for integration testing. Alternatively, you could also have a QAS system per production system. This depends on the test data refresh strategy and on whether the production systems differ much in the processes and interfaces used.
  - If the production systems only differ with respect to data, but the processes and interfaces are more or less the same across all production systems, a single QAS system is sufficient to perform a good integration test. The QAS system could be refreshed from either of the production systems.
If the production systems differ greatly in the processes used (for example, DEV contains all process variants, but they are only partially used in the production systems), then testing in a single QAS system with representative data from one production system may not be sufficient (for example, data missing to test the other process variants). The simplest way to solve this challenge and keep test data creation simple is to have one QAS system per production system.

- For speed of changes and downtime planning, this model allows for more flexibility in comparison to a single production system, but the flexibility is limited because the production systems should be kept in sync to achieve high quality in production support. Limited flexibility for changes means the following:
  - Single changes (in particular emergency transports) can be transported to a single PRD system first, but should be regression tested and imported to all production systems with the next minor release (for example, within the next 14 days).
  - Medium/major business releases should be imported to all parallel production systems within a few days, ideally within a weekend.
  - Upgrades and major changes for which a project landscape is in place can be done – if required – at different (consecutive) weekends using the project landscape temporarily for production support (of course, with the respective limited support and extra retrofit effort).

3.6 Suggested Size of the Development and Test Systems

The following figure provides an overview of the suggested size of the systems in the transport landscape – in t-shirt sizes.

<table>
<thead>
<tr>
<th>System Landscape</th>
<th>S</th>
<th>M</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-system landscape</td>
<td>DEV -&gt; QAS -&gt; PRD</td>
<td>DEV</td>
<td>QAS</td>
</tr>
<tr>
<td>4-system landscape</td>
<td>DEV -&gt; QAS -&gt; PRE -&gt; PRD</td>
<td>DEV, QAS</td>
<td>PRE</td>
</tr>
<tr>
<td>5-system landscape</td>
<td>PSD -&gt; PSQ -&gt; QAS -&gt; PRD</td>
<td>PSD, PSQ, DEV</td>
<td>QAS</td>
</tr>
<tr>
<td>6-system landscape</td>
<td>PSD -&gt; QAS -&gt; PRE -&gt; PRD</td>
<td>PSD, PSQ, DEV, QAS</td>
<td>PRE</td>
</tr>
</tbody>
</table>

Figure 26: Suggested Size of the Systems in the Transport Landscape

What is meant by the t-shirt sizes in the above figure is explained in the following table:
Some remarks for the configuration and size of the development and test systems:

- The development and most of the test systems can be small in size (CPU, RAM, and data volume). The testing purpose (for example, integration testing) can be fulfilled with a smaller amount of data and in a smaller system than production.

- The system for regression testing for projects (QAS respective PRE, depending on the landscape [see Figure 26]) should ideally have the full data volume compared to production, but can have less CPU/RAM deployed. This is required, but also sufficient to identify data-driven dependencies during functional tests (all types and combinations of data) and for performance and scalability tests. The size of tables very often has a critical influence on the performance of certain transactions. These tests can, however, also be done in a reduced CPU and RAM environment (extrapolation of the load).

- High availability is not required to ensure availability of the respective development or test system. But at least one system in the track should have the high availability configuration of the production system to test the respective technical settings. The system for regression testing for projects (QAS respective PRE, depending on the landscape [see Figure 26]) is generally the the best candidate for validating all technical settings of production. The system should be based on exactly the same technology as the production system (not necessarily of the same size), including high availability configurations.
4 SAP Tools and Services for Software Change Management

4.1 Motivation

As shown in the previous sections, software change management is mainly about implementing processes, governance, and control. The right tool support is essential to automate and control the process. The following section provides a brief overview of the main tools that SAP offers for change control management.

4.2 Overview of SAP Tools for Change Control Management

Management and control of software changes is one area in which SAP Solution Manager can help. These functionalities can be implemented with or without a non-SAP Change Request Management tool in place. The main building blocks/functionalities are shown below.

![Diagram of SAP Tools for Software Change Management]

Change and Transport System (CTS):

- The foundation of standardized change management processes is given by CTS and CTS+, which allow consolidation of the processing of changes independent of whether they are based on ABAP or non-ABAP or even non-SAP.
- For further information, see the documentation: [CTS+ at SAP SCN](https://scn.sap.com), [CTS+ at SAP HELP](https://help.sap.com).

Transport Management:
• On top of CTS and CTS+, SAP Solution Manager allows creation, approval and release of transports centrally across multiple SAP applications, as well as control of transport sequences either in a single track or in a dual-track environment. A “light” setup of SAP Solution Manager ChaRM is used for this.

• For further information, see the documentation: Change Request Management at SAP HELP, CSOL at SAP HELP.

Quality Gate Management (QGM):

• SAP Quality Gate Management provides an integrated and consistent quality process for all operational units across the various organizations of a company.

• SAP QGM ensures the quality of software changes by quality gates.

• It includes a central transport mechanism and change control to manage changes across the technology framework and application, including synchronized changes to business processes that run in ABAP and non-ABAP.

• It also includes flexible release management and built-in SAP transport best practices.

• For further information, see the documentation: QGM at SAP HELP.

Change Request Management (ChaRM):

• With ChaRM, you can implement an ITIL certified change management process. It provides full control and transparency over change execution and comes with predefined management processes and workflows.

• For further information, see the documentation: Change Request Management at SAP HELP.

Retrofit (dual-landscape synchronization):

• In case of a dual-track transport landscape (see section 3.4) the synchronization effort between the development systems can be minimized using the retrofit functionality in SAP Solution Manager.

• For more information, see section 3.4.5 on the best practices for the retrofit process.

• For further information see the documentation: Retrofit at SAP HELP.

Transport Analytics/Change Diagnostics:

• SAP Solution Manager includes end-to-end change analysis, which tracks changes (for example, technical configuration, code, content) across the landscape. This information is useful—for example, in cases of an unplanned service degradation—to find out if any ad-hoc changes were applied recently and may be the root cause.

• SAP Solution Manager also provides configuration validation, which helps to understand how homogeneous the productive systems are, including technical configuration. It offers, for example, comparison of OS level or DB level, system configuration (SAP or DB parameter), kernel release, security policy settings, and status of transports (for example, have certain transports arrived in the systems?).
For further information see the documentation: Change Diagnostics at SAP HELP, Configuration Validation at SAP HELP.

In addition to these tools for change control management, SAP Solution Manager offers support for various other areas in Software Change Management and Application Lifecycle Management, incl. e.g. support for test management or custom code management. The whitepaper “Two Value Releases per Year” offers more information on these tools (see: https://support.sap.com/solution-manager/knowledge-transfer.html → White Papers → Two Value Releases per Year.)

4.3 Overview of SAP Services for Software Change Management

As part of the SAP Enterprise Support contract, customers can benefit from several offerings that help improve the software change management process (see http://service.sap.com/esacademy for further information and updates). Below are some examples of the available offerings:

- Continuous Quality Checks:
  - CQC EarlyWatch Check – Provides early warnings in all configured areas, including missing CTS/CTS+
  - CQC Transport Execution Analysis – Provides customers transparency on change management practices and potential improvement areas

- Expert-Guided Implementations – Tools and Process Setup:
  - Change Control Management I: Enhanced Transport Management (CTS+) – Use of CTS+ for change and transport for ABAP and non-ABAP objects
  - Change Control Management II: Quality Gate Management – Use of SAP Solution Manager QGM for quality assurance process on top of the technical transport management system
  - Change Control Management III: Change Request Management – Use of SAP Solution Manager ChaRM for workflow-driven Change Request Management processes
  - Change Diagnostics – Use of SAP Solution Manager for comparison of systems to ensure the consistency of the landscape with respect to changes

- Expert-Guided Implementations – Guided self-services execution:
  - Transport Execution Analysis – Enables customers to analyze transports on their own with focus on QA check of existing processes
  - Transport Execution Analysis for Projects – Enables customers to analyze transports on their own with focus on proactive checks before the imports of a project are done.

There are also Rapid-Deployment Services available that help you to quickly setup the tools.

For SAP customers with a Premium Engagement contract, there are additional services available, for example, the SAP IT Planning services that offer planning or review of the software change management processes and the landscape on a strategic level.
# Abbreviations

<table>
<thead>
<tr>
<th>Abbreviations</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAU</td>
<td>Business as Usual</td>
</tr>
<tr>
<td>CAB</td>
<td>Change Advisory Board</td>
</tr>
<tr>
<td>ChaRM</td>
<td>Change Request Management, functionality in SAP Solution Manager</td>
</tr>
<tr>
<td>CQC</td>
<td>Continuous Quality Check, SAP Service delivery format</td>
</tr>
<tr>
<td>CSOL</td>
<td>Cross-System Object Lock, functionality in SAP Solution Manager</td>
</tr>
<tr>
<td>CTS</td>
<td>Change and Transport System</td>
</tr>
<tr>
<td>CTS+</td>
<td>Enhanced Change and Transport System to cover non-ABAP objects</td>
</tr>
<tr>
<td>DEV</td>
<td>used in this document for: Development System</td>
</tr>
<tr>
<td>PRD</td>
<td>used in this document for: Production System</td>
</tr>
<tr>
<td>PRE</td>
<td>used in this document for: Pre-production System</td>
</tr>
<tr>
<td>PSD</td>
<td>used in this document for: Development System in the Production Support Track</td>
</tr>
<tr>
<td>PSP</td>
<td>used in this document for: Pre-Production System in the Production Support Track</td>
</tr>
<tr>
<td>PSQ</td>
<td>used in this document for: Quality Assurance System in the Production Support Track</td>
</tr>
<tr>
<td>QAS</td>
<td>used in this document for: Quality Assurance System</td>
</tr>
<tr>
<td>QGM</td>
<td>Quality Gate Management, functionality in SAP Solution Manager</td>
</tr>
<tr>
<td>SBX</td>
<td>used in this document for: Sandbox System</td>
</tr>
<tr>
<td>SNOTE</td>
<td>Transaction in the SAP system to implement SAP Notes (also known as Note Assistant)</td>
</tr>
<tr>
<td>SWCM</td>
<td>Software Change Management</td>
</tr>
</tbody>
</table>