



GE VERNOVA

Asset Performance Management

APM Classic

V4.6.11.0.0

Data Structure

Contents

Chapter 1: Overview	1
Overview of the Data Structure	2
Families	2
Family Hierarchy	2
Records	3
Fields	4
Relationship Definitions	5
Successors and Predecessors	6
Cardinality	6
Links	7
Rules	7
Data Spreading	8
Connections Between Families, Records, and Fields	11
Site Filtering	12

Copyright Digital, part of GE Vernova

© 2025 GE Vernova and/or its affiliates. All rights reserved.

GE, the GE Monogram, and Predix are trademarks of General Electric Company used under trademark license.

This document may contain Confidential/Proprietary information of GE Vernova and/or its affiliates. Distribution or reproduction is prohibited without permission.

THIS DOCUMENT AND ITS CONTENTS ARE PROVIDED "AS IS," WITH NO REPRESENTATION OR WARRANTIES OF ANY KIND, WHETHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO WARRANTIES OF DESIGN, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. ALL OTHER LIABILITY ARISING FROM RELIANCE UPON ANY INFORMATION CONTAINED HEREIN IS EXPRESSLY DISCLAIMED.

Access to and use of the software described in this document is conditioned on acceptance of the End User License Agreement and compliance with its terms.

Chapter 1

Overview

Topics:

- Overview of the Data Structure
- Families
- Family Hierarchy
- Records
- Fields
- Relationship Definitions
- Successors and Predecessors
- Cardinality
- Links
- Rules
- Data Spreading
- Connections Between Families, Records, and Fields
- Site Filtering

Overview of the Data Structure

Before you can begin working in APM, you must understand some fundamental concepts about how information is stored and organized in the APM database.

Families

Important: In most cases, throughout GE Vernova does not support any modifications made to database tables directly or via third-party tools. Do not write data directly into or change data in database tables through any third-party tool. Only access, add, or modify data related to GE Vernova products via approved GE Vernova procedures and processes. Accessing, adding, or modifying data by any other method may result in data corruption, and may void your GE Vernova product warranty. For more information, consult a member of the GE Vernova Professional Services department.

A family is an organizational unit that helps classify data in the database. APM uses two types of families:

- **Entity family:** An organizational unit that is used for classifying elements within your organization. Records belonging to entity families store information that is related to physical items (equipment, people, analyses, inspections, etc.) in your company or facility. For example, Full Inspection is an entity family that APM provides in the baseline database. An individual Full Inspection would be one entity belonging to that family.
- **Relationship family:** Corresponds to a database table that is used to connect two entity family tables. Relationship definitions can be defined for each relationship family to specify the entity families to which it relates. When two entity families have been related to one another through a relationship family, records within those entity families can be [linked](#), meaning that a connection can be created between the two records.

Each family has a corresponding table in the database. Family tables are used to store all the data belonging to a given family, and each row in a family table corresponds to a [record](#) in that family.

APM provides many families, both entity families and relationship families, in the baseline APM database. You can also create your own families. The power behind families lies in their flexibility. You can create as many families as you need to categorize all of your records. In addition, you can assign families whatever names are appropriate for classifying the type of data that they will store for your organization.

Family Hierarchy

Entity families can be organized into a hierarchy so that one family is defined as a subfamily of another family. By creating a hierarchy, you can enhance clarity.

When you look at the baseline APM database, you will see that some families are further divided into subfamilies. For example, the Recommendation family contains several subfamilies that define the types of Recommendation records that you can create.

As you customize your database, you will need to look at all of the different groupings of items that you will manage in APM. You need to divide these items and name each group, which will become your families. Next, you need to look for situations where one item is a

type of another item. In some cases, you may want to create a family that helps form your hierarchy but to which you would not directly save records.

Consider an example where you create three families: Failure, Equipment Failure, and Shutdown. These families are distinctly different, but they also share some commonality: a shutdown is a type of equipment failure, and equipment failure is a type of failure.

If you set up your hierarchy so that all of these families are stored at the root level, there would be no connection between them to organize the families logically. A better choice would be to set up families and subfamilies like this:

- Failure
 - Equipment Failure
 - Shutdown

In this example, the Failure family is the root family of the Equipment Failure family, and the Equipment Failure family is the root family of the Shutdown family.

Each root level will be useful for defining data that is common to its sublevels. In addition, the metadata is well ordered based on the type of data that is being collected.

Records

A record is simply a row in a database table. Because APM uses various types of database tables, in APM, there are several types of records, including:

- Entity family records (i.e., records associated with a specific entity family and stored in one or more entity family tables).
- Relationship family records (i.e., records stored in relationship family tables).

Entity family records are the most common type of record and typically represent a physical, functional, or organizational element in your company or facility. For example, an Equipment record would store the information associated with a physical piece of machinery.

Important: In most cases, throughout APM and documentation, when we use the term record alone, we are referring to a record belonging to an entity family.

Because APM is flexible in the type of information it can track, and because you have the ability to define how detailed your database will be, a record can represent almost anything. Some other examples of the types of records that you can set up include:

- Planned and unplanned events (e.g., maintenance requests and recommendations)
- Measurement readings
- Processes

Each record is assigned a unique value, or key, that identifies it within the system and differentiates it from all other records in the database. When you access a record, APM takes the record key, searches the database, and retrieves the data associated with that key. This data is then displayed through the interface.

How Are Records Uniquely Identified?

Each record in the APM database has a unique Entity Key that is stored in the ENTY_KEY field. The value in the ENTY_KEY field is unique to a given database. After you save a new record, it is assigned a unique Entity Key automatically.

You cannot display an Entity Key on a datasheet, but you can find it by running a query and including the ENTY_KEY field.

Note: All records also contain the system field CONTENT_GUID, which uniquely identifies a record across all databases. This field is populated automatically with a system-generated value when a record is initiated (i.e., when you initiate the record-creation process).

What is a Record ID?

Throughout the documentation, we refer to the value that exists in the ENTY_ID field as the Record ID. A Record ID can help you identify a record in the database. A Record ID is determined via the ID Template configuration. If an ID Template has not been configured for a family, records created in those families will not have a Record ID.

While Record IDs are not required to be unique, if you set up your system such that you configure ID fields for all families and you enforce a policy of supplying unique values in all ID fields, the Record IDs of those records will be unique.

Record IDs are often required in URLs, so keep in mind that you can locate an Record ID by running a query and displaying the ENTY_ID field.

Fields

Fields represent the individual pieces of information that will be stored for [records](#) belonging to a given [family](#). For example, records stored in the Full Inspection family will contain information related to full inspections. Therefore, the Full Inspection family contains fields that help define the inspection. Using [data spreading](#), fields that are common to all families within a branch of the hierarchy can be defined on a family and then spread down to subfamilies.

Each field that is defined for a family corresponds to a column in that family table. Any field that you define for a family can be displayed on a datasheet so that values can be displayed and collected. Additionally, fields can be used in queries, graphs, reports, and other utilities to help you retrieve and display specific information about records in the database.

Fields can store:

- Letters and numbers (e.g., Equipment or Equipment-123).
- Numbers only.
- Boolean values (i.e., true or false).
- Dates and times.

Note: All dates and times are stored in Coordinated Universal Time (UTC). They are displayed to each user according to the time zone that is defined in his or her Security User record. For example, if a time is stored as 6:00:00 P.M. UTC, if you are logged in as a Security User with the time zone (UTC - 05:00) Eastern Time (US & Canada), you will see the time as 1:00:00 P.M.

System Field Details

In addition to the fields that you define manually for each family, each record that you create will also contain system fields. System fields:

- Are identified by a field ID (i.e., you cannot define a field caption for them).
- Are managed automatically as you perform actions within APM.
- Cannot be modified or configured in the same way that family fields can be managed or configured (e.g., you cannot define custom rules for them).
- Do not appear in many of the places where family fields appear (e.g., on datasheets).
- Can be included in queries.

- Are referenced throughout APM, but in ways that most end users do not see (e.g., in rules).

The following system fields exist for every record.

Field ID	Description	Behavior/Usage
CONTENT_GUID	A global unique identifier that uniquely identifies a record across all databases.	This field is populated automatically with a system-generated value when a record is instantiated (i.e., when you initiate the record-creation process). If you include this field in a query, when you run the query, values will not be displayed in the CONTENT_GUID column.
ENTY_CAPTION_TX	A field that stores a copy of the Record ID.	This field is updated when the ENTY_ID field is updated.
ENTY_ID	The Record ID of a given record.	This value is controlled by the ID Template defined for a family. This field is populated automatically when a record is created and when one of the fields that are defined in the ID Template is modified.
ENTY_KEY	A value that uniquely identifies a record within a given database.	This field is populated automatically with a system-generated value the first time a record is saved.
FMLY_KEY	A value that identifies the family to which a record belongs.	This field is populated automatically when a record is first saved with the unique key value that identifies a family within a database.
LAST_UPBY_SEUS_KEY	A value that identifies the user who last updated the record.	This field is updated automatically when a record is modified.
LAST_UPDT_DT	A value that identifies the date that a record was last updated.	This field is updated automatically when a record is modified.
LOCK_SEQ_NBR	A value that is used internally by APM to indicate how many times a record has been modified.	This field is updated automatically when a record is modified.

Relationship Definitions

A relationship definition is, for a given [relationship family](#), a rule that specifies which two entity families are related, [which family is the predecessor](#), [which family is the successor](#), and the [cardinality](#) of each.

By setting up relationship definitions, you can define the various types and levels of connections that exist between entities. For example, a piece of equipment might be related to items such as:

- The location that contains it.

- The people who inspect and maintain it.
- The inspection and repair events performed on it.
- The workflows in which it participates.

Defining, tracking, and analyzing the relationships between entities is one of the most powerful tools in APM. After you have associated entity families with one another using relationship definitions, you can create [rules](#) that are based on those relationships. In this way, individual entities, each stored separately within the database can interact with, use information from, and even be updated based on changes made to other, related entities.

When deciding how many relationships to create, remember that too many relationships can be confusing to users and could create unnecessary system overhead. Too few relationships, however, will prevent you from defining them specifically enough to be useful to the end user.

Successors and Predecessors

Before you can create a relationship definition, you must first decide which two families will participate in the relationship. Next, you must decide which family will be the predecessor and which will be the successor in the relationship.

In many cases, the predecessor is the single part of the relationship equation and the successor is the multiple part. For example, if the relationship is Has Failure, and you determine that a piece of equipment, such as a pump, has a failure, then the pump experiencing the failure is the predecessor because there is one pump, whereas the failure is the successor because a pump might have many failures.

Another way to look at predecessors and successors is to consider that the predecessor comes before the successor. For example, using this guideline, a piece of equipment could be the predecessor and a repair could be the successor because the equipment was in place before it was repaired.

Note that these are only guidelines. APM will allow any type of setup that you feel best suits your needs.

Cardinality

Cardinality specifies, for a particular [relationship definition](#), how many links can be created between records of the [predecessor and successor families](#). Consider an example where the Has Maintenance relationship relates the Compressor family to the Work Order family. Within this relationship definition, the cardinality would specify how many Work Order records could be linked to a given Compressor record, and vice versa.

The following list provides descriptions of the cardinality rules that can be defined for successors and predecessors. For the purposes of our examples, assume that the Compressor family is the predecessor and the Work Order family is the successor in the Has Maintenance relationship.

- One and only one predecessor record can be linked to one and only one successor record. For example, one and only one Compressor record can be linked to one and only one Work Order record. Compressor A can be linked to Work Order 1, but Compressor A cannot be linked to any other Work Orders.
- One and only one predecessor record can be linked to many successor records. For example, one and only one Compressor record can be linked to many Work Order records. Compressor A can be linked to Work Order 1, Work Order 2, and Work Order 3, and so on.

- Many predecessor records can be linked to one and only one successor record. For example, many Compressor records can be linked to one and only one Work Order record. Compressor A and Compressor B could be linked to Work Order 1.
- Many predecessor records can be linked to many successor records. For example, many Compressor records can be linked to many Work Order records. Compressor A and Compressor B can be linked to Work Order 1, and Compressor A and Compressor B can be linked to Work Order 2.

Based on the cardinality defined between entity families through a given relationship family, you will be limited to the number of links that you can create between records. For example, if your system is configured such that a Work Order record can be linked to only one Pump record, you could create a link between Pump 101 and Work Order 1, but you could not link Work Order 1 to any other records. If you tried to do so, APM would generate an error. In this way, cardinality helps you maintain integrity in your database because it prevents you from creating relationships that should not exist.

A given family can also act as the predecessor in one relationship and the successor in another. For example, the Equipment family is the successor in the Functional Location Has Equipment relationship and the predecessor in the Equipment Has Equipment relationship.

Links

A [relationship family](#) corresponds to a database table that is used to relate two entity family tables. [Relationship definitions](#) specify which entity families can be connected through that relationship for a given relationship family. A link is a relationship family record that stores the information relating two entity family records.

Consider an example where the Has Maintenance relationship family relates the Compressor entity family to the Work Order entity family. If a Compressor record exists in the database to identify a specific piece of equipment and you create a Work Order record to store the information associated with the maintenance that was performed on that piece of equipment, you might want to relate the two entity family records to one another. You would do so by creating a link between the two entity family records.

When you link two entity family records together, APM creates a record in the appropriate relationship family table. The relationship family record, or link, contains:

- A key identifying the link itself.
- A value identifying the predecessor record.
- A value identifying the successor record.

The link may also contain other data if additional fields have been defined for the relationship family.

Rules

In APM, rules consist of code that determines how records in the APM database will behave under specific conditions. Rule code is written in Visual Basic.Net (VB.Net), a programming language that is compatible with the language in which APM is written. In this way, you can specify that when certain actions are taken in APM, certain rules should be executed.

The purpose of writing rules for a family is to control how records in that family will behave when you work with those records in APM.

Rules can be defined:

- At the family level to specify how records within that family should behave.
- At the field level to specify how fields within records in a given family should behave.

Data Spreading

Grouping families into hierarchies encourages the use of data spreading. Using data spreading, fields that are common to all families within a branch of the hierarchy can be defined on a family and then spread down to subfamilies.

Fields that are specific only to one subfamily can be defined directly at the sublevel. Any fields that are defined for a family will be available for display at any sublevel.

You should not overuse the data-spreading feature. The deeper a family is positioned in the hierarchy, the more time it takes when querying data from the family because the query must look at many family tables to gather all the information for generating the query results.

Example of Data Spreading

Consider a family hierarchy that looks like this:

- Failure
 - Equipment Failure
 - Shutdown

According to this hierarchy, you could add records for all of the shutdowns to the Shutdown family. You would need to record data for those shutdown records such as Failure ID, Failure Date, and Failure Comments. To provide data storage for this information, you could put all four data fields in the Shutdown family.

Now, consider a hierarchy that looks like this:

>Failure

- >Equipment Failure
 - >Leak
 - >Shutdown

For Leak records, you would need to record data such as Failure ID, Failure Date, Failure Comments, and Leak Substance. Note that three of these fields store the same type of data that you wanted to store for shutdowns. If the fields are all on the Shutdown family, you would have to recreate these fields for the Leak family. By using data spreading, however, you can create a streamlined database and avoid the duplication of effort.

Consider the following examples. Example A does not use data spreading. Example B uses data spreading.

Example A: No Data Spreading Applied

Family	Fields
Failure	None
Failure\Equipment Failure	None

Family	Fields
Failure\Equipment Failure \Shutdown	Failure ID Failure Date Failure Comments
Failure\Equipment Failure \Leak	Failure ID Failure Date Failure Comments Leak Substance

Example B: Data Spreading Applied

Family	Fields
Failure	Failure ID Failure Date Failure Comments
Failure\ Equipment Failure	None
Failure\Equipment Failure \Shutdown	None
Failure\Equipment Failure \Leak	Leak Substance

Example B, which uses data spreading, has only four fields as compared to the seven fields created for Example A. Also, Example B has no duplicate fields, whereas Example A has three duplicate fields: Failure ID, Failure Date, and Failure Comments.

Data spreading multiplied over hundreds of families can help save administrative time that is required to create and maintain the data fields, and it can help limit the size of the physical database.

Note: Only the field definitions are spread to lower levels. Values themselves do not spread.

About Data Spreading and Physical Record Storage

In a data model that does not use data spreading, each record is stored in one row of one table, where the columns of that row contain all the information for the record.

In a data model that uses data spreading, however, the values for a single record are actually spread across multiple rows in multiple tables. Consider the following example, where spreading has been used.

Family	Fields
Failure	Failure ID Failure Date Failure Comments
Failure\Equipment Failure	None
Failure\Equipment Failure\Shutdown	None
Failure\Equipment Failure\Leak	Leak Substance

In this example, when you create a Leak record and define values for the Leak Substance, Failure Comments, Failure Date, and Failure ID fields, the only value that is actually stored in the Leak table is Leak Substance. The other three values are stored in the family table where the fields are defined (i.e., where the physical columns exist): the Failure table.

Each of the tables will contain a physical row, or record. When you view a record belonging to the Leak family, you are actually viewing a composite record that consists of the record stored in the Leak table and the record stored in the Failure table.

Typically, it is not necessary to refer to each record individually as a distinct row in separate physical tables. Therefore, in cases where spreading has been used, we use the term record to refer to the composite record that has been created as a result of data spreading.

Main Types of Data Spreading

There are three main aspects of field spreading:

- **Metadata:** When a field is spread from one family to another, the metadata (i.e., Field Identification Properties) associated with the root family field is inherited by the subfamily field when the subfamily is created. In other words, the field properties at the target level match the field properties defined at the source level. Most of this information cannot be modified on the family to which the field has been spread. The Caption, Description, and Help Text can be modified.
- **Physical Tablespace:** Within the family hierarchy, a table will be created for each family, regardless of its position in the hierarchy. Similarly, a column will be created within a subfamily table for every field in that family, regardless of whether or not it has been defined directly within that family or inherited from a higher-level family. In other words, even fields that are created through spreading will have their own physical tablespace in each family in which they exist.
- **Rules:** For field-level rules, you have two options. Within the subfamily, you can choose to:
 - Inherit the rules that are defined for the source family. This is the default condition for spread fields and is indicated by the **Keep all properties of this field the same as they are in the Parent family** option. When you accept this default selection, a code item will not be created for the field in the subfamily. All rules will be defined in the source family code item and applied to the subfamily.
 - or-
 - Define rules directly within the subfamily by selecting the **Change the properties of this field for Child** option. When you select this option, a code item will be created for the field in the field in the subfamily. The rules for the subfamily field will be defined directly in the subfamily code item. Any rules that exist in the associated source family code item will not be applied to the subfamily.

Note: If a subfamily code item exists for a spread field and you change the rules setting back to inherit from the source family, the existing subfamily code item will not be

deleted or modified but will no longer be used. In this way, if you later decide to define rules at the subfamily level, any rules that previously existed will already be there for you to use or modify as needed.

There are two ways to spread fields:

- Within the properties of any family field, select the **Spread to subfamilies** check box. When you do so, all subfamilies created under that family will automatically inherit that field when the subfamily is created. Spread fields will not be inherited by subfamilies that existed before the **Spread to subfamilies** check box was selected.
- Within any subfamily, use the Field Chooser to select fields in higher-level families that you want to create at the subfamily level. This option is useful for spreading fields to subfamilies that existed before a spread field was created. and for spreading fields only to some subfamilies.

After a field has been spread, clearing the **Spread to subfamilies** check box will prevent that field from being spread to additional families but will not remove the field from families to which it has already been spread. The metadata, physical tablespace, and rules will continue to exist at the subfamily level until they are deleted manually. Un-spreading a field will, however, break the connection between the source field and the subfamily field, meaning that changes made to the source family field will not be applied at the subfamily level, even if the subfamily field originated through spreading.

Connections Between Families, Records, and Fields

The following explanation illustrates the connection between families, records, and fields. While the example is generic, the concept applies to APM families, family fields, and records.

Note: In this example, data spreading has not been used.

Suppose that employees will be stored in the Employee entity family. Various fields will be defined for the Employee family to create columns in the Employee table and store data for individual Employee records. The Employee family table might look something like this:

Family →	Employee			
Fields →	Employee ID	First Name	Last Name	Date of Birth
Records {	E-101	John	Smith	1/10/71
	E-97	Mary	Jones	5/25/65
	E-113	Bob	Wray	7/1/68
	E-119	Jane	Duncan	2/12/59

The following fields are defined for the Employee family:

- Employee ID
- First Name
- Last Name
- Date of Birth

Four records belong to the Employee family and are stored as rows in the Employee table. Each record has a value in each field. The first record in the table defines the information for John Smith, the second record defines the information for Mary Jones, and so on. Each record has an ID, which is unique to the record and can be used for searching.

Because each employee has a certain number of years of service to the employer, the Employee family could be related to the Service family through the Has Service relationship family. The Service family, then, could contain fields such as Start Date and End Date. The Service family table might look similar to the following table.

Family →		Service			
Fields →		Service ID	Employee ID	Start Date	End Date
Records {		S-101	E-101	6/7/1980	12/15/2001
		S-97	E-97	4/29/1980	1/1/1985
		S-113	E-113	4/1/1981	
		S-119	E-119	6/1/1981	7/15/1989

In the Service family example, you can see that there are four records, each assigned a specific Service ID. The Employee ID field indicates the employee with which the record is associated, the Start Date field defines the date on which the employee began service, and the End Date field indicates the date on which the employee ended his or her service. Because the End Date field is empty for the employee defined by the Employee ID E-113, we can infer that he or she is the only one who remains with the company out of the four employees defined in the table.

Site Filtering

Site Filtering helps users who are assigned to specific sites find relevant records more easily. Site Filtering uses a Site Reference Key on families that have site filtering enabled. Site Filtering is a useful tool for companies that have facilities at multiple sites because users are prevented from viewing records associated with sites to which they are not assigned.

Important: Site Filtering is not a security feature. While Site Filtering can limit the number of results shown, it should not be considered a measure to block data access to certain users. Site Filtering improves the performance of APM so that the data shown is limited to that which is most relevant and necessary for your needs.

The Site Filtering feature utilizes data stored in the Site Reference family (i.e., sites). Site Filtering filters data in APM according to the sites stored in Site Reference records. You can generally only see data assigned to the same sites to which you are assigned as a APM Security User.

A user who is assigned to multiple sites can choose the site to which a record should be assigned when creating records for families that are enabled for site filtering. Records of families that are not enabled for site filtering are considered global records, which can be seen by all APM users regardless of site assignments.

Key Terms

- **Global:** Global is not actually considered a site. The Global designation means that the record contains a null Site Reference Key. The data is not assigned to any particular site.
- **Global Data:** Global data is data to which site filtering does not apply. Global data can be seen by any user.

Note: Your individual license and family privileges still affect the data that you can see. For example, if you do not have View access to a family containing global records, then you will not be able to see those records.

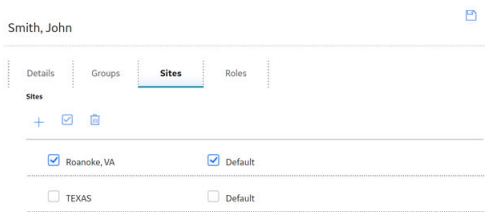
- **Global Record:** A record that contains global data, either because the specific record has been designated in the UI as Global, or because the family itself is not enabled for site filtering. In either case, the record has no site assigned to it (i.e., the Site Reference Key is null).
- **Super User:** A Super User can see all data. Super Users can see data for all sites as well as all global data.

Users and Site Filtering

- All users must be assigned a site.
- If you are a user who is assigned to only one site, you see all of the data associated with the site to which you are assigned, as well as any global records.
- Every user of APM has a Site Reference Key attribute. All users will be assigned at least one site and one default site. All users will also be able to access global data (i.e., data that does not have a specific Site Reference Key).

Note: Super Users are able to see data assigned to all sites as well as all global data.

- For example, when a user is added to the system in Security Manager, the user must be added to one or more sites, as shown in the following example.



Site Control

For databases in which there is only one site stored in the Site Reference family, the site control never appears, and records of families that are enabled for site filtering are assigned to that site.

When you have access to only one site, you can only see that site and the Global option when creating new records, as seen in this example in the Asset Criticality Analysis module.

Assessment **Analysis Definition** 0 Team Members 0 Reference Documents

Datasheet ID: ACA Analysis

ACA Analysis

Analysis

ID: A11-A11

Description: This belongs to ROA site

Scope: Text area

Owner: Text input Facilitator: Text input

Site: Roanoke, VA

Global

Roanoke, VA

When you have access to more than one site, but you do not have Super User privileges, you can see only the sites to which you are assigned as well as the **Global** option, as seen in this example in the Asset Criticality Analysis module.

Assessment **Analysis Definition** 0 Team Members 0 Reference Documents

Datasheet ID:

ACA Analysis

Analysis

ID

Description

Scope

Owner

Facilitator

Site:
 Roanoke, VA

- Global
- Perth, Australia
- Roanoke, VA

If you are a Super User, you can see all of the sites available, including the **Global** option, when creating new records, as seen in this example of an analysis in the Reliability Distribution Builder in the Reliability Analytics module.

Probability Distribution Builder

Define New Analysis

Analysis Name

Description

Site:
 Roanoke, VA

- Global
- Roanoke, VA

Example: Site Filtering in Asset Criticality Analysis

1. Log in as a user named Analyst, whose default site is Roanoke, VA.
2. Access ACA.
3. Create a new Asset Criticality Analysis.
 The new analysis is set to the default site for the Analyst.

Assessment **Analysis Definition** 0 Team Members 0 Reference Documents

Datasheet ID:
ACA Analysis

ACA Analysis

Analysis

ID
A11-A11

Description
This belongs to ROA site

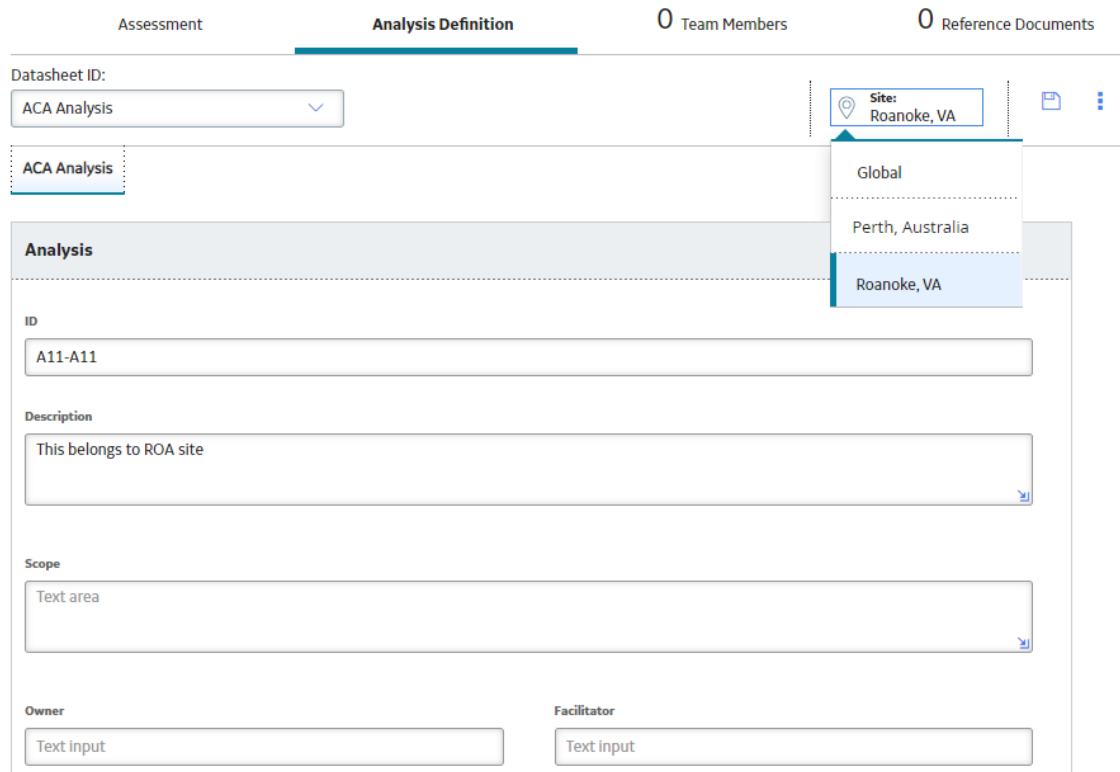
Scope
Text area

Owner
Text input

Facilitator
Text input

Site:
Roanoke, VA

- Global
- Perth, Australia
- Roanoke, VA



4. Keep the default site selection and save the analysis.

Note:

Unless you are a Super User, you cannot change the site after saving, as seen in the following image.

Datasheet ID:

Asset Criticality Analysis

In Progress
Not Assigned

Site:
Roanoke, VA



Main

Log

System ID:

~ NO.1 STEAM REHEATER ~ HXST 152 -ACA analysis-SYstem for ~ NO.1 STEAM REHEATER ~ F

System Description:

Text area

Site ID:

Text input

Unit ID:

Text input

System Comments:

Text area

The analysis is saved.

5. Create an ACA System.

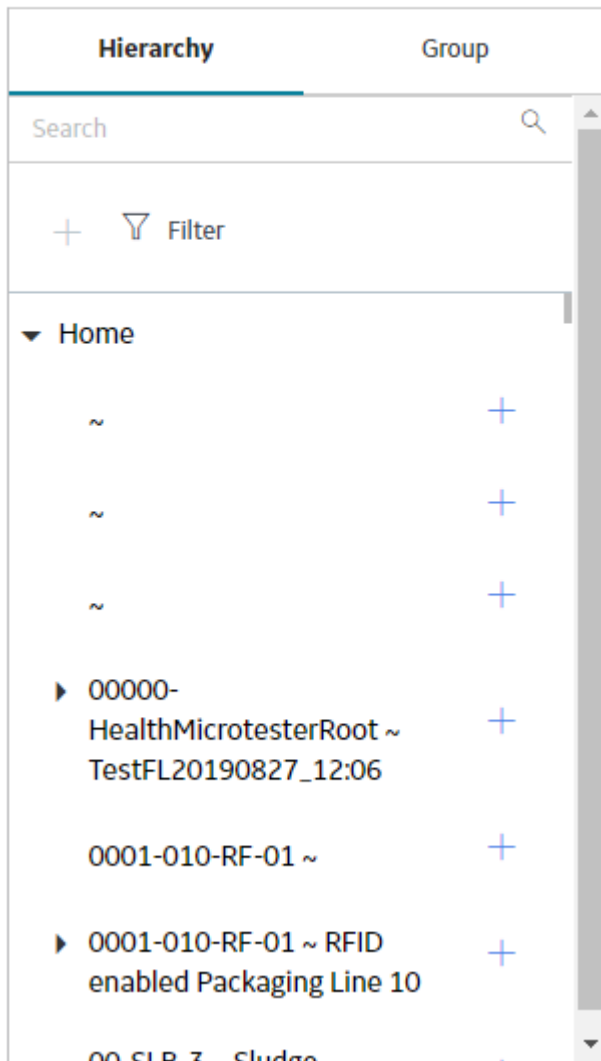
The new ACA system inherits the site from the analysis. The site can only be changed by a Super User.

Note: If a Super User changes the site reference on the ACA System to a reference that is different than the related Asset Criticality Analysis record, then users may see an incomplete analysis if they do not have the correct permissions.

6. Link Functional Locations or Equipment to the system.

You are only able to see Functional Locations and Equipment assigned to a site to which you are assigned.

Asset Finder



Site Filtering Configuration

If you would like additional information about Site Filtering, as well as instructions on how to configure it for your APM system, see the Sites section of the Administrative User Help.

Site Filtering Within Specific Modules

For additional information about how Site Filtering impacts specific modules, consult the Site Filtering topic in the documentation for that module, if applicable.