

Sharing and Visibility Designer Study Guide

Force.com Security	6
Users and security	6
User Password	6
User Authentication	6
Network-based Security	6
Session Security	6
Auditing.....	6
Data Auditing.....	6
Programmatic security	7
Security Tokens.....	7
OAuth.....	7
Platform security framework	7
System Permissions	7
Administrative Permissions	7
Reports	7
Data	7
Component Permissions	8
Record-based Sharing.....	8
Organization-wide Defaults	8
Sharing	8
Sharing Architecture.....	9
Licenses.....	9
Full Sharing Model Usage Users/Licenses.....	9
High Volume Customer Portal License.....	9
Chatter Free License.....	9
Community License	9
Community Licenses and limits	10
Components	10
Control Data Access.....	11
Profiles and Permission Set.....	11
Record Ownership and Queues	11
Organization wide Defaults	11
Role Hierarchy	12
Public Groups.....	12
Ownership-based Sharing Rules.....	13
Criteria-based Sharing Rules	13
Manual Sharing.....	13
Teams.....	13
Territory Hierarchy	13
Account Territory Sharing Rules.....	14
Programmatic Sharing.....	14
Implicit Sharing.....	14
Sharing between accounts and child records	15
Sharing behavior for portal users	15
Considerations when territory management is need.....	15
What happens to the Role Hierarchy?.....	15
Can You Still Use Teams?	15
Realignment and Reassignment.....	16

Large Data Volumes.....	16
Defer Sharing Calculations	16
Data Skews/Ownership Skews	16
Account Data Skew	16
How to Avoid Account Data Skew	16
The Account Hierarchies Impact on Data Access	16
Troubleshooting	17
Territory Management	18
Setup	18
Territory Model	18
Territory Model State	19
Assignment Rule	19
Filter-based opportunity territory assignment	19
Get the Most from Territory Management	20
Report on Territories	20
Differences - Territory Management (1.0) and Enterprise Territory Management (2.0)	21
Permissions Affect Enterprise Territory Management?	21
Considerations:	21
Account & Opportunity Teams.....	22
Account Teams	22
Set Up and Manage Account Teams	22
Enable Account Teams	22
Customize Account Team Roles	22
Considerations.....	22
Add Account Team Members.....	22
Considerations for Removing Account Team Members.....	22
Account Team Fields.....	23
Opportunity Teams.....	23
Understanding Sharing	24
Managed Sharing.....	24
User Managed Sharing	24
Apex Managed Sharing.....	24
The Sharing Reason Field	24
Access Levels	25
Sharing Considerations.....	25
Security and Sharing in Customer & Partner Community	26
Share Group.....	26
Sharing Sets	26
Objects Supported	26
User licenses	26
Usage.....	27
Sharing Data with Partner Users	27
Groups/Categories	27
Usage	27
Apex Managed Sharing.....	28
Sharing a Record Using Apex.....	28
Share Object Properties	28
Creating User Managed Sharing Using Apex	29
Creating Apex Managed Sharing.....	29
Apex Sharing Reason Creation	29
Considerations.....	30

Creating Apex Managed Sharing for Customer Community Plus users	30
Ways to share:	30
Apex Sharing Recalculation	30
Associate an Apex managed sharing recalculation class	30
Considerations for recalculations	30
With Sharing	31
Without Sharing	31
Inherited Sharing	31
Enforcing Sharing Rules	31
Who Can See My File?	33
Actions for your file permissions	33
Considerations	33
Create a Custom List View in Salesforce Classic	34
USER PERMISSIONS NEEDED	34
Considerations	34
Share a Report or Dashboard Folder in Salesforce Classic	34
USER PERMISSIONS NEEDED	34
Access and Limits	34
Designing Record Level Access for Enterprise Scale	36
Surviving Owner Change Operations	36
Using Apex Sharing Reasons	36
Using Outbound Messaging	36
Using a Trigger	36
Using a Shadow Table	36
Completing the Architecture	36
Group Maintenance Tables	37
Groups and Composition	37
Example	38
Territory Management Groups	38
Considerations	38
Obtain peak performance:	38
Access Grants	39
Common Group and Data Updates	39
Group Membership Locking	40
Takeaway: Tuning Group Membership for Performance	40
Takeaway: Tuning Data Relationships and Updates for Performance	40
Force.com Record Locking Cheatsheet	40
Tools for Large-Scale Realignments	41
Parallel Sharing Rule Recalculation	41
Deferred Sharing Maintenance	41
How works in practice	41
How it helps	41
Considerations:	42
Granular Locking	42
Key Advantages	42
Considerations:	42
Classic Encryption for Custom Fields	43
Restrictions	43
Best Practices	43

Salesforce Shield	44
Platform Encryption	44
Encrypt Fields	44
Difference Between Classic Encryption and Shield Platform Encryption.....	45
Shield Platform Encryption Best Practices	45
Event Monitoring	46
Field Audit Trail	46
Usage	46
Objects Supported.....	46
Field can't be tracked	47
Considerations.....	47
Data Leak Prevention	48
Authorization	48
How the Salesforce Platform Enforces Authorization?	48
User Context	48
System Context.....	48
Purpose of Multiple Contexts	48
CRUD and FLS Enforcement in VisualForce and Lightning	49
Protect Against CRUD and FLS Violations	49
Is My Application Vulnerable?	49
How Can I Test My Application?	49
runAs Method	50
Nesting	50
Other Uses of runAs	50
Injection Vulnerability Prevention	51
Cross-Site Scripting(XSS)	51
Types of XSS Attacks	51
Impact of XSS	51
Common XSS Mitigations	51
Built-in XSS Protections in Lightning Platform	51
Prevent XSS in Lightning Platform Applications	52
Platform Encoding in Apex	52
SOQL Injection	52
Impact of SOQL Injection.....	52
SOQL Injection Prevention	52
Storing Sensitive Data	53
Sensitive Data - What is it?	53
Measures	53
Hardcoded Secrets	53
Debug Logs	53
Sensitive Info in URL	53
Salesforce.com Integrations.....	53
Sample Vulnerability	53
Securing Data in Application	54
Is My Application Vulnerable?	54
How Can I Test My Application?	54
How Do I Protect My Application?	54
Problem 1	54
Problem 2	54

Problem 3.....	54
Apex and Visualforce Applications.....	55
General Guidance	56
ASP.NET.....	56
Java	56
PHP.....	56
Ruby on Rails.....	56
Python.....	57
Flash/Air apps	57

Force.com Security

Users and security

User Password

- Password Policies lets you determine password expiration, minimum password complexity requirements and lockout periods.
- Reset password of selected users.

User Authentication

- Delegated Authentication
 - a user logs into the platform as usual, but the platform uses a web service callout to submit the user name and password to an external authorization authority.
 - Once that authority approves the logon, the approval is passed back to the platform and the user can proceed.
- Security Assertion Markup Language (SAML)
 - Using SAML, your request goes to the SAML "identity provider", a login page hosted by your organization that validates your identity and returns a token.
 - The token is passed to the platform, which verifies the user by validating that it is signed by the appropriate identity provider.
 - This approach is typically used when your users are accessing your platform applications through a portal, which would handle the initial authentication and avoid the need to log into Force.com again.

Network-based Security

- The first option is to allow from users from trusted locations, but challenge them when they come from new and untrusted locations. Setup | Security Controls | Network Access allows you to whitelist a set of IP address ranges that you trust.
- If a profile has Login IP restrictions defined, any user with that profile can only login to the platform from those IP addresses.
- The limitations imposed on IP addresses are used to help protect against phishing attacks.

Session Security

- The Setup | Security Controls | Session Settings page lets you control this session.
 - session timeout
 - all pages always be accessed using a secure connection

Auditing

- Setup | Manage Users | Login History displays the last 20 logins to your organization, as well as access to download 6 months worth of login data, which includes IP addresses, browser types and so on.
- The Setup | Security Controls | View Setup Audit Trail page lets you audit metadata and system changes.

Data Auditing

- Object-level auditing tracks changes in the overall object records, such as record creation.
- You can also enable auditing for individual fields, automatically tracking any changes in the values of selected fields.

Programmatic security

Security Tokens

- The email challenge mechanism is a way to allow a user to login from outside of an IP range.
- If a client is run from a host outside the whitelisted IP ranges, the client has to append a security token to the password of the user that is being authorized.

OAuth

- OAuth is an open protocol that allows a website to access resources of another website without having to expose a user's credentials.
- Instead of supplying a username and password, OAuth allows users to hand out security tokens to specific sites for access to specific resources for a defined duration.

Platform security framework

System Permissions

- System permissions are granted to profiles

Administrative Permissions

- Manage Users - allows user to modify all user attributes.
- API-enabled - Without this permission, a user cannot access the Force.com system from outside of the environment.
- API-Only User - prevents users with this permission in their profile from logging into the Force.com platform, except through one of the Web services APIs.
- View Setup and Configuration - allows users to view complete Setup menu, without the ability to make changes.
- Password never expires - as it says.
- Customize application - allows complete editing access to options for Force.com applications
- Edit HTML Templates, Manage Letterheads, Manage Public Templates - all related to components used for Force.com messages.
- Author Apex - allows users with this permission in their profile to create and edit Apex. Requires the Modify All Data permission as a prerequisite.

Reports

- Create and Customize Reports - grants access to create new reports or modify existing reports.
- Run Reports - allows users to access the reports tab.
- Export Reports - allows users to export data from reports to an Excel spreadsheet format.
- Manage Custom Report Types, Manage Dashboards, Manage Public Reports, Schedule Dashboards - allows users to manage and modify the respective component types.

Data

- The following permissions regard data manipulation, but from an administrative perspective. See Record-based Sharing for a developer perspective:
- Modify All Data - a very powerful permission that, if granted globally, allows users to modify all data in the Force.com organization.
- View All Data - allows user to see all data in the Force.com organization, if granted globally.
- Edit Read-Only Fields - allows users with this permission in their profile to edit read-only limitations set in a page layout.
- View Encrypted Data - allows users with this permission in their profile to see plain text representation of encrypted data.
- Weekly Data Export - allows users with this permission in their profile to perform a weekly data export.
- Disable Outbound Messaging - prevents the use of outbound messaging for the profile.

Component Permissions

Force.com platform also allows you to set permissions on individual Force.com components:

- Applications
 - Tabs
 - Record types
 - Apex classes
 - Visualforce pages
- However, permission sets and profiles don't include access for some custom processes and apps
- Custom permissions let you define access checks that can be assigned to users via permission sets or profiles, similar to how you assign user permissions and other access settings.

You can query custom permissions in these ways.

- To determine which users have access to a specific custom permission, use Salesforce Object Query Language (SOQL) with the SetupEntityAccess and CustomPermission sObjects.
- To determine what custom permissions users have when they authenticate in a connected app, reference the user's Identity URL, which Salesforce provides along with the access token for the connected app.

Record-based Sharing

Organization-wide Defaults

- Specify the absolute minimum level of access to the records in an object.

Sharing

- Manually
 - This button is on all detail pages by default, although the button can be removed from a page layout.
 - The Share button will not appear for records whose organization-wide default is set to Public Read-Write, as there is no need to grant further sharing privileges for records in this object.
- Sharing rules
 - You can share records to a role or a group, or with a territory, which is designed to support CRM implementations.
- Apex
 - either by automatically assigning sharing when a record is created, or by using the Apex managed sharing (which only applies to custom objects.)

Sharing Architecture

Licenses

Full Sharing Model Usage Users/Licenses

- Most Standard Salesforce license types take full advantage of the sharing model components.
- The license might not make a module accessible, or even some objects accessible. For example, the Force.com Free edition can't access any CRM objects.
- However, the sharing entities, and functionality, still exists and is ready when and if the module ever does become active.

High Volume Customer Portal License

- High Volume Customer Portal (HVPU) license users (including Community and Service Cloud license users) do not utilize the sharing model.
- HVPU licenses have their own sharing model that works by foreign key match between the portal user (holding the license) and the data on Account and Contact lookups.
- HVPU license is only used for the Customer Portal and not the Partner Portal.

Chatter Free License

- The Chatter Free license doesn't follow the standard sharing model.
- Chatter Free is a collaboration-only license with the following features: Chatter, Profile, People, Groups, Files, Chatter Desktop, and limited Salesforce1 app access.
- The license doesn't have access to CRM records (standard or custom objects) and Content functionality, and therefore, there is no sharing.

Community License

1. Customer Community
 - a. Basic License
 - b. Don't have any roles, so can't use sharing rules but can use sharing ser and groups.
 2. Customer Community Plus
 - a. Customer Community +
 - i. Reports and dashboards
 - ii. Delegated admin
 - iii. Content libraries
 - iv. Records across accounts
 3. Partner Community
 - a. Customer Community +
 - i. Leads and opportunities
 - ii. Campaigns
- Customer and Customer Community Plus licenses require unique usernames within the Salesforce org that a community belongs to.
 - Partner Community licenses and Employee Community licenses require unique usernames across all Salesforce orgs that the user belongs to.
 - Communities licenses are associated with users, not a specific community.
 - Unlike other community users, high-volume community users don't have roles, which eliminates performance issues associated with role hierarchy calculations

	Customer Community	Customer Community Plus	Partner Community	Employee Community

Sharing Set	✓			
Share Group	✓			
Account Team Sharing			✓	✓
Case Team Sharing			✓	✓
Opportunity Team Sharing			✓	✓
Manual Sharing		✓	✓	✓
Role Hierarchy		✓	✓	✓
Sharing Rules		✓	✓	✓
Apex Sharing		✓	✓	✓

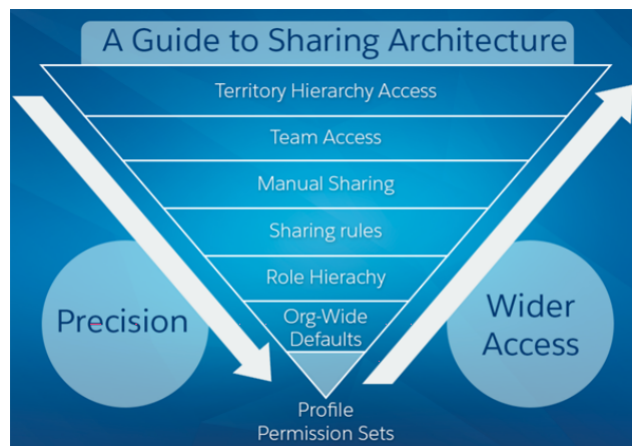
Community Licenses and limits

- In Enterprise, Performance, and Unlimited orgs, you can create up to 100 communities without buying communities licenses.
- The default number of roles per org is 5,000
- Unauthenticated or guest users who access your community do not use up any of your community's licenses.
- However, to create communities using the Partner Central template, you need to purchase at least one Partner Community license.
- Even without communities' licenses,
 - external users have some access to your communities
 - use your community as a public knowledge base for unauthenticated (or guest) users
- Purchase Community Cloud licenses to allow members to log in or give access to Salesforce objects based on your business needs.

Community License Type	Number of Users
Partner or Customer Community Plus	1 million
Customer	10 million

Salesforce Edition	Number of Page Views
Enterprise Edition	500,000/month
Unlimited Edition	One million/month

Components



Control Data Access

1. Create profiles and permission sets – Identify the different types of users you need for your application, based on the different functions each type needs to access.
 - a. Create a base level profile for each type of user so that each profile has only the permissions required for that type of user to perform these functions.
 - b. Then create permission sets to handle exceptions—situations in which a user may need a few more permissions.
2. Assign users – Assign each user to the appropriate profile and permission sets.
3. Set sharing models – For each object, set the organization-wide default record sharing settings to determine whether the records that each user owns are public or private.
4. Share private records – Use roles, groups, record sharing rules, and other means to share private records with other users.

Profiles and Permission Set

- For each object, the “View All” and “Modify All” permissions ignore sharing rules and settings, allowing administrators to quickly grant access to records associated with a given object across the organization.
- These permissions are often preferable alternatives to the “View All Data” and “Modify All Data” administrative permissions

Record Ownership and Queues

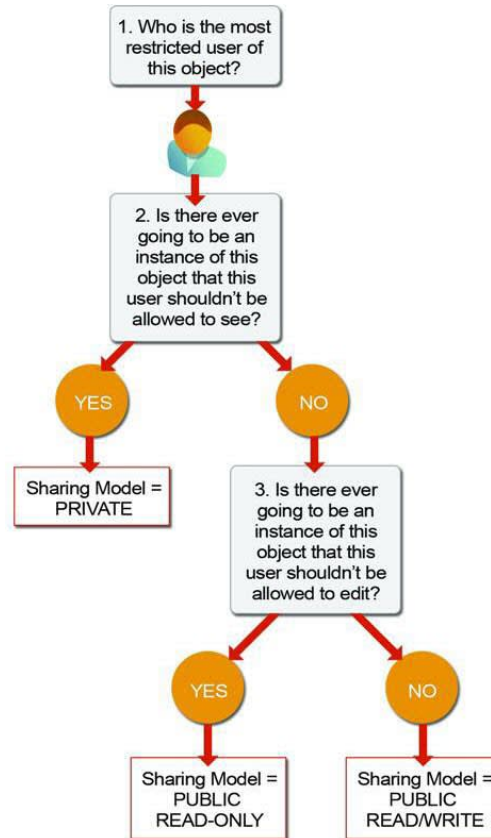
- Every record must be owned by a single user or a queue
- Users higher in a hierarchy (role or territory) inherit the same data access as their subordinates for standard objects
- Queues help you prioritize, distribute, and assign records to teams who share workloads.
- Queue members and users higher in a role hierarchy can access queues from list views and take ownership of records in a queue.

If a single user owns more than 10,000 records, as a best practice:

- The user record of the owner should not hold a role in the role hierarchy.
- If the owner's user record must hold a role, the role should be at the top of the hierarchy in its own branch of the role hierarchy.

Organization wide Defaults

- You use organization-wide sharing settings to lock down your data to the most restrictive level, and then use the other record-level security and sharing tools to selectively give access to other users.
- Organization-wide defaults are the **only way** to restrict user access to a record.
- For custom objects only, use the Grant Access Using Hierarchies setting, which if unchecked (default is checked), prevents managers from inheriting access.
- Even if Grant Access Using Hierarchies is deselected, some users—such as those with the “View All” and “Modify All” object permissions and the “View All Data” and “Modify All Data” system permissions—can still access records they don't own.



Role Hierarchy

- An organization is allowed 500 roles; however, this number can be increased by Salesforce.
- As a best practice, keep the
 - number of non-portal roles to 25,000 and the number of portal roles to 100,000.
 - role hierarchy to no more than 10 levels of branches in the hierarchy.
- Overlays are always the tricky part of the hierarchy. If they're in their own branch, they'll require either sharing rules, teams, or territory management to gain needed access

Public Groups

Public groups can consist of:

- Users
- Customer Portal Users
- Partner Users
- Roles
- Roles and Internal Subordinates
- Roles, Internal and Portal Subordinates
- Portal Roles
- Portal Roles and Subordinates
- Territories
- Territories and Subordinates
- Other public groups (nesting)

- As a best practice, keep the total number of public groups for an organization to 100,000.

- Groups also have the ability to protect data shared in the group from being made accessible to people in the role hierarchy above the group members.
- This (and dealing with the access of record owners and their management hierarchy) allows the creation of groups in which very highly confidential information can be shared—the data will be accessible **ONLY** to group members, and nobody else in the organization. This is accomplished by using the Grant Access Using Hierarchies setting.

Ownership-based Sharing Rules

- Ownership-based sharing rules are based on the record owner only.
- Contact ownership-based sharing rules don't apply to private contacts.
- As a best practice, keep the number of ownership-based sharing rules per object to 1,000

Ownership-based Sharing is used to provide data access to

- peers who hold the same role/territory
- other groupings of users (public groups, portal. roles, territories).

Criteria-based Sharing Rules

- As a best practice, keep the number of criteria-sharing rules per object to 50; however, can be increased by Salesforce.
- To provide data access to users or groups based on the value of a field on the record.

Manual Sharing

- Manual sharing is removed when the record owner changes or when the sharing access granted doesn't grant additional access beyond the object's organization-wide sharing default access level.
- Only manual share records can be created on standard objects
- Manual share records are defined as share records with the row cause set to manual share
- All share records (standard and custom objects) with a row cause set to manual share can be edited and deleted by the Share button on the object's page layout, even if the share record was created programmatically.
- You have access to the Sharing button when your sharing model is either **Private** or **Public Read Only** for a type of record or related record.

Teams

- Only owners, people higher in the hierarchy, and administrators can add team members and provide more access to the member.
- A team member with read/write access can add another member who already has access to the record with which the team is associated. The team member can't provide them additional access.
- The team object is not a first-class object. You can't create custom fields, validations rules, or triggers for teams.

Territory Hierarchy

- Territory management is not reversible, so it's extremely important to know its implications
- When territory management is enabled you must manage both the role hierarchy and territory hierarchy
- Territories exist only on Account, Opportunity and master/detail children of Accounts and Opportunities.
- Organizations can have up to 500 territories;
- If the assignment rules for a territory are changed, any Account Territory sharing rules using that territory as the source will be recalculated. Likewise, if the membership of a territory changes, any ownership-based sharing rules that use the territory as the source will be recalculated

Territory Management will be used or considered:

- Multiple groups of people (multiple teams) require either read-only or read/write access to accounts.
- An additional hierarchical structure (different from the role hierarchy) is needed.
- A single user needs to hold multiple levels in the hierarchy.
- Global users (GAM – global account manager) need to see everything from the global account downward

Account Territory Sharing Rules

- Account territory sharing rules become available only when Territory Management has been enabled for an organization.
- To provide data access to accounts within a territory (not based on ownership) to a grouping of users.

Programmatic Sharing

If you create a share record programmatically, and the out-of-box row cause (manual share) is used, then you can maintain this share record with the Share button in the app

Programmatic Sharing will be used or considered:

- No other method of sharing (declarative) meets the data access needs.
- There is an existing, external system of truth for user access assignments which will continue to drive access and be integrated with Salesforce.
- Poor performance by using native sharing components. (Usually applies to very large data volumes)
- Team functionality on custom objects

Implicit Sharing

- You can neither turn it off, nor turn it on—it is native to the application.
- Parent implicit sharing is providing access to parent records (account only) when a user has access to children opportunities, cases, or contacts for that account
- Child implicit sharing is providing access to an account's child records to the account owner.
 - only applies to contact, opportunity, and case objects (children of the account).
- Implicit sharing doesn't apply to custom objects.
- The access levels that can be provided are View, Edit, and No access for each of the children objects when the role is created.

Type of Sharing	Provides	Details
Parent	Read-only access to the parent account for a user with access to a child record	<ul style="list-style-type: none"> • Not used when sharing on the child is controlled by its parent • Expensive to maintain with many account children • When a user loses access to a child, Salesforce needs to check all other children to see if it can delete the implicit parent.
Child	Access to child records for the owner of the parent account	<ul style="list-style-type: none"> • Not used when sharing on the child is controlled by its parent • Controlled by child access settings for the account owner's role • Supports account sharing rules that grant child record access • Supports account team access based on team settings • When a user loses access to the parent, Salesforce needs to remove all the implicit children for that user.
Portal	Access to portal account and all associated contacts for all portal users under that account	Shared to the lowest role under the portal account

Type of Sharing	Provides	Details
High Volume1	Access to data owned by high volume users associated with a sharing set for users member of the sharing set's access group	All members of the sharing set access group gain access to every record owned by every high volume user associated with that sharing set
High Volume Parent	Read only access to the parent account of records shared through a sharing set's access group for users member of the group	Maintains the ability to see the parent account when users are given access to account children owned by high volume users

- To allow portal users to scale into the millions, Community users have a streamlined sharing model that does not rely on roles or groups, and functions similarly to calendar events and activities.
- Community users are provisioned with the Service Cloud Portal or Authenticated Website licenses.

Sharing between accounts and child records

- Access to a parent account
 - If you have access to an account's child record, you have implicit Read Only access to that account.
- Access to child records
 - If you have access to a parent account, you have access to the associated child records. The account owner's role determines the level of access to child records.

Sharing behavior for portal users

- Account and case access
 - An account's portal user has Read Only access to the parent account and to all of the account's contacts.
- Management access to data owned by Service Cloud portal users
 - Since Service Cloud portal users don't have roles, portal account owners can't access their data via the role hierarchy. To grant them access to this data, you can add account owners to the portal's share group where the Service Cloud portal users are working. This step provides access to all data owned by Service Cloud portal users in that portal.
- Case access
 - If a portal or customer community plus user is a contact on a case, then the user has Read and Write access on the case.

Considerations when territory management is need

What happens to the Role Hierarchy?

- You are now managing two hierarchies, which means sharing is more complex.
- The best practice is to flatten (or simplify) the role hierarchy as much as possible
- The rule of thumb is to make your role hierarchy your non-sales hierarchy, try to flatten the sale department branch(es), and then use the territory hierarchy as your "sales" hierarchy.

Can You Still Use Teams?

- Yes. However, only implement teams if no other existing sharing component will satisfy the requirement.

Realignment and Reassignment

- As a rule of thumb, have hierarchy structural (realignment) changes occur no more than quarterly and all changes of high volume (bulk or mass changes) be well planned, tested, and coordinated.

Large Data Volumes

- If you have more than two million accounts, and have implemented teams or Territory Management, you especially need to pay attention to performance.
- These are complex sharing model components that can make for a huge volume of share records and hence, long running transactions.

Defer Sharing Calculations

- Natively, every individual change to the role hierarchy, territory hierarchy, groups, sharing rules, user roles, team membership, or ownership of records can initiate automatic sharing calculations.
- Defer sharing helps here.

Data Skews/Ownership Skews

- Data skews are defined as a few parent records with many children records.
 - The ratio where we start seeing performance degradation is 1:10,000.
 - As a best practice, keep the ratio as close to that as possible (lower is preferred).
- Ownership skews where a single user, role, or group owning a large number of records for an object
 - The recommended ratio of owner to number of records is also 1:10,000.

If a single user owns more than 10,000 records, as a best practice:

- The user record of the owner should not hold a role in the role hierarchy.
- If the owner's user record must hold a role, the role should be at the top of the hierarchy in its own branch of the role hierarchy.
- If the user(s) must have a role to share data, we recommend that you:
 - Place them in a separate role at the top of the hierarchy
 - Not move them out of that top-level role
 - Keep them out of public groups that could be used as the source for sharing rules

Account Data Skew

- Account data skew occurs when an Account's parent object has more than 10,000 child objects

Two situations in particular pose a risk of producing locking errors.

- Updates to parent records and their children are being processed simultaneously in separate threads.
- Updates to child records that have the same parent records are being processed simultaneously in separate threads

How to Avoid Account Data Skew

- Design architecture to limit account objects to 10,000 children. Some possible methods include creating a pool of Accounts and assigning children in a round robin fashion or using Custom Settings for the current Account and the number of children.
- If possible, consider a Public Read/Write sharing model in which the parent account stays locked, but sharing calculations don't occur.
- If you have a skewed account, redistribute child objects in chunks during off-peak hours to lessen the impact of record-level lock contention. Batch Apex or the Bulk API are useful ways to re-parent.

The Account Hierarchies Impact on Data Access

- a parent/child relationship between two records does not drive access.

Troubleshooting

Why a user can or can't see a record. Here is a troubleshooting flow:

1. Verify that the user has permissions to access to the object.
2. Identify the user's role who can't see the record and note it.
3. Identify the owner's role of the record and note it.
4. Review the role hierarchy and verify these two roles are in two different branches (they should be).
5. Now you need to review the sharing rules for the object and make sure there is no rule that will grant the user access.
6. If you are using teams, should this user be on the team for that record? How are teams maintained and how did the miss occur?
7. If manual sharing is used, the user may have lost access because the record owner changed. Manual shares are dropped when ownership changes. The manual share could also have been removed using the Share button.
8. If you are using territory management, is the user missing from one of the territories? Where is the membership of territories maintained and how did the miss occur? Or, maybe the record did not get stamped with the territory where the user is a member.
9. If you are creating programmatic shares and there are criteria for creating the share in code, review the code to understand why this user was omitted.

Territory Management

Enterprise Territory Management:

- Salesforce admins can set up and test territory models before implementing them.
- It's easy to make assignments between territories, accounts, and opportunities.
- Reports help teams organize for optimal coverage and assess territory effectiveness.
- If you use Collaborative Forecasts, you can forecast by territory.

What It's Called	What It Does
Territory	<ul style="list-style-type: none"> • Helps you organize groups of accounts and the sales reps who work with those accounts. • You create territories based on territory types.
Territory type	<ul style="list-style-type: none"> • Every territory you create has a territory type. You use territory types only to organize and create territories. They don't appear on territory model hierarchies.
Territory type priority	<ul style="list-style-type: none"> • Helps you choose the appropriate territory type for territories you create or edit. You create your own priority scheme.
Territory model	<ul style="list-style-type: none"> • Modeling lets you create and preview multiple territory structures and different account and user assignments before you activate the model that works best.
Territory hierarchy	<ul style="list-style-type: none"> • You start from the hierarchy to create, edit, and delete territories; run assignment rules for territories, and navigate to territory detail pages for more information. • From the hierarchy, you can also assign territories to opportunities, run assignment rules at the model level, and activate or archive the model. Your territory hierarchy in the active territory model also determines the forecasts hierarchy for territory forecasts.
Territory model state	<ul style="list-style-type: none"> • Indicates whether a territory is in the planning stage, in active use, or archived.

Setup

- Setup -> Manage Territories -> Settings -> Enable Territory Management
- Create territory type and model.
- From the territory model -> View Hierarchy -> create a territory.

Territory Model

- This limit includes models created by cloning.

Edition	Number of Models in Production Orgs	Number of Models in Sandbox Orgs
Developer	4	4
Enterprise	2	2
Performance	4	4
Unlimited	4	4

Territory Model State

- Territory model state indicates whether a territory is in the planning stage, in active use, or archived.
- You can have only one active territory model at a time, but you can create and maintain multiple models in planning or archived state to use for extra modeling or reference.

LIFECYCLE STATE	DEFINITION
Planning	The default state for every new territory model you create. The Planning state lets you preview a model's territory hierarchy before deploying it.
Active	The state of a territory model after you activate it and all processing is complete. Only one model in your Salesforce organization can be active at a time.
Archived	The state of a territory model after you archive it and all processing is complete. An archived model lets admins view hierarchy and rule assignments as they were configured when the model was active. Only the active model can be archived, and archived models cannot be reactivated.

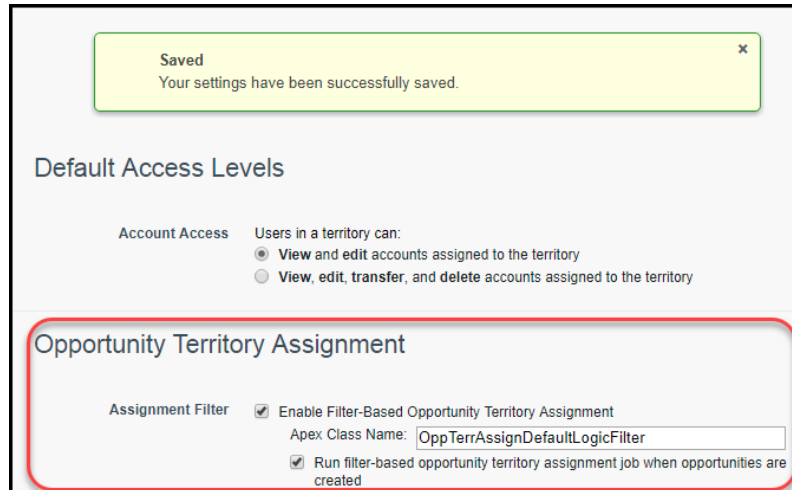
ERROR STATE	DEFINITION
Activation Failed	An error occurred during activation. Check your email for more information from Salesforce.
Archiving Failed	An error occurred during activation. Check your email for more information from Salesforce.

Assignment Rule

- A rule tells Enterprise Territory Management to assign accounts with those characteristics to that territory.
- If your territory is in Planning state, running rules lets you preview account assignments.
- If your territory is in Active state when you run rules, accounts are assigned to territories according to your rules.

Filter-based opportunity territory assignment

- Manually assigning a opportunity to territory using "territory field" in opportunity.
- Apex Class required for Filter-Based Opportunity Territory Assignment
 - Implements OpportunityTerritory2AssignmentFilter interface.
- Enable Filter-Based Opportunity Territory Assignment and set the class.
- You can assign territory type priority via the API by updating the Territory2Type object's Priority field.
- Manually Exclude an Opportunity from Filter-Based Territory Assignment using "Exclude from the territory assignment filter logic" in opportunity.
- Run Opportunity Filter from Territory model's hierarchy in Setup.



Saved
Your settings have been successfully saved.

Default Access Levels

Account Access Users in a territory can:

- ☒ View and edit accounts assigned to the territory
- ☐ View, edit, transfer, and delete accounts assigned to the territory

Opportunity Territory Assignment

Assignment Filter ☒ Enable Filter-Based Opportunity Territory Assignment

Apex Class Name:

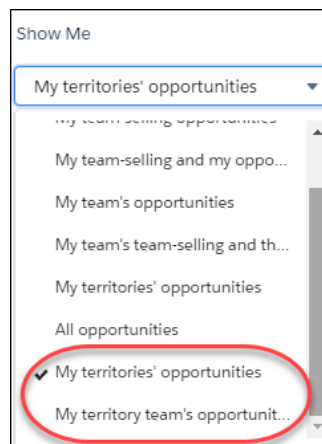
☒ Run filter-based opportunity territory assignment job when opportunities are created

Get the Most from Territory Management

- Clone a territory from territory model page.
- Territory Users by Territory Role
 - Setup -> Territory Associations -> Role in Territory
 - Assign roles . Territory Record -> Assigned User Related list -> Edit -> Role in Territory.
- Chatter to Collaborate on Territory Models using Setup -> Chatter -> Feed Tracking -> Territory Model -> Enable feed tracking

Report on Territories

- First you need a custom Territory Management report type that relates the objects you want to report on.
- Then you create reports that belong to that type.
- Steps:
 - On the Reports tab, click New Report
 - Choose an account or opportunity report type, and then click Continue.
 - In the Show Me filter field, select My territories' or My territory team's as the filter criterion.



Show Me

My territories' opportunities

My team-selling and my oppo...

My team's opportunities

My team's team-selling and th...

My territories' opportunities

All opportunities

✓ My territories' opportunities

My territory team's opportunit...

Differences - Territory Management (1.0) and Enterprise Territory Management (2.0)

NOTE: Please note that (x) means "Available" in the below table

Features	Territory Management 1.0	Enterprise Territory Management 2.0
Multiple Territories/Hierarchy		X
Run Territories on Territory Tree/List View Page		X
Preview Territory	X(partial, not persisted)	X
Inherited Territory Rules	X	X
Territory Type/Priority		X
Territory Models		X
Enable/Disable Territory Management	X	X
Assignment of Territory on Opportunities	X	X (Spring '15)
Integration with Customizable Forecasting	X	
Integration with Collaborative Forecasts		X (forecasts based on territory hierarchy, not role hierarchy)
Manual Assignment of Account to Territory	X	X
Separation of Rule Execution vs Deployment		X
Reports/Dashboards	X	X
Territory Hierarchy Deep Clone		X
Rule Sharing among multiple Territories		X
My Territories Scope in Account List Views	X	X
My Territories Scope in Account Reports	X	X (Spring '15)
Audit Trail		X
Metadata API Support		X
User Role in Territory		X
Trigger on User to Territory Association Object		X
Share a report/dashboard folder with a territory	X	
Create a public group with territory	X	

Permissions Affect Enterprise Territory Management?

- Sales Operations managers and selected Sales managers to be able to manage territories.
 - If so, assign them the **“Manage Territories”** permission.
- Anyone who will also create account assignment rules also
 - needs the **“View All” permission on Accounts**.
- Salesforce Setup tree, including territories settings
 - **View Setup and Configuration**

Considerations:

- If using Enterprise Territory Management, territory sharing groups can't be used in a sharing rule.
- If using Enterprise Territory Management, users can't manually share a record to a territory.
- If using Enterprise Territory Management, you can't use territory sharing groups programmatically

Account & Opportunity Teams

Account Teams

- An account team is a team of users who work together on an account.
- Use account teams to easily track collaboration on accounts.
- Account teams aren't the same as opportunity teams, although they have the same team member roles.

Set Up and Manage Account Teams

Enable Account Teams

- From Setup, enter Account Teams in the Quick Find box, then select Account Team Settings.
- Define the settings.
- Save your changes.

Customize Account Team Roles

Every account team member has a role in working with that account, such as Account Manager or Sales Rep. To track the roles that team members play in your company, customize your account team roles in Salesforce.

- From Setup, enter Team Roles in the Quick Find box, then select Team Roles under Account Teams.
- Edit the picklist values for team roles as needed.
- Save your changes.
- To update a changed picklist value in all your files, enter Replace Team Roles in the Quick Find box, then select Replace Team Role.

Considerations

- Account teams share roles with opportunity teams. If you remove an account team role, that role is no longer listed as an opportunity team role.
- Account teams can only be used to grant greater access to an account. They can't be used to restrict access to account records beyond the org-wide sharing defaults.
- When the org-wide default for contacts is set to Controlled by Parent, Contact Access isn't available for account team members.
- Disabling account teams irreversibly removes existing teams from all accounts and delete users' default account teams, and removes the Account Team related list from all page layouts.
- You cannot disable account teams for your organization if team members are referenced in Apex.
- Only administrators can grant access to child records that are greater than the account owner's access level.

Add Account Team Members

Account record owners and users above the owner in the role hierarchy can add, edit, and delete team members.

To edit or delete an account team member, you must be one of the following.

- The account owner
- Above the owner in the role hierarchy
- Any user granted full access to the record
- An administrator

Considerations for Removing Account Team Members

- If a team member is on your default account team and you remove them from a specific account, those changes only affect that account. The setup of your default account team does not change.
- If a user on an account team has Read/Write access (Account Access, Contact Access, Opportunity Access, and Case Access) and is deactivated, the access will default to Read Only if the user is reactivated.

Account Team Fields

FIELD	DESCRIPTION
Account Access	The level of access that a team member has to the account. The access level can be read/write or read only, but it can't be less than your Salesforce org's default account sharing access.
Case Access	The level of access that a team member has to the cases associated with the account.
Contact Access	The level of access that a team member has to the contact. The access level can be read/write or read only, but it can't be less than your Salesforce org's default contact sharing access.
Opportunity Access	The level of access that a team member has to the opportunities associated with the account.
Team Member	The user who's listed as part of the team.
Team Role	The role that the team member plays for the account, such as Account Manager.

Opportunity Teams

- Opportunity teams show who's working on the opportunity and what each team member's role is, making it easy to collaborate with your colleagues.
- You can grant your opportunity team members special access to the opportunity and its related records, making it easier for everyone to work together.
- In opportunity reports, filter opportunities by the opportunity teams that you're a member of.

Understanding Sharing

A share object includes records supporting all three types of sharing:

1. Managed sharing
2. user managed sharing,
3. Apex managed sharing.

Managed Sharing

Managed sharing involves sharing access granted by Lightning Platform based on record ownership, the role hierarchy, and sharing rules:

- Record Ownership
 - Each record is owned by a user or optionally a queue for custom objects, cases and leads. The *record owner* is automatically granted Full Access, allowing them to view, edit, transfer, share, and delete the record
- Role Hierarchy
 - enables users above another user in the hierarchy to have the same level of access to records owned by or shared with users below
- Sharing Rules
 - used by administrators to automatically grant users within a given group or role access to records owned by a specific group of users.
 - Sharing rules cannot be added to a package and cannot be used to support sharing logic for apps installed from AppExchange.
 - Sharing rules can be based on record ownership or other criteria. You can't use Apex to create criteria-based sharing rules. Also, criteria-based sharing cannot be tested using Apex.
- All implicit sharing added by Force.com managed sharing cannot be altered directly using the Salesforce user interface, SOAP API, or Apex.

User Managed Sharing

- allows the record owner or any user with Full Access to a record to share the record with a user or group of users.
- This is generally done by an end user, for a single record

Apex Managed Sharing

- provides developers with the ability to support an application's particular sharing requirements programmatically through Apex or the SOAP API.
- This type of sharing is similar to managed sharing.

The Sharing Reason Field

In the Salesforce user interface, the Reason field on a custom object specifies the type of sharing used for a record. This field is called `rowCause` in Apex or the API.

Sharing	<code>rowCause</code> Value (Used in Apex or the API)
---------	---

Managed	ImplicitChild, ImplicitParent
User Managed	Manual
Apex Managed	Defined by developer

Access Levels

When determining a user's access to a record, the most permissive level of access is used. Most share objects support the following access levels:

Access Level	API Name	Description
Private	None	Only the record owner and users above the record owner in the role hierarchy can view and edit the record. This access level only applies to the AccountShare object.
Read Only	Read	The specified user or group can view the record only.
Read/Write	Edit	The specified user or group can view and edit the record.
Full Access	All	The specified user or group can view, edit, transfer, share, and delete the record. This access level can only be granted with managed sharing.

Sharing Considerations

- If a trigger changes the owner of a record, the running user must have read access to the new owner's user record if the trigger is started through the following:
 - API
 - Standard user interface
 - Standard Visualforce controller
 - Class defined with the with sharing keyword
- If a trigger is started through a class that's not defined with the with sharing keyword, the trigger runs in system mode. In this case, the trigger doesn't require the running user to have specific access.

Security and Sharing in Customer & Partner Community

Share Group

- Share group allows you to specify the Salesforce other external users who can access records owned by high-volume community users.
- Deactivating a share group removes all other users' access to records owned by high-volume community users. An email isn't sent to you when the deactivation process finishes.

Sharing Sets

Grant portal or community users access, based on their user profiles, to records that are associated with their accounts or contacts using sharing sets.

- Access granted to users via sharing sets does not roll up to users higher to them in their role hierarchies.
- the share groups functionality isn't available to users with Customer Community Plus and Partner Community licenses.

Objects Supported

- | | |
|---|--|
| <ul style="list-style-type: none">• Account<ul style="list-style-type: none">◦ Account sharing sets can control access to Contract, Entitlement, and OrderItem objects• Asset• Campaign (in beta)• Case• Contact• Custom Objects | <ul style="list-style-type: none">• Individual• Opportunity (in beta)• Order (in beta)• ServiceAppointment• Service Contract• User• Work Order |
|---|--|

User licenses

- | | |
|---|--|
| <ul style="list-style-type: none">• Authenticated Website• Customer Community Login• Customer Community Plus• Partner Community Licenses (new) | <ul style="list-style-type: none">• Customer Community User• High Volume Customer Portal• High Volume Portal• Overage Authenticated Website User• Overage High Volume Customer Portal User |
|---|--|
- Portal or community users gain access to all order entitlements and order items under an account to which they have access. To share records owned by high-volume portal users, use a share group instead.

Usage

Sharing Set

Community User Sharing Set

H

[« Back to Customer Portal Setup](#)

Sharing Set

Edit

Delete

Name Community User Sharing Set

Description Allow read/write on any cases associated to a Community User's Contact.

Applies to Profiles

Name

[Community User](#)

Access Granted

Action	Object	Access Determined By	Access Level
Edit Del	Case	Contact: Contact Name	Read/Write

Sharing Data with Partner Users

- Sharing groups and a sharing rule category are available by default in your org to share Salesforce data with partner users in a community.
- Org-wide defaults and field-level security also control data access for partners in communities. Set the Default External Access setting to Private for all the objects you want to expose to partner users in your community.

Groups/Categories

After you buy partner licenses for your org, the following groups and sharing rule category are created:

GROUP OR CATEGORY	DESCRIPTION
All Partner Portal Users group	Contains all partner users in your organization
All Internal Users group	Contains all Salesforce users in your organization
Roles and Internal Subordinates sharing rule category	Allows you to create sharing rules in which you can choose specific Salesforce users in your organization by role, including users in roles below the selected role. Partner roles are excluded.

Usage

The screenshot shows the 'Setup Sharing Settings' page in Salesforce. The left sidebar contains a search bar with 'sharing' and a 'Security' section with 'Sharing Settings' highlighted. The main content area is divided into five steps:

- Step 1: Rule Name**: Fields for Label, Rule Name (containing 'Share_With_Sales_Exec'), and Description.
- Step 2: Select your rule type**: Radio buttons for 'Based on record owner' and 'Based on criteria' (selected and highlighted with a blue box).
- Step 3: Select which records to be shared**: A table with columns 'Criteria', 'Field', 'Operator', 'Value', and a logical connector. The first row is 'Billing City', 'equals', 'San Francisco', and 'AND'. Subsequent rows are empty with '--None--' in the Field and Operator columns. A blue box highlights the first row. Below the table is an 'Add Filter Logic...' link.
- Step 4: Select the users to share with**: A 'Share with' section with a dropdown menu showing 'Roles' and 'Sales Executive'.
- Step 5: Select the level of access for the users**: Two dropdown menus for 'Default Account and Contract Access' (set to 'Read Only') and 'Opportunity Access' (set to 'Private').

Annotations in blue text are placed over the interface: 'Step 1: Select "Based on Criteria"' points to the 'Based on criteria' radio button, and 'Step 2: Enter criteria' points to the criteria table.

Apex Managed Sharing

Sharing a Record Using Apex

To access sharing programmatically, you must use the share object associated with the standard or custom object for which you want to share

- StandardObjectShare
 - AccountShare
 - ContactShare
- CustomObject__Share
 - UnitTree__Share

Share Object Properties

Property Name	Description
objectNameAccessLevel	<p>The level of access that the specified user or group has been granted for a share sObject. Valid values are:</p> <ul style="list-style-type: none"> ● Edit ● Read ● All <p>The All access level can only be used by managed sharing.</p>
ParentID	The ID of the object. This field cannot be updated.

RowCause	<p>The reason why the user or group is being granted access. The reason determines the type of sharing, which controls who can alter the sharing record.</p> <p>This field cannot be updated.</p>
UserOrGroupId	<p>The user or group IDs to which you are granting access. A group can be:</p> <ul style="list-style-type: none"> • A public group or a sharing group associated with a role. • A territory group if you use the original version of Territory Management, but not with Enterprise Territory Management. <p>This field cannot be updated.</p>

Creating User Managed Sharing Using Apex

- It is possible to manually share a record to a user or a group using Apex or the SOAP API.
- If the owner of the record changes, the sharing is automatically deleted
- Manual shares written using Apex contains **RowCause="Manual"** by default. Only shares with this condition are removed when ownership changes.

Creating Apex Managed Sharing

- This type of sharing is similar to managed sharing.
- Apex managed sharing must use an Apex sharing reason.
- Apex sharing reasons are a way for developers to track why they shared a record with a user or group of users.
- Using multiple Apex sharing reasons simplifies the coding required to make updates and deletions of sharing records.
- They also enable developers to share with the same user or group multiple times using different reasons.

Each Apex sharing reason has a label and a name:

- The label displays in the Reason column when viewing the sharing for a record in the user interface.
 - This label allows users and administrators to understand the source of the sharing.
 - The label is also enabled for translation through the Translation Workbench.
- The name is used when referencing the reason in the API and Apex.

Apex sharing reasons can be referenced programmatically as follows:

```
Schema.CustomObject__Share.rowCause.SharingReason__c
```

Apex Sharing Reason Creation

- **Apex sharing reasons and Apex managed sharing recalculation are only available for custom objects.**
1. From the management settings for the custom object, click New in the Apex Sharing Reasons related list.
 2. Enter a label for the Apex sharing reason.
 - a. The label displays in the Reason column when viewing the sharing for a record in the user interface. The label is also enabled for translation through the Translation Workbench.
 3. Enter a name for the Apex sharing reason. The name is used when referencing the reason in the API and Apex.
 - a. This name can contain only underscores and alphanumeric characters, and must be unique in your org.
 - b. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
 4. Click Save.

Considerations

- Under certain circumstances, inserting a share row results in an update of an existing share row.
- If an account sharing rule is created, the sharing rule row cause (which is a higher access level) replaces the parent implicit share row cause, indicating the higher level of access.
- When packaging custom objects, be aware that associated Apex sharing recalculations are also included and may prevent the package from installing.
- Deleting an Apex sharing reason will delete all sharing on the object that uses the reason.
- You can create up to **10 Apex sharing reasons per custom object**.
- You can create Apex sharing reasons using the Metadata API.

Creating Apex Managed Sharing for Customer Community Plus users

- Share objects, such as AccountShare and ContactShare, aren't available to these users.

Ways to share:

- If you must use share objects as a Customer Community Plus user, consider using a trigger, which operates with the without sharing keyword by default
- Use an inner class with the same keyword to enable the DML operation to run successfully.

Apex Sharing Recalculation

- When packaging custom objects, be aware that associated Apex sharing recalculations are also included and may prevent the package from installing.
- Developers can write batch Apex classes that recalculate the Apex managed sharing for a specific custom object.
- You can associate these classes with a custom object on its detail page, and execute them if a locking issue prevents Apex from granting access to a user as defined by the application's logic.
- Apex sharing recalculations are also useful for resolving visibility issues due to coding errors.
- You can also run them programmatically using the Database.executeBatch method
- Salesforce automatically recalculates sharing for all records on an object when its organization-wide sharing default access level changes.

Associate an Apex managed sharing recalculation class

1. From the management settings for the custom object, go to Apex Sharing Recalculations.
2. Choose the Apex class that recalculates the Apex sharing for this object.
 - a. The class you choose must implement the Database.Batchable interface.
 - b. You cannot associate the same Apex class multiple times with the same custom object.
3. Click Save.

Considerations for recalculations

- The Apex code that extends the sharing recalculation can process **a maximum of five million records**.
 - If this Apex code affects more than five million records, the job fails immediately.
- You can monitor the status of Apex sharing recalculations in the Apex job queue.
- You can associate a maximum of five Apex sharing recalculations per custom object.
- You cannot associate Apex sharing recalculations with standard objects.

With Sharing

The with sharing keyword allows you to specify that the sharing rules for the current user are taken into account for a class.

```
public with sharing class sharingClass {
}
```

Without Sharing

Use the without sharing keywords when declaring a class to ensure that the sharing rules for the current user are not enforced.

```
public without sharing class noSharing {
}
```

Implementation Details in regards to sharing and without sharing Keywords

- If a class isn't declared as either with or without sharing, the current sharing rules remain in effect.
 - if the class is called by another class that has sharing enforced, then sharing is enforced for the called class.
- Both inner classes and outer classes can be declared as with sharing.
- Inner classes do not inherit the sharing setting from their container class.
- Classes inherit this setting from a parent class when one class extends or implements another
- Sharing doesn't depend on whether the class executes asynchronously as a scheduled job or batch job. If your class accesses standard or custom fields, prevent sharing violations by declaring the "with sharing" keyword.

Inherited Sharing

- An explicit inherited sharing declaration makes the intent clear, avoiding ambiguity arising from an omitted declaration or false positives from security analysis tooling.
- Using inherited sharing enables you to pass AppExchange Security Review and ensure that your privileged Apex code is not used in unexpected or insecure ways.
- An Apex class with inherited sharing runs as with sharing when used as a Lightning component controller, a Visualforce controller, an Apex REST service, or any other entry point to an Apex transaction.
- There is a distinct difference between an Apex class that is marked with inherited sharing and one with an omitted sharing declaration.
- A class declared as inherited sharing runs as without sharing only when explicitly called from an already established without sharing context.
 - Because of the inherited sharing declaration, only contacts for which the running user has sharing access are displayed.
 - If the declaration is omitted, even contacts that the user has no rights to view are displayed due to the insecure default behavior of omitting the declaration.

```
public inherited sharing class InheritedSharingClass{
    public List<Contact> getAllTheSecrets(){
        return [SELECT Name FROM Contact];
    }
}
```




Enforcing Sharing Rules

Enforcing the current user's sharing rules can impact:

- SOQL and SOSL queries. A query may return fewer rows than it would operating in system context.
- DML operations. An operation may fail because the current user doesn't have the correct permissions.

- For example, if the user specifies a foreign key value that exists in the organization, but which the current user does not have access to.

Who Can See My File?

SHARING SETTING	DEFINITION	WHEN DOES A FILE HAVE THIS SETTING?
 Private	<ul style="list-style-type: none"> The file is private. It hasn't been shared with anyone else besides the owner. The file owner and users with “Modify All Data” permission can find and view this file. However, if the file is in a private library, only the file owner has access to it. 	<p>A file is private when you:</p> <ul style="list-style-type: none"> Upload it in Files home Publish it to your private library Stop sharing it with everyone (Make Private) Delete posts that include the file and the file isn't shared anywhere else
 Privately Shared	<ul style="list-style-type: none"> The file has only been shared with specific people, groups, or via link. It's not available to all users in your company. Only the file owner, users with “Modify All Data” or “View all Data” permission, and specific file viewers can find and view this file. 	<p>A file is privately shared when it's:</p> <ul style="list-style-type: none"> Only shared with specific people or a private group Posted to a private group Shared via link Posted to a feed on a record Published to a shared library
 Your Company	All users in your company can find and view this file.	A file is shared with your company when it's posted to a feed that all users can see, a profile, a record, or a public group.

Actions for your file permissions.

ACTION	FILE OWNER	FILE COLLABORATOR	FILE VIEWER
View or Preview	Yes	Yes	Yes
Download	Yes	Yes	Yes
Share	Yes	Yes	Yes
Attach a File to a Post	Yes	Yes	Yes
Upload New Version	Yes	Yes	
Edit Details	Yes	Yes	
Change Permission	Yes	Yes	
Make a File Private	Yes		
Restrict Access	Yes		
Delete	Yes		

Considerations

- No Access means that only the people in your company with whom this file is shared can find or view the file. If the file is shared with a private group, only members of the group can find or view the file.
- Users with “Modify All Data” permission can view, preview, download, share, attach, make private, restrict access, edit, upload new versions, and delete files they don't own. However, if the file is in a private library, then only the file owner has access to it.
- Users with “View All Data” permission can view and preview files they don't own. However, if the file is in a private library, then only the file owner has access to it.
- Groups (including group members) and records have viewer permission for files posted to their feeds.
- Permissions for files shared with libraries depend on the library.

Create a Custom List View in Salesforce Classic

USER PERMISSIONS NEEDED

- To create custom list views:
 - Read on the type of record included in the list AND Create and Customize List Views
- To create, edit, or delete public list views:
 - **Manage Public List Views**

Considerations

- As a Salesforce admin or a user with the “Manage Public List View” permission, you have the option to hide the list view, so only you can see this list view.
 - Open the list view. Select Visible to certain groups of users. Choose the type of group or role from the drop-down list, select the group or role from the list, then click Add.
- Enterprise, Unlimited, Performance, and Developer Edition users can give access to a public group or role, including all users below that role.
- List views are visible to your community users with Customer Community Plus, Partner Community, Lightning Platform Starter, and Lightning Platform Plus licenses,
 - if the Visible to all users setting is enabled for views of objects in community user profiles.
- To make list views visible only to your Salesforce users, select Visible to certain groups of users. Then share the view with the All Internal Users group or a selected set of internal groups and roles.
- When implementing a community, create custom views that contain only relevant information for community users. Then make those views visible to community users by sharing them with the All Customer Portal Users group, or a set of community groups and roles.

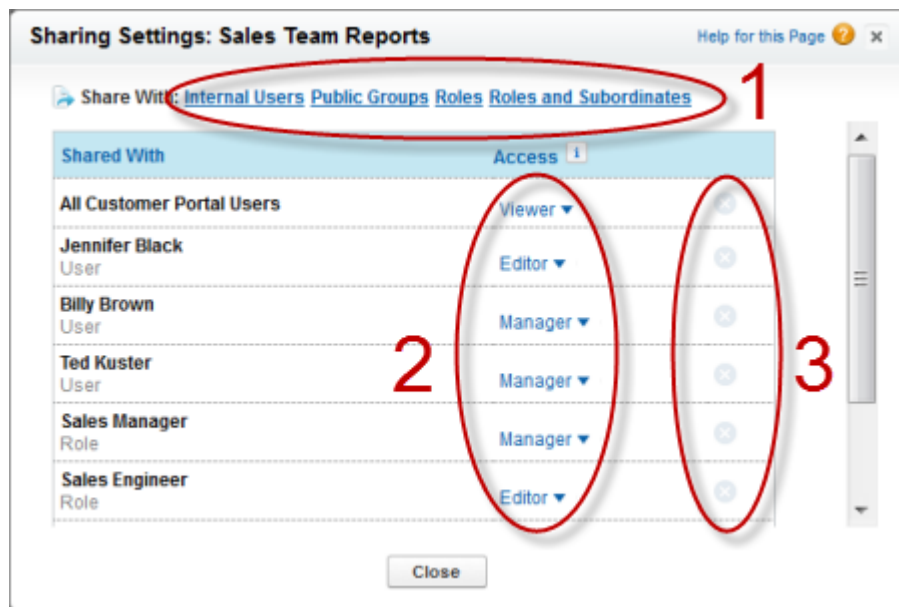
Share a Report or Dashboard Folder in Salesforce Classic

USER PERMISSIONS NEEDED

- To share a report folder with public groups:
 - **Run Reports AND Manage Dashboards** OR
 - **Manage Reports in Public Folders**
- To share a dashboard folder with public groups:
 - **Run Reports AND Manage Dashboards** OR
 - **Manage Reports in Public Folders**

Access and Limits

- When you create a folder, you’re its manager.
 - Only you and others with administrative permissions can see it.
- If a folder does not have Manager access, it’s public, and users with the View Reports in Public Folders permission can view it.
- You can share a report or dashboard folder with up to 25 users, groups, roles, or territories at one time.
- You can share a folder with up to 100 users, groups, roles, or territories using the folder sharing REST API.



Designing Record Level Access for Enterprise Scale

Surviving Owner Change Operations

- So when the owner of a record is changed, the platform deletes all the manual shares associated with the record.
- In effect, we “clean the slate” for the new owner and let them decide whether they want to share it to anybody.
- And if you have been writing code that shares the record, your shares will get deleted, too, because they have the same ‘manual’ row cause—the platform cannot distinguish between a sharing row you created and a sharing row created through the UI

Using Apex Sharing Reasons

- Because your row cause for these shares will no longer be ‘manual’, the platform won’t touch them when performing the change owner operation.
- Standard objects like accounts or contacts?
 - The relationship between these objects can be complex, and there might be good reasons for the platform to change or delete a sharing row, even one that you have created programmatically.

Using Outbound Messaging

- With standard objects, where you can’t use a custom sharing reason, and you are integrating with an assignment engine external to Salesforce.
- You can configure a workflow rule to detect when a record owner is changed, and use an outbound message to trigger your assignment engine to take appropriate action
- Enter “OwnerId <>PRIORVALUE(OwnerId)” for the formula.

Using a Trigger

- Applies to standard objects where you can’t use a custom sharing reason, but in this case you are integrating with an assignment engine built on the Salesforce platform.

Using a Shadow Table

- Your logic for standard objects might not be complex enough to justify building a full-blown assignment engine.
- You might be able to accomplish the same goal through the use of a trigger and a custom object that keeps track of your programmatic shares.

Account: Lookup to Account

Team Member: Lookup to User

Account Access: Picklist (Read, Edit)

Opportunity Access: Picklist (None, Read, Edit)

Case Access: Picklist (None, Read, Edit)

Contact Access: Picklist (None, Read, Edit)

Team Role: Picklist (Account Manager, Channel Manager, Executive Sponsor, Lead Qualifier, Pre-Sales Consultant, Sales Manager, Sales Rep)

Completing the Architecture

there are additional platform features that could impact the sharing system you have built, which you can't code around.

- A user with appropriate access to a record can change or remove your programmatic shares through the sharing button on the record's detail page.
- A user with permission to update the membership of an account team can change or remove shares your code has written to manage team membership.
 - This does not apply to sales teams, because the platform now includes the ability to define triggers for the sales team object that you can use to protect your shares.
- Any Apex or API operation performing DML on the sharing objects could also impact your sharing system.

Group Maintenance Tables

- Sharing rows grant access to users and groups, but the data that specifies who belongs to each group resides in the Group Maintenance tables.
- These tables store membership data for every Salesforce group, including system-defined groups.
 - System-defined groups are groups of users that Salesforce creates and manages internally to support various features and behaviors, such as queues.
 - role hierarchy
 - territory hierarchy
 - queues
 - User-defined groups in Salesforce are groups that directly modify the group membership object. They differ from system-defined groups in that you cannot directly modify system-defined groups
 - public groups
 - private groups
- This type of management lets the data that supports queues and personal or public groups coexist in the same database tables, and unifies how Salesforce manages the data.

Groups and Composition

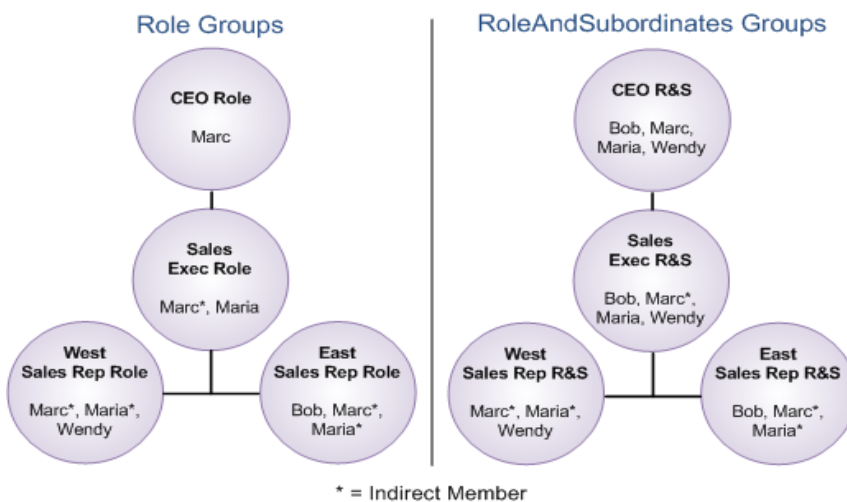
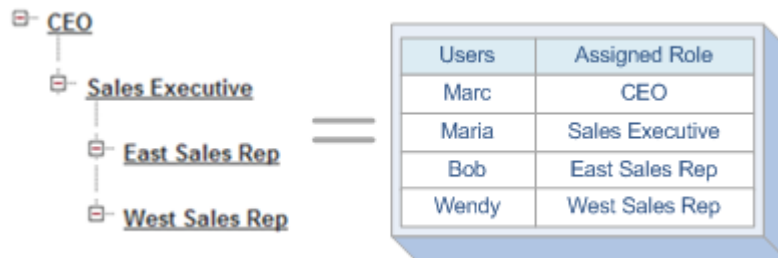
- Salesforce also uses system-defined groups to implement hierarchies.
- During recalculation, Salesforce creates two types of system-defined groups, Role groups and RoleAndSubordinates groups, for every node in the role hierarchy.
- If the organization has external organization-wide defaults enabled, a third type of system-defined group, RoleAndInternalSubordinates, is created.

Group	Consists of	Purpose
Role	Users assigned to any of the following. <ul style="list-style-type: none"> • A specific role • One of its manager roles 	Used to give managers access to their subordinates' records
RoleAndSubordinates	Users assigned to any of the following. <ul style="list-style-type: none"> • A specific role • One of its manager roles • One of its subordinate roles 	Used when an organization defines a rule that shares a set of records with: <ul style="list-style-type: none"> • A particular role • Its subordinates
RoleAndInternalSubordinates	Users assigned to any of the following. <ul style="list-style-type: none"> • A specific role • One of its manager roles • One of its subordinate roles, excluding Portal roles 	Used when an organization defines a rule that shares a set of records with: <ul style="list-style-type: none"> • A particular role • Its subordinates, excluding Portal roles

All three group types have:

- Indirect members, who inherit record access from the group's direct members and are assigned to manager roles.
- Direct members, who are defined according to their group type

Example



Territory Management Groups

- Territory group, in which users who are assigned to the territory are direct members, while users assigned to territories higher in the hierarchy are indirect members
- TerritoryAndSubordinates group, in which users who are assigned to that territory or territories lower in the hierarchy are direct members, while users assigned to territories higher in that branch are indirect members

Considerations

- Users can't modify system-defined groups through the user interface or API in the ways that they can personal and public groups

Obtain peak performance:

- Moving users from one group to another trigger organization wide group membership locks, so highly dynamic groups can have a negative impact on performance.
- The use case which will provide peak performance includes a group of users who share the same visibility and don't frequently move from one group to another via an automated process.

- The sharing performance benefit will decrease as the number of group members decreases, and the frequency of user movement within the groups increases.

Access Grants

When an object has its organization-wide default set to Private or Public Read Only, Salesforce uses access grants to define how much access a user or group has to that object's records.

Salesforce uses four types of access grants:

1. Explicit Grants
 - a. Salesforce uses explicit grants when records are shared directly to users or groups. Specifically, Salesforce uses explicit grants when:
 - i. A user or a queue becomes the owner of a record.
 - ii. A sharing rule shares the record to a personal or public group, a queue, a role, or a territory. 1
 - iii. An assignment rule shares the record to a user or a queue.
 - iv. A territory assignment rule shares the record to a territory.
 - v. A user manually shares the record to a user, a personal or public group, a queue, a role, or a territory.2
 - vi. A user becomes part of a team for an account, opportunity, or case.
 - vii. A programmatic customization shares the record to a user, a personal or public group, a queue, a role, or a territory
 - b. If your organization doesn't have an efficient sharing architecture, it might encounter performance problems when you use automated processes that generate a very large number of explicit grants, such as major sales realignments
 - c. d
2. Group Membership Grants
 - a. Grants that occur when a user, personal or public group, queue, role, or territory is a member of a group that has explicit access to the record.
3. Inherited Grants
 - a. Grants that occur when a user, personal or public group, queue, role, or territory inherits access through a role or territory hierarchy, or is a member of a group that inherits access through a group hierarchy.
4. Implicit Grants
 - a. Ss Grants that occur when non-configurable record-sharing behaviors built into Salesforce Sales, Service, and Portal applications grant
 - b. access to certain parent and child records.

Common Group and Data Updates

Instead of moving a user from one branch of the hierarchy to another, we can:

- Moving a role to another branch in the hierarchy
 - One benefit to moving a whole role is that any portal accounts simply move along with their parent role, and Salesforce doesn't have to change the related sharing.
 - On the other hand, Salesforce must do all of the work involved in moving a single user for all users in the role being moved and for all of those users' data
- Changing the owner of a portal account
 - The effort required for what looks like a simple data update—changing the name of the user in the Account Owner field—can be surprising.

- When the old and new owners are in different roles, Salesforce is not only moving the portal roles to a new parent role but also adjusting the sharing for all the data associated with the portal account.

Group Membership Locking

Customers can lessen the chance of locking errors by:

- Scheduling separate group maintenance processes carefully so they don't overlap
- Implementing retry logic in integrations and other automated group maintenance processes to recover from a failure to acquire a lock
- Using the granular locking feature to allow some group maintenance operations to proceed simultaneously

Takeaway: Tuning Group Membership for Performance

Here are some specific suggestions.

- Identify user and group updates that are complex, such as user role and portal account ownership changes, or updates that involve a large amount of associated data. Allow for additional time to process these changes.
- When making changes to the hierarchy, process changes to the bottom (leaf) nodes first, then move upward to avoid duplicate processing.
- Limit the number of records of an object owned by a single user to 10,000.
- Run group maintenance operations single threaded to prevent locking. Investigate whether the use of granular locking will allow some of your operations to run simultaneously.
- Tune your updates for maximum throughput by experimenting with batch sizes and using the bulk API, where possible.
- Remove redundant paths of access, such as sharing rules that provide access to people who already have it through the hierarchy

Takeaway: Tuning Data Relationships and Updates for Performance

Here are some specific suggestions.

- Use a Public Read Only or Read/Write organization-wide default sharing model for all non-confidential data.
- To avoid creating implicit shares, configure child objects to be Controlled by Parent wherever this configuration meets security requirements.
- Configure parent-child relationships with no more than 10,000 children to one parent record.
- If you are encountering only occasional locking errors, see if the addition of retry logic is sufficient to solve the problem.
- Sequence operations on parent and child objects by ParentID and ensure that different threads are operating on unique sets of records.
- Tune your updates for maximum throughput by working with batch sizes, timeout values, the Bulk API, and other performance-optimizing techniques

Force.com Record Locking Cheatsheet

http://resources.docs.salesforce.com/194/0/en-us/sfdc/pdf/record_locking_cheatsheet.pdf

Tools for Large-Scale Realignment

Parallel Sharing Rule Recalculation

- Normally, when an administrator creates, deletes, or edits a sharing rule, the recalculation required to make those changes take effect is processed synchronously.
- When a sharing rule change affects access rights to a very large amount of data, the recalculation can run longer. In addition, a recalculation job can get killed if it is running when Salesforce performs a scheduled feature or patch release.
- If you have experienced long-running processing times or jobs that were killed during realignments, consider using parallel sharing rule recalculation.
- When this feature is turned on, sharing rules are processed asynchronously and split into multiple simultaneous execution threads based on load.
- The processing is also more resilient; during a server restart, the jobs will be reinstated on the queue, and the process will continue when the server comes back online.

Deferred Sharing Maintenance

- In an enterprise environment in which multiple systems are continually processing updates, it can be difficult to schedule an organization or sharing rule change that might take substantial time to complete.
- In order to increase the predictability of these kinds of updates, the Lightning Platform platform has recently introduced the concept of deferred sharing maintenance.

How works in practice

1. Based on requests from the business, an administrator identifies a number of changes to the role hierarchy and group membership, or updates to sharing rules.
2. Given best estimates of the remaining overall work, the administrator negotiates a maintenance window for completing the processing.
3. This window should be modelled in a sandbox environment to get the best estimate possible.
4. Instead of processing each separate update and waiting for it to complete, the administrator prepares all the information required to perform all updates ahead of the planned maintenance window.
5. At the start of the maintenance window, the administrator uses the deferral feature to essentially “turn off” processing of group maintenance operations, and then makes all the desired changes to role and group membership at the same time.
6. Sharing rule processing is also deferred at this time so the administrator can perform all sharing rule updates.
7. Once the changes have completed, the administrator resumes processing group maintenance, and the system performs a recalculation to make all the role and group changes take effect.
8. At this point, the system is in a state that requires a full recalculation of all sharing rules for user access rights to be complete and accurate. The administrator can resume sharing rule processing immediately or wait to start the process at a later time. After the sharing rule recalculation has completed, all the access changes take effect.

How it helps

- Benchmark how long the overall recalculation is likely to take in production
- Smooth out any kinks in orchestrating deferred sharing maintenance
- Deferred sharing maintenance does not defer the recalculation of implicit sharing as described in the [implicit sharing table](#). The cascading effects to implicit shares continue to be processed immediately when sharing rules are changed by administrators or through the code.

Considerations:

- if you are able to negotiate downtime with your business customers and have been struggling to complete updates in a timely fashion, deferred sharing might be a great solution to your problem.

Granular Locking

- Lightning Platform platform locks the entire group membership table to protect data integrity when Salesforce makes changes to roles and groups.
- This locking makes it impossible to process group changes in multiple threads to increase throughput on updates.
- When the granular locking feature is enabled, the system employs additional logic to allow multiple updates to proceed simultaneously if there is no hierarchical or other relationship between the roles or groups involved in the updates.
- Administrators can adjust their maintenance processes and integration code to take advantage of this limited concurrency to process large-scale updates faster, all while still avoiding locking errors.

Key Advantages

- Groups that are in separate hierarchies are now able to be manipulated concurrently.
- Public groups and roles that do not include territories are no longer blocked by territory operations.
- Users can be added concurrently to territories and public groups.
- User provisioning can now occur in parallel.
 - Portal user creation requires locks only if new portal roles are being created.
 - Provisioning new portal users in existing accounts occurs concurrently.
- A single-long running process, such as a role delete, blocks only a small subset of operations.

See table for more details in the other document

Considerations:

- The user must not own any partner or customer portal accounts.
- Customers may consider using granular locking if they experience frequent and persistent locking that severely restricts their ability to manage manual and automated updates at the same time, or severely degrades the throughput of integrations or other automated group maintenance operations.

Classic Encryption for Custom Fields

Restrictions

Encrypted text fields:

- Cannot be unique, have an external ID, or have default values.
- For leads are not available for mapping to other objects.
- Are limited to 175 characters because of the encryption algorithm.
- Are not available for use in filters such as list views, reports, roll-up summary fields, and rule filters.
- Cannot be used to define report criteria, but they can be included in report results.
- Are not searchable, but they can be included in search results.
- Are not available for: Connect Offline, Salesforce for Outlook, lead conversion, workflow rule criteria or formulas, formula fields, outbound messages, default values, and Web-to-Lead and Web-to-Case forms.

Best Practices

- Encrypted fields are editable regardless of whether the user has the “View Encrypted Data” permission.
 - Use validation rules, field-level security settings, or page layout settings to prevent users from editing encrypted fields.
- You can still validate the values of encrypted fields using validation rules or Apex.
- Both work regardless of whether the user has the “View Encrypted Data” permission.
- Encrypted field data is not always masked in the debug log.
 - Encrypted field data is masked if the Apex request originates from an Apex Web service, a trigger, a workflow, an inline Visualforce page (a page embedded in a page layout), or a Visualforce email template. In other cases, encrypted field data isn’t masked in the debug log, like for example when running Apex from the Developer Console.
- Existing custom fields cannot be converted into encrypted fields nor can encrypted fields be converted into another data type.
 - To encrypt the values of an existing (unencrypted) field, export the data, create an encrypted custom field to store that data, and import that data into the new encrypted field.
- Mask Type is not an input mask that ensures the data matches the Mask Type.
 - Use validation rules to ensure that the data entered matches the mask type selected.
- Use encrypted custom fields only when government regulations require it because they involve more processing and have search-related limitations.

Salesforce Shield

- Salesforce Shield is a trio of security tools that admins and developers can use to build a new level of trust, transparency, compliance, and governance right into business-critical apps.
- It includes Platform Encryption, Event Monitoring, and Field Audit Trail. Ask your Salesforce administrator if Salesforce Shield is available in your organization.

Platform Encryption

- It enables you to encrypt sensitive data at rest, and not just when transmitted over a network, so your company can confidently comply with privacy policies, regulatory requirements, and contractual obligations for handling private data.
- Data stored in many standard and custom fields and in files and attachments is encrypted using an advanced HSM-based key derivation system, so it is protected even when other lines of defense have been compromised.

Encrypt Fields

1. Make sure that your org has an active encryption key. If you're not sure, check with your administrator.
2. From Setup, in the Quick Find box, enter **Platform Encryption**, and then select Encryption Policy.
3. Click Encrypt Fields.
4. Click Edit.
5. Select the fields you want to encrypt.
6. Click Save.

FEATURE	CLASSIC ENCRYPTION	PLATFORM ENCRYPTION
Pricing	Included in base user license	Additional fee applies
Encryption at Rest	✓	✓
Native Solution (No Hardware or Software Required)	✓	✓
Encryption Algorithm	128-bit Advanced Encryption Standard (AES)	256-bit Advanced Encryption Standard (AES)
HSM-based Key Derivation		✓
Manage Encryption Keys Permission		✓
Generate, Export, Import, and Destroy Keys	✓	✓
PCI-DSS L1 Compliance	✓	✓
Masking	✓	
Mask Types and Characters	✓	
View Encrypted Data Permission Required to Read Encrypted Field Values	✓	
Encrypted Standard Fields		✓
Encrypted Attachments, Files, and Content		✓
Encrypted Custom Fields	Dedicated custom field type, limited to 175 characters	✓

Encrypt Existing Fields for Supported Custom Field Types		✓
Search (UI, Partial Search, Lookups, Certain SOSL Queries)		✓
API Access	✓	✓
Available in Workflow Rules and Workflow Field Updates		✓
Available in Approval Process Entry Criteria and Approval Step Criteria		✓

Difference Between Classic Encryption and Shield Platform Encryption

With Shield Platform Encryption, you can encrypt a variety of widely used standard fields, along with some custom fields and many kinds of files. Shield Platform Encryption also supports person accounts, cases, search, approval processes, and other key Salesforce features. Classic encryption lets you protect only a special type of custom text field, which you create for that purpose.

Shield Platform Encryption Best Practices

- This process helps you distinguish data that needs encryption from data that doesn't, so that you can encrypt only what you need to.
1. Define a threat model for your organization.
 2. To identify the threats that are most likely to affect your organization.
 - Use your findings to create a data classification scheme, which can help you decide what data to encrypt.
 3. Encrypt only where necessary.
 - Focus on information that requires encryption to meet your regulatory, security, compliance, and privacy requirements. Unnecessarily encrypting data impacts functionality and performance.
 - Balance business-critical functionality against security and risk measures and challenge your assumptions periodically.
 4. Create a strategy early for backing up and archiving keys and data.
 5. If your tenant secrets are destroyed, reimport them to access your data. You are solely responsible for making sure that your data and tenant secrets are backed up and stored in a safe place.
 6. Read the Shield Platform Encryption considerations and understand their implications on your organization.
 - Test Shield Platform Encryption in a sandbox environment before deploying to a production environment. Encryption policy settings can be deployed using change sets.
 - Before enabling encryption, fix any violations that you uncover.
 - When requesting feature enablement, such as pilot features, give Salesforce Customer Support several days lead time.
 7. Analyze and test AppExchange apps before deploying them.
 - If you use an app from the AppExchange, test how it interacts with encrypted data in your organization and evaluate whether its functionality is affected.
 - If an app interacts with encrypted data that's stored outside of Salesforce, investigate how and where data processing occurs and how information is protected.
 - If you suspect Shield Platform Encryption could affect the functionality of an app, ask the provider for help with evaluation. Also discuss any custom solutions that must be compatible with Shield Platform Encryption.
 - Apps on the AppExchange that are built exclusively using Lightning Platform inherit Shield Platform Encryption capabilities and limitations.
 8. Use out-of-the-box security tools.
 - Shield Platform Encryption is not a user authentication or authorization tool.
 9. Grant the Manage Encryption Keys user permission to authorized users only.
 - Users with the Manage Encryption Keys permission can generate, export, import, and destroy organization-specific keys. Monitor the key management activities of these users regularly with the setup audit trail.

10. Existing field and file data is not automatically encrypted when you turn on Shield Platform Encryption. To encrypt existing field data, update the records associated with the field data. To encrypt existing files or get help updating other encrypted data, contact Salesforce.
 - allow at least a week before you need the background encryption completed.
11. Handle currency and number data with care.
 - Currency and Number fields can't be encrypted because they could have broad functional consequences across the platform
12. Communicate to your users about the impact of encryption.
 - Before you enable Shield Platform Encryption in a production environment, inform users about how it affects your business solution.
13. Encrypt your data using the most current key.
 - When you generate a new tenant secret, any new data is encrypted using this key. However, existing sensitive data remains encrypted using previous keys. In this situation, Salesforce strongly recommends re-encrypting these fields using the latest key. Contact Salesforce for help with re-encrypting your data.
14. Use discretion when granting login as access to users or Salesforce Customer Support.
 - that user is able to view encrypted data in that field in plaintext.
15. If you want Salesforce Customer Support to follow specific processes around asking for or using login as access, you can create special handling instructions.
 - To set up these special handling instructions, contact your account executive.

Event Monitoring

Field Audit Trail

- Field Audit Trail lets you know the state and value of your data for any date, at any time.
- You can use it for regulatory compliance, internal governance, audit, or customer service.
- Built on a big data backend for massive scalability, Field Audit Trail helps companies create a forensic data-level audit trail with up to 10 years of history, and set triggers for when data is deleted.

Usage

- Use Salesforce Metadata API to define a retention policy for your field history for fields that have field history tracking enabled.
- Then use REST API, SOAP API, and Tooling API to work with your archived data
- Field history is copied from the History related list into the FieldHistoryArchive big object.
- You define one HistoryRetentionPolicy for your related history lists, such as Account History, to specify Field Audit Trail retention policies for the objects you want to archive.
- You can then deploy the big object by using the Metadata API (Workbench or Ant Migration Tool).

Objects Supported

- | | |
|---------------------------------------|------------------------|
| ● Accounts, including Person Accounts | ● Price Books |
| ● Assets | ● Products |
| ● Cases | ● Service Appointments |
| ● Contacts | ● Service Contracts |

- Contracts
- Contract Line Items
- Entitlements
- Leads
- Opportunities
- Solutions
- Work Orders
- Work Order Line Items
- Custom objects with field history tracking enabled

Field can't be tracked

- Formula, roll-up summary, or auto-number fields
- Created By and Last Modified By
- Expected Revenue field on opportunities
- Master Solution Title or the Master Solution Details fields on solutions
- Long text fields
- Multi-select fields

Considerations

- HistoryRetentionPolicy is automatically set on the supported objects, once Field Audit Trail is enabled.
- By default, data is archived after 18 months in a production organization, after one month in a sandbox organization, and all archived data is stored for 10 years.
- After you define and deploy a Field Audit Trail policy, production data is migrated from related history
- The first copy writes the field history that's defined by your policy to archive storage and sometimes takes a long time.
- A bounded set of SOQL is available to query your archived data.
- Use Async SOQL to build aggregate reports from a custom object based on the volume of the data in the FieldHistoryArchive big object.
- If your organization has Field Audit Trail enabled, previously archived data isn't encrypted if you turn on Platform Encryption later.
- If your organization needs to encrypt previously archived data, contact Salesforce.
 - We encrypt and rearchive the stored field history data, then delete the unencrypted archive.

Data Leak Prevention

Authorization

Access to online data is generally restricted to only those who are

- identified,
- authenticated,
- and authorized.

This is accomplished in three main ways:

1. Create, read, update, and delete (CRUD) settings
Determine which objects a user can create, read, update, and delete
2. Field level security (FLS) settings
Determine which fields a user can read and edit
3. Sharing Rules
Determine which records are visible to users

How the Salesforce Platform Enforces Authorization?

User Context

The platform runs in user context when:

- A user browses the application via the standard Salesforce-provided UI
- A user views a Visualforce page that uses a standard controller
- A user views a Visualforce page that references objects with standard object notation
- The platform executes Anonymous Apex via console or API calls
- An application on the platform makes a standard API call

System Context

- Apex generally runs in system context; that is,
 - the current user's permissions,
 - field-level security,
 - and sharing rules aren't taken into account during code execution.
- The only exceptions to this rule are Apex code that is executed with the
 - `executeAnonymous` call
 - Chatter in Apex.

The platform executes code in system context in:

- Apex Classes (including web services)
- Apex Triggers
- Apex web services called from the API

Purpose of Multiple Contexts

- From a security perspective, user context is preferable because user access controls are maintained throughout the transaction.

- This is why standard pages and Visualforce pages built on standard controllers run in user context.
- Custom Apex and Visualforce applications often require permissions beyond the scope of user's access. System context provides the necessary flexibility for these applications.

CRUD and FLS Enforcement in VisualForce and Lightning

- When rendering VisualForce pages, the platform will automatically enforce CRUD and FLS when the developer references SObjects and SObject fields directly in the VisualForce page.
- If a user without FLS visibility, it be automatically removed from the table.
- Input tags such as apex:inputText and apex:inputTextArea will also automatically enforce FLS restrictions.
- Lightning components don't automatically enforce CRUD and FLS when you reference objects or retrieve the objects from an Apex controller, CRUD and FLS should be enforced when using the "@AuraEnabled" notation.

Protect Against CRUD and FLS Violations

- You can enforce these permissions in your code that check the current user's access permission levels by explicitly calling the
 - sObject describe result methods (of [Schema.DescribeSObjectResult](#)) and field describe result methods (of [Schema.DescribeFieldResult](#))
 - IsCreateable()
 - IsAccessible()
 - IsUpdateable()
 - IsDeleteable()

Is My Application Vulnerable?

- CRUD and FLS always needs to be enforced for create, read, update, and delete operations on standard objects.
- Any application performing creates/updates/deletes in Apex code, passing data types other than SObjects to VisualForce pages, using Apex web services or the "@AuraEnabled" notation should be checked that it is calling the appropriate access control functions.

How Can I Test My Application?

1. Data Displayed to the User
 - a. Examine each VisualForce page and at the areas on the page where data is embedded using merge fields (i.e. {!object.field}).
 - i. Merge fields referencing SObject data through other objects like strings, integers, or Apex classes require that the page controller or controller extension perform the appropriate access control check.
 - b. Apex web services do not have a VisualForce layer to automatically enforce CRUD/FLS and always need to call isAccessible() on all SObject fields before returning data to the user,
 - i. same goes for Lightning components or controllers.
2. Create, Update, and Delete Operations
 - a. Examine each Apex class that calls insert, update, upsert, delete, or similar commands.
 - i. For create and update operations, each field assigned a value directly in Apex should have its describe result isCreateable() or isUpdateable() method checked before performing the operation.
 - ii. Delete operations occur at an object level by nature and the object's describe result isDeleteable() method should be called instead of field-level checks.

- b. Apex web services and aura enabled methods always need to perform the appropriate access control checks on all objects and fields before performing create, update, and delete operations.

runAs Method

- The system method runAs enables you to write test methods that change the user context to an existing user or a new user so that the user's record sharing is enforced.
- The runAs method doesn't enforce user permissions or field-level permissions, only record sharing.
- You can use runAs only in test methods.

Nesting

You can nest more than one runAs method. For example:

```
User u2 = new User(Alias = 'newUser', Email='newuser@testorg.com',
System.runAs(u2) {
    //something
    User u3 = [SELECT Id FROM User WHERE UserName='newuser@testorg.com'];
    System.runAs(u3) {
    }
}
```

Other Uses of runAs

- the runAs method to perform mixed DML operations in your test by enclosing the DML operations within the runAs block.
- There is another overload of the runAs method (runAs(System.Version)) that takes a package version as an argument. This method causes the code of a specific version of a managed package to be used

Injection Vulnerability Prevention

Cross-Site Scripting(XSS)

- XSS is an injection vulnerability that occurs when an attacker can insert unauthorized **JavaScript, VBScript, HTML, or other active content** into a web page.
- When other users view the page, the malicious code executes and affects or attacks the user.

```
basicText=apexpages.currentPage().getParameters().get('text');
outputText = basicText.replace('\r\n', '<br/>');
document.getElementById('{!$Component.textOutput}').innerHTML =
'<p>{!outputText}</p>';
```

```
<img src=x onerror="alert('\I said, HEAR YE, HEAR YE, COME ONE, COME ALL!!\');"></img>
```

Types of XSS Attacks

- **Stored XSS**
 - Stored XSS occurs when a malicious input is permanently stored on a server and reflected back to the user in a vulnerable web application.
- **Reflected XSS**
 - Reflected XSS occurs when malicious input is sent to a server and reflected back to the user on the response page.
- **DOM-based XSS**
 - DOM-based XSS occurs when an attack payload is executed as a result of modifying the web page's document object model (DOM) in the victim user's browser.

Impact of XSS

- Arbitrary requests
 - An attacker can use XSS to send requests that appear to be from the victim to the web server.
- Malware download
 - XSS can prompt the user to download malware. Since the prompt looks like a legitimate request from the site, the user may be more likely to trust the request and actually install the malware.
- Log keystrokes
 - The attacker can monitor keyboard entries, possibly finding usernames and passwords to access accounts at later dates.

Common XSS Mitigations

- Input Filtering
 - Blacklisting — Specific “bad” characters or combinations of characters are banned, meaning they can't be entered or stored.
 - Whitelisting — Only characters or words from a known list of entries are permitted, preventing malicious input
- Output Encoding

Built-in XSS Protections in Lightning Platform

- Automatic HTML Encoding
 - Salesforce automatically HTML encodes any values and merge fields placed in HTML context.

- The platform changed "<" and "<" into "<" and ">" by automatically HTML encoding the special characters. The platform treats the data as text, not code.
 - Disabling Automatic HTML Encoding
 - `<apex:outputText escape="false">`
- Salesforce Default Protections in Different Execution Contexts
 - HTML Context
 - Script Context
 - Stype Context

Prevent XSS in Lightning Platform Applications

- If the value is going to be parsed by the JavaScript parser, use `JSENCODE()`.
- If the value is going to be parsed by the HTML parser, use `HTMLENCODE()`.
- If it's a combination of both ...
 - Use `JSENCODE(HTMLENCODE())`
 - Or `JSINHTMLENCODE()`.

Platform Encoding in Apex

- Salesforce provides various Apex encoding functions through the Lightning Platform ESAPI, which exports global static methods that you can use in your package to perform security encoding.
- This package can be installed in any Salesforce org as an unmanaged package.

SOQL Injection

Impact of SOQL Injection

- Since SOQL is narrower than SQL in terms of what a user can do, SOQL reduces the attack surface and limits what an attacker can do with a vulnerable query.
 - No command execution, therefore no ability to exploit the underlying OS running the Salesforce service.
 - No delete method, therefore no ability to interact destructively.
 - No insert or update methods, therefore no ability to add data, user accounts, or permissions to the system
- An attacker who is able to successfully exploit SOQL injection can access fields that a developer did not intend to reveal or that a user should not ordinarily have access to.

SOQL Injection Prevention

- Static queries with bind variables
- `String.escapeSingleQuotes()`
- Type casting
- Replacing characters
 - Use `var.replaceAll('[^\\w]', '')`
- Whitelisting

Storing Sensitive Data

- Sensitive data is also called personally-identifying information (PII) or high business impact (HBI) data.

Sensitive Data - What is it?

Sensitive data can include:

- Passwords
- Passphrases
- Encryption keys
- OAuth tokens
- Purchase instruments, such as credit card numbers
- Personal contact information such as names, phone numbers, email addresses, account usernames, physical addresses, and more
- Demographic information such as income, gender, age, ethnicity, education
- In some states and countries: machine identifying information such as MAC address, serial numbers, IP addresses, and more

Measures

Hardcoded Secrets

- Storing sensitive information in the source code of your application might not always be a good practice, anyone that has access to the source code can view the secrets in clear text.

Debug Logs

- Debug logs in apex code should not contain any sensitive data
- Sensitive information should also be not be sent to 3rd party by emails or other means as part of reporting possible errors.

Sensitive Info in URL

- Long term secrets like username/passwords, API tokens and long lasting access tokens should not be sent via GET parameters in the query string.
- It is fine to send short lived tokens like CSRF tokens in the URL. Salesforce session id or any PII data should not be sent over URL to external applications.

Salesforce.com Integrations

- External applications should not store Salesforce.com user credentials (usernames, passwords, or session ID's) in external databases.
- In order to integrate an external application with Salesforce.com user accounts, the OAuth flow should be used.

Sample Vulnerability

- If you must store passwords (including non-Salesforce passwords), note that storing them in plaintext or hashed (such as with the MD5 function) makes your application vulnerable to mass user exploitation
- if an attacker can get access (even just read-only access) to your database (such as through stealing a backup tape or SQL injection).
- Although a successful SQL injection or data exposure attack is a huge problem in itself, if the attacker can recover passwords from the data, they can transparently compromise user accounts on a mass scale.

Securing Data in Application

Is My Application Vulnerable?

- If your application stores the salesforce.com user password, your application may be vulnerable.
- If your application collects other forms of sensitive data, your application may not be compliant with industry standards and the leakage of that sensitive data may cause a significant privacy incident with legal consequences.

How Can I Test My Application?

- Review the scheme used to store sensitive data and identify information collected in use cases and workflows.

How Do I Protect My Application?

Consider an application that must authenticate users.

- We have to store some form of the user's password in order to authenticate them.
- We don't want to store the password in plaintext form

Problem 1

- We could encrypt the passwords, but that would require an encryption key — and where would we store that?
- Developers have historically used a cryptographic hash function, a one-way function that is (supposedly) computationally infeasible to reverse. They then store the hash output:

```
hash = md5          # or SHA1, or Tiger, or SHA512, etc.  
storedPasswordHash = hash(password)  
authenticated? = hash(password) == storedPasswordHash
```

- The plaintext password is never stored.

Problem 2

- the attacker can easily pre-compute the hashes of a large password dictionary. Then the attacker matches their hashes to those in their stolen database.
- To address this problem, developers have historically “salted” the hash:

```
salt = generateRandomBytes(2)  
storedPasswordHash = salt + hash(salt + password)
```
- The goal is to make attackers have to compute a much larger dictionary of hashes
- The only obstacle here is the cost of the computing resources required to perform these calculations, and a single round of MD5 or SHA-1 is no longer expensive enough to slow attackers down.

Problem 3

- Fast, cheap and highly parallel computation on specialized hardware or commodity compute clusters makes brute force search with a dictionary quite affordable and accessible, even to adversaries with few resources.

- The canonical solution is bcrypt by Niels Provos and David Mazières. The idea is that we tune the hashing function to be pessimal; Provos and Mazières use a modified form of the Blowfish cipher to pessimize its already-slow setup time
- The benefit of this approach is that it slows down the attacker greatly, but for the application to verify a single password candidate still takes essentially no time.

Apex and Visualforce Applications

There are multiple ways to protect sensitive data, depending on the type of secret being stored, who should have access, and how the secret should be updated.

- **Protected Custom Metadata Types**
 - Within a namespaced managed package, protected custom metadata types are suitable for storing authentication data and other secrets
 - Custom metadata types can also be updated via the Metadata API in the organization that created the type, and can be read (but not updated) at runtime via SOQL code within an apex class in the same namespace as the metadata type.
- **Protected Custom Settings**
 - Setting the visibility of the Custom Setting Definition to “Protected” and including it in a managed package ensures that it’s only accessible programmatically via Apex code that exists within your package
 - Unlike custom metadata types, custom settings can be updated at runtime in your Apex class, but cannot be updated via the Metadata API.
 - The “transient” keyword should be used to declare instance variables within Visualforce controllers to ensure they are not transmitted as part of the view state.
- **Apex Crypto Functions**
 - The Apex crypto class provides algorithms for creating digests, MACs, signatures and AES encryption.
 - When using the crypto functions to implement AES encryption, keys must be generated randomly and stored securely in a Protected Custom Setting or Protected Custom Metadata type.
 - Never hardcode the key in within an Apex class.

Method	Supported Standards
Encrypt() EncryptWithManagedIv() Decrypt() DecryptWithManagedIv()	AES128, AES192, AES256 for encryption. PKCS#5 padding and Cipher Block Chaining.
generateDigest() generateMac()	MD5, SHA1, SHA256, SHA512
sign()	SHA1 with RSA

- **Encrypted Custom Fields**
 - Encrypted custom fields are text fields that can contain letters, numbers, or symbols but are encrypted with 128-bit keys and use the AES algorithm.
 - The value of an encrypted field is only visible to users that have the “View Encrypted Data” permission.
 - We do not recommend storing authentication data in encrypted custom fields, however these fields are suitable for storing other types of sensitive data (credit card information, social security numbers, etc).
- **Named Credentials**
 - Named Credentials are a safe and secure way of storing authentication data for external services called from your apex code such as authentication tokens.

- Be aware that users with customize application permission can view named credentials, so if your security policy requires that the secrets be hidden from subscribers, then please use a protected custom metadata type or protected custom setting.

General Guidance

When storing sensitive information on a machine:

- All authentication secrets must be encrypted when stored on disk.
- For client apps running on a desktop, laptop, tablet, or mobile device, store all secrets in the vendor provided key store (keychain in OS X/iOS devices, keystore in Android devices, or in the registry protected with the DP-API on windows devices.) This is a hard requirement to pass the security review.
- For services running on servers that must boot without user interaction, store secrets in a database encrypted with a key not available to the database process. The application layer should provide the key as needed to the database at runtime or should decrypt/encrypt as needed in its own process space.
- Do not store any cryptographic keys used for protecting secrets in your application code
- Salt hashes, and if possible store salts and hashes separately
- Leverage strong platform cryptographic solutions
- Check if frameworks/platforms have already addressed the problem
- Use SSL/TLS to transmit sensitive data

ASP.NET

- ASP.NET provides access to the Windows CryptoAPIs and Data Protection API (DPAPI).
- This is intended to be used for the storage of sensitive information like passwords and encryption keys if the DataProtectionPermission has been granted to the code.
- The strongest solution for ASP.NET would be to rely on a hardware solution for securely storing cryptographic keys, such as a cryptographic smartcard or Hardware Security Module (HSM), that is accessible by using the underlying Crypto API with a vendor supplied CryptoAPI Cryptographic Service Provider (CSP).

Java

- Java provides the KeyStore class for storing cryptographic keys. By default this uses a flat file on the server that is encrypted with a password. For this reason, an alternative Cryptographic Service Provider (CSP) is recommended.
- The strongest solution for Java would be to rely on a hardware solution for securely storing cryptographic keys, such as a cryptographic smartcard or Hardware Security Module (HSM), that is accessible by using the vendor's supplied CSP in that java.security configuration file
- When not using a CSP, if the product is a client application, you must use JAVA bindings to store the passphrase protecting the keystore in the vendor provided key store

PHP

- PHP does not provide cryptographically secure random number generators. Make sure to use /dev/urandom as the source for random numbers.
- Use the mcrypt library for cryptography operations. Salted hashes and salts could be subsequently stored in a database.
- A framework called phpass offers "OpenBSD-style Blowfish-based bcrypt" for PHP.
- For client apps, you must use native bindings to store user secrets in the vendor provided key store.

Ruby on Rails

- There is a copy of bcrypt specifically for Ruby called bcrypt-ruby.
- For client apps, you must use ruby bindings to store secrets in the vendor provided key store.

Python

- Use a module that interacts with the vendor provided keystores such as the python keyring module.

Flash/Air apps

- Use the Encrypted Local Store which contains bindings to use vendor provided keystores to store secrets.