GROUP FND

SFDC GURU'S CHEAT SHEET

FOR

DEVELOPMENT LIFECYCLE AND DEPLOYMENT DESIGNER CERTIFICATION EXAM

Sandboxes

Sandbox Uses

Development

Type of Sandbox : Developer or Developer Pro

Notes: Full sandboxes are more costly in terms of create and refresh time, and would also give developers access to data that might not be appropriate

Testing

Type of Sandbox: Developer or Dev Pro for Unit tests and Apex Tests

Feature Tests & Regression Tests: Partial Copy with standard set of data loaded.

Testing with External Integrations

Type of Sandbox: Full sandbox is best when external system expects full production data to be present.

Note: Partial copy may be appropriate in special cases when you want to use sample data or a subset of actual data. Works well if you're using external ids.

Staging & UAT

Full sandbox is best for validation of new applications against production configuration and data.

Note: Partial Copy sandboxes are appropriate if testing against a subset of production data is acceptable, for example, for regional tests.

Production Debugging

Full sandbox

Metadata API Limits

You can deploy or retrieve up to 10,000 files at once and the maximum size of the deployed or retrieved zip file is 39 MB.

Establish a Change Process for Production

Different Models:

- Allow no changes on production – The simplest but you are sacrificing immediate setup changes for easier deployment. All development happens in a sandbox.
- Modify only components in the Metadata API – Manually tracking and deploying changes is far more complicated than using the Metadata API. If you can limit producion changes to components accessible through the Metadata API, then this will simplify change tracking, merging, and deployment.
- Allow One Admin to make setup changes – Easier to track changes and replicate prod changes back to sandboxes. More flexible approach that allows changes in production and project based development at the same time. Only a workable solution if your org is small enough that one admin can make all the setup changes.
- Schedule production changes – If your org requires frequent changes to production, schedule a time to migrate those changes to your development environments.

Manually Tracking Changes

Will likely have to track changes not available through the Metadata API through a custom application or spreadsheets or other tools. Each list should track at least:

- Who made the change
- Org where change occurred
- Date & Time
- Which component(s) changed

Development Lifecycle Roles

In a smaller company, one person can wear many hats, but in a larger company, specialized roles define what each person is responsible for. Logical roles include the following. Source

-
- **Release manager**—Manages the release schedule and coordinates releases with the business. The release manager could be in charge of pulling changes from version control.

-
- **Product manager**—Provides the business requirements of apps and features, and works with the development team to implement those requirements. The product

manager also performs user acceptance testing to ensure that requirements have been implemented.

- **Software developer**—Develops new functionality in sandbox, including both declarative point-and-click development and code.
- **Quality engineer**—Tests new functionality in sandbox.
- **Administrator**—Performs administrative tasks in the production org, and tracks all changes made in production.
- **Trainer**—Conducts training of company employees for new applications and features.

Governance to Manage Change

You've learned how to keep your organization lean and clean using the tools that Salesforce provides. However, governance, a method of management, is about more than tools. Governance improves agility by ensuring all members of your team are working together to achieve goals that are aligned with overall business goals. Source

Three elements of a responsive, adaptable framework for governance are:

Center of Excellence
A few stakeholders from different functional groups work together to ensure that changes support business goals and follow IT best practices and processes.

Release Management
You've already learned how to use tools like change sets and a sandbox to manage changes. If you use a backlog list to manage priorities, you can work on the most important changes first. And if you design and document a complete release management process as you learn more about your organization, everyone who works with Salesforce will be able to know how to do so safely.

Design Standards

Follow key standards for coding, testing, integration, large data volumes, and other areas that affect the services you share with other Salesforce customers.

Design Standards

Salesforce is configurable, extendable, and supports integration with other applications and services. By following the same design standards, anyone who modifies your organization can ensure that their changes aren't in conflict with another group's changes. In addition, with Salesforce's multi-tenant architecture, employing design standards helps ensure your changes stay within set governor limits.

Some examples of design standards include:

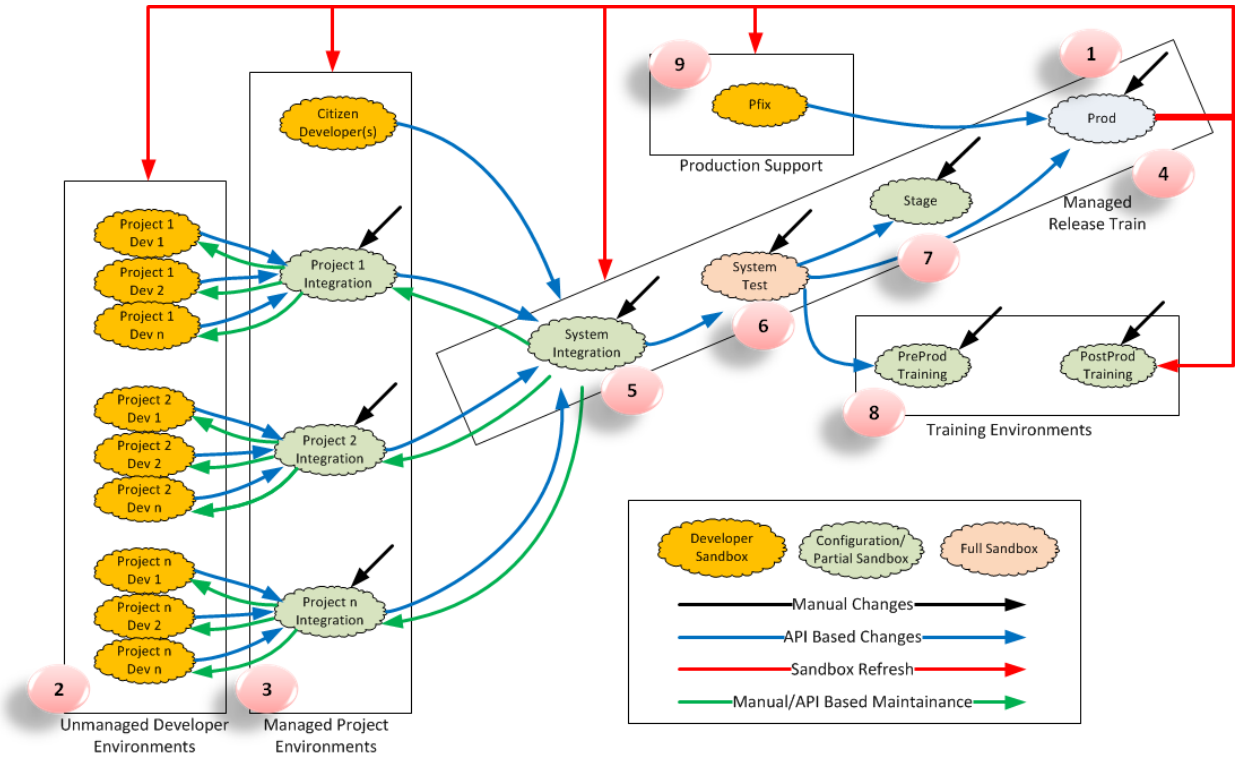
- Standard naming conventions
- Consistently using the Description field
- Bulkified code
- Standard methods for deprecating classes and fields
- Consistent data architecture across all projects

It's important to have design standards for the following areas:

- Coding
- Testing
- Integration
- Handling large data volumes
- Documentation

The architect or architecture team in your center of excellence defines your company's design standards. Publish your design standards, and communicate them to all teams that work on Salesforce projects, and the rest of IT.

Enterprise Environment Management



Source

9 Steps to Effective Change Management

Source

1. Get a Strategy
2. Engage an executive sponsor
3. Collect input from end-users
4. Define scope and impact
5. Prioritize
6. Configure and Test
7. Communicate and train
8. Deploy
9. Follow up & Support

CoE Roles & Responsibilities

Source

Figure 6: CoE Roles & Responsibilities

The day to day management of the CoE will be by the program team, which requires three
key roles discussed below. Depending on the size, scope, customer culture, and state of the
project, there will be a number of functional teams reporting into the program team. These
teams are:

- **Release** – this stream is responsible for creating and owning the overall project roadmap and release plan.
- **Business** – this team is responsible for converting the high-level goals and strategies into the business backlog, which is passed over to the development team for delivery.
- **Scrum Teams** – this stream is responsible for the delivery of the functionality as defined by the business stream.
- **Architectural** – this stream is responsible for defining the overall system architecture and making sure they adhere to the corporate standards, including data architecture.
- **Adoption/Training** – key for the long-term success of the project is to make sure that the end users understand how to use the system and have access to the appropriate level of training.
- **Support** – with all systems, when users have issues, they need to have an effective support structure and this stream owns this responsibility. Furthermore, if there is a system defect, they can add the issue to the business backlog or raise a system bug report.

CoE Key Roles

Source

One question commonly posed to Salesforce is this: what is the minimal key roles and associated skill sets required to operate an effective governance program? Typically, these roles include:

- **Business Analyst** – this role defines the entire project's business requirement roadmaps, looking for commonality across projects, and reviews the Salesforce roadmap to see if there is any overlap. This role also needs to understand the overall Business architecture and how to engage with the appropriate Business owners.
- **Enterprise Architect** – this role defines the company's best practices and standards around integration, security, SDLC, coding standards, etc., and makes sure that these best practices adhere to any relevant corporate standards. They are also the main link between Salesforce teams and the rest of IT, and need to have a strong background in system design (including Object-oriented Design) and data management and architecture. Remember that it is important to have consistent data architecture across all the Salesforce Orgs.
- **Project Management Office** – this role defines the overall delivery schedule, roadmap, and end user training approach. Also, this role owns the communications strategy, including keeping the project teams and user community in sync. The PMO also acts as the catalyst for adoption of new delivery models (i.e., Scrum) and to enforce the business and IT feedback loop.

Tools/Matrices

Source

RTM (Requirements Traceability Matrix)

Used for tracking project requirements, great for outlining functional test that need to be accounted for during planning and project management. When refactoring code this can be helpful for creating regression test. Be aware of which scenarios this should be used

R.A.C.I. (Responsible, Accountable, Consulted, Informed)

Used to define each team member's role and assists with reducing confusion on expectations, in turn, increasing project efficiency.

- Responsible – Who is completing the task.
- Accountable – Who is making decisions and taking actions on the task(s).
- Consulted – Who will be communicated with regarding decisions and tasks.
- Informed – Who will be updated on decisions and actions during the project.

Steering Committee

Source