Field Audit Trail Implementation Guide

Salesforce, Spring '18

@salesforcedocs

Last updated: March 20, 2018

CONTENTS

FIELD AUDIT TRAIL

Field Audit Trail lets you define a policy to retain archived field history data up to 10 years, independent of field history tracking. This feature helps you comply with industry regulations related to audit capability and data retention.

Use Salesforce Metadata API to define a retention policy for your field history. Then use REST API, SOAP API, and Tooling API to work with your archived data. For information about enabling Field Audit Trail, contact your Salesforce representative.

Field history is copied from the History related list into the `FieldHistoryArchive` object and then deleted from the History related list. 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 object by using the Metadata API (Workbench or Ant Migration Tool). You can update the retention policy on an object as often as you like.

You can set field history retention policies on the following objects.

- Accounts
- Cases
- Contacts
- Leads
- Opportunities
- Assets
- Entitlements
- Service Contracts
- Contract Line Items
- Solutions
- Products
- Price Books
- Custom objects with field history tracking enabled

Note: `HistoryRetentionPolicy` is automatically set on the above 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. The default retention policy is not included when retrieving the object's definition through the Metadata API. Only custom retention policies are retrieved along with the object definition.

You can include field history retention policies in managed and unmanaged packages.

The following fields 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

EDITIONS

Available in: Salesforce Classic

Available in: **Enterprise**, **Performance**, and **Unlimited** Editions

USER PERMISSIONS

To specify a field history retention policy:

- Retain Field History

- Multi-select fields

After you define and deploy a Field Audit Trail policy, production data is migrated from related history lists such as Account History into the `FieldHistoryArchive` object. The first copy writes the field history that's defined by your policy to archive storage and sometimes takes a long time. Subsequent copies transfer only the changes since the last copy and are much faster. A bounded set of SOQL is available to query your archived data.

Note: If your organization has Field Audit Trail enabled, previously archived data isn't encrypted if you turn on Platform Encryption later. For example, your organization uses Field Audit Trail to define a data history retention policy for an account field, such as the phone number field. After enabling Platform Encryption, you turn on encryption for that field, and phone number data in the account is encrypted. New phone number records and previous updates stored in the Account History related list are encrypted. However, phone number history data that is already archived in the `FieldHistoryArchive` object continues to be stored without encryption. If your organization needs to encrypt previously archived data, contact Salesforce. We will encrypt and rearchive the stored field history data, then delete the unencrypted archive.

DELETE FIELD HISTORY AND FIELD AUDIT TRAIL DATA

Use Apex or SOAP to delete field history and field history archive data.

To delete field history and audit trail data, the user permissions Delete From Field History and Delete From Field History Archive must be enabled through a permission set or a user profile. The org preferences to enable these permissions, Delete From Field History and Delete From Field History Archive, are located in **Setup | User Interface**.

Delete field history data, such as AccountHistory, and field history archive data by passing in a list of ID values as strings using the Apex or SOAP `delete()` method. The Apex `delete()` method also works with a list of sObjects with the Id field populated.

The sObject acts like a template. All rows that match the sObject's fields and values are deleted. You can only specify fields that are part of the big object's index. You must specify all fields in the index. You can't include a partially specified index or non-indexed field, and wildcards are not supported.

Sample for deleting AccountHistory:

```
List<AccountHistory> ah = new List<sObject>();
ah.addAll( [ SELECT Id FROM AccountHistory
WHERE AccountId = '001d000000Ky3xIAB' and CreatedDate = YESTERDAY ] );
Database.delete(ah);
```

Samples for deleting from FieldHistoryArchive:

```
List<FieldHistoryArchive> fha = new List<sObject>();
fha.addAll([SELECT FieldHistoryType, ParentId, CreatedDate, HistoryId FROM
FieldHistoryArchive
WHERE FieldHistoryType = 'Account' AND ParentId = '001d000000Ky3xIAB' AND CreatedDate =
'2017-11-28T19:13:36.000z' AND HistoryId = '017D000000ESURXIA5']);
Database.delete(fha);
```

Alternatively, delete field history archive data with the new SOAP call `deleteByExample()`. Declare an sObject containing the fields and values in the `FieldHistoryArchive` big object to delete. This example deletes all rows that meet the specified criteria.

```
public static void main(String[] args) {
  try{
       //Create two sObjects to delete and place them in an array of sObjects to pass to
the delete method
       FieldHistoryArchive[] sObjectsToDelete = new FieldHistoryArchive[2];
       FieldHistoryArchive fha_1 = new FieldHistoryArchive();
       fha_1.setFieldHistoryType("Account");
       fha_1.setParentId("001d000000Ky3xIAB");
       Calendar dt = connection.getServerTimestamp().getTimestamp();
       dt.add(Calendar.DAY_OF_MONTH, -7);
       fha_1.setCreatedDate(dt);
       fha_1.setHistoryId("017D000000ESURXIA5");
       sObjectsToDelete[0] = fha_1;

       FieldHistoryArchive fha_2 = new FieldHistoryArchive();
       fha_2.setFieldHistoryType("Account");
       fha_2.setParentId("001d000000Ky3xIAB");
       fha_2.setCreatedDate(dt);
```

```
        fha_2.setHistoryId("017D000000ESURXIA5");
        sObjectsToDelete[1] = fha_2;
        DeleteByExampleResult[] result = connection.deleteByExample(sObjectsToDelete);
}   catch (ConnectionException ce) {
        ce.printStackTrace();
   }
}
```

Note: Repeating a successful `deleteByExample()` operation produces a success result, even if the rows have already been deleted.

EXAMPLES

Set Data Retention Policy for Field History

This example demonstrates how to set a field history data retention policy by using Metadata API. You need to edit the metadata only if you want to override the default policy values (18 months of production storage and 10 years of archive storage). Setting data retention policy involves creating a metadata package and deploying it. The package consists of a `.zip` file that contains an `objects` folder with the XML that defines each object's retention policy, and a project manifest that lists the objects and the API version to use.

Note: The first copy writes the entire field history that's defined by your policy to archive storage and might take a long time. Subsequent copies transfer only the changes since the last copy, and will be much faster.

1. Define a field history data retention policy for each object. The policy specifies the number of months that you want to maintain field history in Salesforce, and the number of years that you want to retain field history in the archive. The following sample file defines a policy of archiving the object after six months, and keeping the archives for five years.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
   <historyRetentionPolicy>
       <archiveAfterMonths>6</archiveAfterMonths>
       <archiveRetentionYears>5</archiveRetentionYears>
       <description>My field history retention</description>
   </historyRetentionPolicy>
   <fields>
       <fullName>AccountSource</fullName>
...
</CustomObject>
```

   The file name determines the object to which the policy is applied. For example, to apply the above policy to the Account object, save the file as `Account.object`. For existing custom objects, this works the same way, with the file named after the custom object. For example: `myObject__c.object`.

2. Create the project manifest, which is an XML file that's called `package.xml`. The following sample file lists several objects for which data retention policy is to be applied. With this manifest file, you expect the objects folder to contain five files: `Account.object`, `Case.object`, and so on.

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Account</members>
        <members>Case</members>
        <members>Contact</members>
        <members>Lead</members>
        <members>Opportunity</members>
    </types>
    <version>32.0</version>
</Package>
```

3. Create the `.zip` file and use the `deploy()` function to deploy your changes to your production environment. For more information, see the Metadata API Guide.

Note: This pilot doesn't support deployment from sandbox to production environments.

That's it! Your field history retention policy will go into effect according to the time periods that you set.

Create a Custom Object and Set Field History Retention Policy at the Same Time

You can use Metadata API to create a custom object and set retention policy at the same time. You must specify the minimum required fields when creating a new custom object. Here's sample XML that creates an object and sets field history retention policy:

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <deploymentStatus>Deployed</deploymentStatus>
    <enableHistory>true</enableHistory>
    <description>just a test object with one field for eclipse ide testing</description>
    <historyRetentionPolicy>
        <archiveAfterMonths>3</archiveAfterMonths>
        <archiveRetentionYears>10</archiveRetentionYears>
        <gracePeriodDays>1</gracePeriodDays>
        <description>Transaction Line History</description>
    </historyRetentionPolicy>
    <fields>
        <fullName>Comments__c</fullName>
        <description>add your comments about this object here</description>
       <inlineHelpText>This field contains comments made about this object</inlineHelpText>

        <label>Comments</label>
        <length>32000</length>
        <trackHistory>true</trackHistory>
        <type>LongTextArea</type>
        <visibleLines>30</visibleLines>
    </fields>
    <label>MyFirstObject</label>
    <nameField>
        <label>MyFirstObject Name</label>
        <type>Text</type>
    </nameField>
    <pluralLabel>MyFirstObjects</pluralLabel>
    <sharingModel>ReadWrite</sharingModel>
</CustomObject>
```

Set `trackHistory` to `true` on the fields that you want to track and `false` on the other fields.

Update Data Retention Policy for Field History

If a field history data retention policy is already defined on an object, you can update the policy by specifying a new value of `HistoryRetentionPolicy` in the metadata for that object. Once you deploy the metadata changes, the new policy overwrites the old one.

Note: To check the current data retention policy for any object, retrieve its metadata using Metadata API and look up the value of `HistoryRetentionPolicy`.

Query Archived Data

You can retrieve archived data by making SOQL queries on the `FieldHistoryArchive` object. You can filter on the `FieldHistoryType`, `ParentId`, and `CreatedDate` fields, as long as you specify them in that order. For example:

```
SELECT ParentId, FieldHistoryType, Field, Id, NewValue, OldValue FROM FieldHistoryArchive
 WHERE FieldHistoryType = 'Account' AND ParentId='906F000000
```

HistoryRetentionPolicy

Represents the policy for retaining field history data. By setting a policy, you can specify the number of months you want to maintain field history in Salesforce and the number of years that you want to retain field history in the archive.

This component is only available to users with the "RetainFieldHistory" permission.

Declarative Metadata File Suffix and Directory Location

Field history retention policies are defined as part of a standard or custom object. You can set field history retention policies for objects individually. See CustomObject for more information.

Version

Available in API version 31.0 and later.

Fields

Field Name	Field Type	Description
`archiveAfterMonths`	int	Required. The number of months that you want to keep field history data in Salesforce before archiving. You can set a minimum of 1 month and a maximum of 18 months. If you don't set a number, the default is 18 months. (That is, Salesforce maintains data for 18 months before archiving.)
`archiveRetentionYears`	int	Required. The number of years that you want to retain data in the archive. You can set a minimum of zero years, and a maximum of 10 years. If no number is set, the default is 10 years.
`description`	string	A text description for the history retention.
`gracePeriodDays`	int	The number of days of extra time after the `archiveAfterMonths` period before the data is archived. The `gracePeriodDays` interval applies only to the first time that the data is archived; because all the data is copied the first time, the operation may take longer than subsequent times when only the data that changed since the last archival operation is copied. The `gracePeriodDays` provides extra time for the administrator to prepare the organization before the initial archive operation. You can set a minimum of zero days and a maximum of 10 days. If no number is set, the default is 1 day.

Declarative Metadata Sample Definition

This sample shows the definition of a history retention policy for a custom object:

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
   <historyRetentionPolicy>
       <archiveAfterMonths>6</archiveAfterMonths>
       <archiveRetentionYears>5</archiveRetentionYears>
       <description>My field history retention</description>
   </historyRetentionPolicy>
   <fields>
       <fullName>AccountSource</fullName>
...
</CustomObject>
```

FieldHistoryArchive

Represents field history values for all objects that retain field history. `FieldHistoryArchive` is a big object, available only to users with the "Retain Field History" permission. This object is available in API version 29.0 and later.

Each instance of the `FieldHistoryArchive` object represents a single change in the value of a field. `FieldHistoryArchive` stores history for both standard and custom fields.

The `Field` field returns the name of the field unless the parent field or object is deleted, in which case it returns the field ID. You can use the ID to retrieve the old field and object name from the `FieldNameAfterArchival` and `ParentNameAfterArchival` fields, respectively.

Supported Calls

`describeSObjects()`,`query()`

Fields

Field Name	Details
`ArchiveFieldName`	**Type** string **Properties** Nillable **Description** The name of the field at the time the data was archived. If the field name changed, the name is sometimes not the same for all records related to a single field.
`ArchiveParentName`	**Type** string **Properties** Nillable **Description** The name of the parent object at the time the data was archived. If the object name changed, the name is sometimes not the same for all records related to a single field.
`ArchiveParentType`	**Type** string **Properties** Nillable

Field Name	Details
	Description The type of the field at the time the data was archived. If the field type changed, the type is sometimes not the same for all records related to a single field.
`ArchiveTimestamp`	**Type** dateTime **Properties** Nillable **Description** The date and time at which the data was archived.
`CreatedById`	**Type** reference **Properties** Nillable **Description** The user ID of the user who created the original record.
`CreatedDate`	**Type** dateTime **Properties** Nillable, Sort **Description** The date and time at which the original record was created.
`Field`	**Type** picklist **Properties** Filter, Nillable, Restricted picklist **Description** The name of the field that was changed. If the field is deleted from the parent object, the `Field` field contains the field ID instead.
`FieldHistoryType`	**Type** picklist **Properties** Nillable, Sort, Restricted picklist **Description** The name of the object that contains the field history (for example, Account).
`HistoryId`	**Type** reference

Field Name	Details
	Properties Filter, Sort **Description** The ID of the relevant history object (for example, AccountHistory). This field is available in versions 42.0 and later.
`Id`	**Type** ID **Properties** Defaulted on create, Filter, idLookup **Description** The ID of the archived record. It's useful to have a field's ID for fields that you've deleted. (Field names aren't retained in history when you delete fields from Salesforce.)
`NewValue`	**Type** anyType **Properties** Nillable **Description** The new value of the modified field.
`OldValue`	**Type** anyType **Properties** Nillable **Description** The previous value of the modified field.
`ParentId`	**Type** reference **Properties** Filter, Nillable, Sort **Description** The ID of the object that contains the field (the parent object).

Usage

When sorting fields, order them as follows:

1. `FieldHistoryType ASC`
2. `ParentID ASC`

3. `CreatedDate DESC`

HistoryRetentionJob

Represents the body of retained data from the archive, and the status of the archived data. Available in API version 29.0 or later.

Supported SOAP API Calls

`describeSObjects(),query()`

Supported REST API HTTP Methods

`GET`

Fields

Field Name	Details
`DurationSeconds`	**Type** int **Properties** Filter, Group, Nillable, Sort **Description** How many seconds the field history retention job took to complete (whether successful or not).
`HistoryType`	**Type** picklist **Properties** Create, Filter, Group, Nillable, Restricted picklist, Sort **Description** The object type that contains the field history that you retained. Valid values for standard objects are: • `Account` • `Case` • `Contact` • `Leads` • `Opportunity` For custom objects, use the object name.

Field Name	Details
`NumberOfRowsRetained`	**Type** int **Properties** Filter, Group, Nillable, Sort **Description** The number of field history rows that a field history retention job has retained.
`RetainOlderThanDate`	**Type** dateTime **Properties** Filter, Sort **Description** The date and time before which all field history data was retained.
`StartDate`	**Type** dateTime **Properties** Filter, Nillable, Sort **Description** The start date of the field history retention job.
`Status`	**Type** picklist **Properties** Filter, Group, Nillable, Restricted picklist, Sort **Description** Provides the status of the field history retention job. By default, the pilot feature copies data to the archive, leaving a duplicate of the archived data in Salesforce. Deletion of data from Salesforce after archiving is available upon request. Status can include: • `CopyScheduled` • `CopyRunning` • `CopySucceeded` • `CopyFailed` • `CopyKilled` • `NothingToArchive` • `DeleteScheduled` • `DeleteRunning` • `DeleteSucceeded` • `DeleteFailed`

Field Name	Details
	• `DeleteKilled`

SOQL with FieldHistoryArchive

You can use a subset of SOQL commands to query `FieldHistoryArchive`.

The allowed subset of SOQL commands lets you retrieve archived data for finer-grained processing. You can use the `WHERE` clause to filter the query by specifying comparison expressions for the `FieldHistoryType`, `ParentId`, and `CreatedDate` fields. You must specify the `FieldHistoryType` first, followed by either `ParentId` or `CreatedDate`. You can use =, <, >, <=, or >=, or `IN`. on the last field in your query. Any prior fields in your query can only use the = operator. The `!=`, `LIKE`, `NOT IN`, `EXCLUDES`, and `INCLUDES` operators are not valid in any query.

You can use the `LIMIT` clause to limit the number of returned results. If you don't use the `LIMIT` clause, a maximum of 2,000 results are returned. You can retrieve more batches of results by using `queryMore()`.

```
SELECT fieldList
FROM FieldHistoryArchive
[WHERE FieldHistoryType expression [AND ParentId expression[AND CreatedDate expression]]
 ]
[LIMIT rows]
```

Examples: Allowed Queries

Unfiltered

```
SELECT ParentId, FieldHistoryType, Field, Id, NewValue, OldValue
FROM FieldHistoryArchive
```

Filtered on FieldHistoryType

```
SELECT ParentId, FieldHistoryType, Field, Id, NewValue, OldValue
FROM FieldHistoryArchive
WHERE FieldHistoryType = 'Account'
```

Filtered on FieldHistoryType and ParentId

```
SELECT ParentId, FieldHistoryType, Field, Id, NewValue, OldValue
FROM FieldHistoryArchive
WHERE FieldHistoryType = 'Account' AND ParentId='906F00000008unAIAQ'
```

Filtered on FieldHistoryType, ParentId, and CreatedDate

```
SELECT ParentId, FieldHistoryType, Field, Id, NewValue, OldValue
FROM FieldHistoryArchive
WHERE FieldHistoryType = 'Account" AND ParentId='906F00000008unAIAQ' AND CreatedDate >
 LAST_MONTH
```

Filtered on FieldHistoryType and CreatedDate

```
SELECT ParentId, FieldHistoryType, Field, Id, NewValue, OldValue
FROM FieldHistoryArchive
WHERE FieldHistoryType = 'Account" AND CreatedDate >= LAST_MONTH
```

The following table describes the SOQL functions that are available for querying archived fields.

Note: All number fields that are returned from a SOQL query of archived objects are in standard notation, not scientific notation as in the number fields in the entity history of standard objects.

Table 1: SOQL Functions Available for Archived Fields

Functionality	Details
DATE LITERALS	*yesterday*, *last_week*, and so on
LIMIT	
WHERE	Filtering only on `FieldHistoryType`, `ParentId`, and `CreatedDate`

INDEX