Search documentation Search

Appian

Record Level Security for Entity Backed Records Best Practice

Appian 7.7

NOTE: This site does not include documentation on the latest release of Appian. Please upgrade to benefit from our newest features. For more information on the latest release of Appian, please see the Appian 7.8 documentation

The information on this page is provided by the Appian Center of Excellence

Toggle Na

This page is *not* supported by Appian Technical Support. For additional assistance, please contact your Appian Account Executive to engage with Appian Professional Services.

Background

Each **record type** has a rolemap specifying its Viewers, Auditors, Editors, and Administrators. Security at the **record instance** level, however, is determined based on the security of the underlying data source. For more information, see the Record Security Model documentation.

An entity-backed record is a record which is sourced from data in a relational database via an Appian Data Store. Each individual record represents a ro in the top-level data store entity. Appian allows designers to define which users can view a data store, but it does not allow designers to specify security rolemaps for specific rows within a data store entity. The purpose of this best practice document is to explain the best practice for enforcing record-leve security for entity-backed records.

Note: Record-level security for Process-backed records is managed via process instance security. Record-level security for service-backed records is enforced through the service. This document only applies to entity-backed records.

Explanation

Record Type Default Filters can be used to restrict the records that a user is allowed to access. If information about which users have permission to vie records is stored alongside the record fields in the database, a designer can create a default filter on the entity-backed record which checks that view permission information against a user's group membership to determine whether a user can see a given record or not.

What is a Record Type Default Filter?

A Record Type Default Filters is a configuration made to the Record Type. Only records that pass all default filters will appear in the record list view and have record dashboards that users can access. Records that do not satisfy all default filters are not shown in the record list view and cannot be loaded URL nor queried via `queryrecord()` . This is enforced at the server-level by Appian.

Note: this is very different from Facets which are filters that end users can apply to the record list to view a subset of the records.

Managing View Permission using Groups

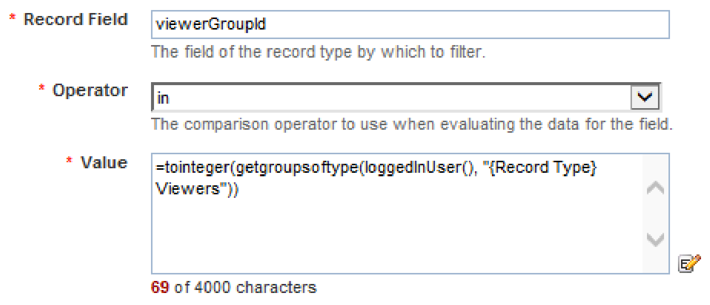
In order to store view permission information in the database for a given record type, the view permission for each record of that type must be manag via groups. By creating a group for each record that contains all the users who can view that record, the record type designer can store the group ID a record field in the database and leverage Appian's group management capabilities to ensure that only members of the group can access the record.

Implementing Record Level Security for Entity-Backed Records

Implement the record-level security default filter by doing the following:

1. Add a new `viewerGroupId` field to the CDT used by the record type data store entity.
2. Create a new group type called *"{Record Type} Viewers"* (replace {Record Type} with the name of the record type)
3. Update processes that create a new record to also create a new viewer group or use an existing viewer group and save that group ID into the `viewerGroupId` field in the CDT before saving the CDT to the database.
4. Update processes that modify a record to also update the membership of the viewer group as necessary.
5. Download and deploy the "People Functions" plugin from Forum which contains the `getgroupsoftypeforuser` function which returns the list of groups of a specific type of which the user is a member.
6. Update the Record Type configuration to add a new default filter as follows:
    - Record Field: `viewerGroupId`

- Operator: IN
- Value: tointeger(getgroupsoftypeforuser(loggedInUser(), "{Record Type} Viewers"))
  - The tointeger() function converts a list of groups to a list of group IDs. This is faster than using apply() and the group() function.
  - Remember to replace {Record Type} Viewers with the name of your group type.



* Record Field	viewerGroupId The field of the record type by which to filter.
* Operator	in The comparison operator to use when evaluating the data for the field.
* Value	=tointeger(getgroupsoftype(loggedInUser(), "{Record Type} Viewers")) 69 of 4000 characters

This default filter checks to see if the user is a member of the group specified by the viewerGroupId for a given record. In other words, the user will only be able to access records for which that user is a member of the viewer group specified by that record's viewerGroupId field.

Performance Tips

Performance of the IN operator in the default filter degrades as the user's group membership increases. In other words, the record list and record dashboard will load faster for a user who is a member of fewer {Record Type} Viewer groups than for a user who is a member of many {Record Type} Viewer groups. The exact performance difference will depend on your infrastructure, but it is strongly encouraged to conduct performance tests of you default filter if a user may be a member of more than 100 groups.

In order to reduce the number of groups of which a user is a member, you may decide you don't need a unique viewer group per record. Instead, you may create a pre-defined set of viewer groups that control security for a collection of records. For example, you may define view permission for records based on the Geographic Region or Department or Case Type of the record. In this case, every record in a specific Region/Department/Type/etc would have the same viewerGroupId values. Records in different Regions/Departments/Types/etc would have different viewerGroupId values.

Viewer Group Creation Tips

If dynamically creating a unique viewer group for each new record using proccess, use the following best practices: - Use the Create Custom Group Smart Service to create new viewer groups. - The new group must be of the "{Record Type} Viewer"s type created previously. - This group must contain all users and groups allowed to view the record. Use the Add Group Members Smart Service to add members to this group. - Remember to set the viewerGroupId field of the record CDT with the ID of the newly created viewer group before inserting or updating the record CDT into the database - Use the Add Group Members Smart Service and Remove Group Members Smart Service to orchestrate adding and removing users to the group base on your business logic.

If using a set of pre-defined viewer groups (e.g., per Department or per Region) instead of a unique viewer group per record, use the following best practices: - Store the set of pre-defined Groups in a constant or in the database - Use tointeger(ri!group) or group(ri!group, "id") to get the ID of a Group - Manage the membership of the groups separately from Record editing/updating.

Security Tips

Security applied using the Record Type Default Filter will be applied when interacting with records (including using queryrecord()), but it will not be enforced when using Query Rules or the queryentity() function which can access the same data via the data store entities.

When using Query Rules or queryentity() , designers must make sure they are not exposing to users any data from the database to which the user should not have access.

User Guides by Role

- Designer
- Developer
- Web Admin
- Server Admin (On-Premise Only)

Tutorials

- Records
- Interfaces
- Process

Release Information

- Release Notes
- Installation
- Migration
- System Requirements
- Hotfixes
- Release History

Other

- STAR Methodology
- Best Practices
- Glossary
- APIs