

Ab initio Session 12

Introduction to Ab Initio



Ab Initio Training

1



- **Sandboxes**
- **Concepts of Projects**
- **EME (Covering Checkin and Checkout procedures)**

Ab Initio Environment



- The Ab Initio Environment is a collection of functionality that allows you to set up a framework in which to organize your Ab Initio development, testing, and production work. It uses EME projects, common projects, and common data areas to build an open-ended hierarchy that can be as simple or complex as you require.
- Note that the term "Ab Initio Environment" can be a little ambiguous in certain contexts. It can be used to refer to an instance of an organizational framework that you have created, or to the set of Ab Initio executables that you use to create it. In these paragraphs, "Ab Initio Environment" signifies the things you create with the software: the collection of projects and sandboxes, the common data areas that are associated with them, and so on.

Main Characteristics of Ab Initio Environment




- Every sandbox (and project) in an Ab Initio Environment has by default an expanded set of subdirectories. In an Ab Initio Environment, you must create new projects either by running the **install-environment** script (when you create the new environment), or by running the **create-project** script.
- Every Ab Initio Environment is by default set up with ready-to-use data areas for serial and multiframe data, as well as logs, error reporting, and the like. Each of these areas is common to all projects in that environment, although each project has its own separate sub-section of it; these can be further subdivided into user-specific sub-areas.

What is a Sandbox?



- A "**sandbox**" is a special directory (folder) containing a certain minimum number of specific subdirectories for holding Ab Initio graphs and related files. These subdirectories have standard names that indicate their function. The sandbox directory itself can have any name; its sandbox properties are recorded in various special and hidden files that lie at its top level (the **my_project** directory, in the following illustration):



my_project

- bin ← you store miscellaneous executable files here
- db ← you store database configuration files here
- dml ← you store non-embedded record format files here
- mp ← you store graph files (.mp files) here
- run ← you store graphs deployed as scripts (.ksh files) here
- sql ← you store SQL files here
- xfr ← you store non-embedded transform files here

CapGemini Ab Initio Training 7

More on Subdirectories of Sandbox

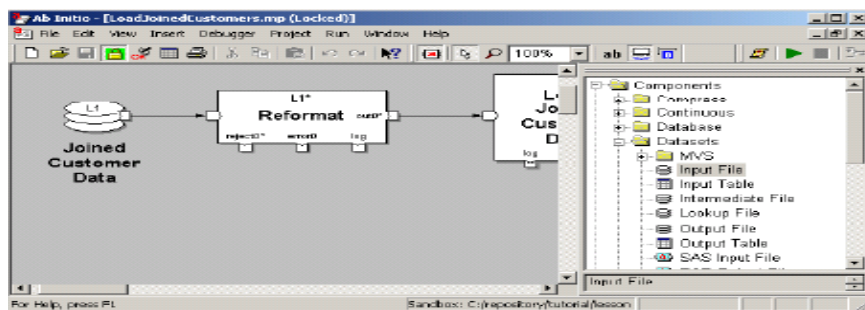


- **bin** — for executable files Usually this directory is used only in public projects, where it is used to hold macro scripts and third-party executables for Run Program components. A "public project" is an Ab Initio Environment concept: it is a common project, intended to be accessible to other projects. Public projects are the Ab Initio environment's mechanism for sharing data among projects.
- **sql** — for SQL files Used to hold **.sql** files used by database components, in particular by the Run SQL component.

Identifying the Sandbox in the GDE



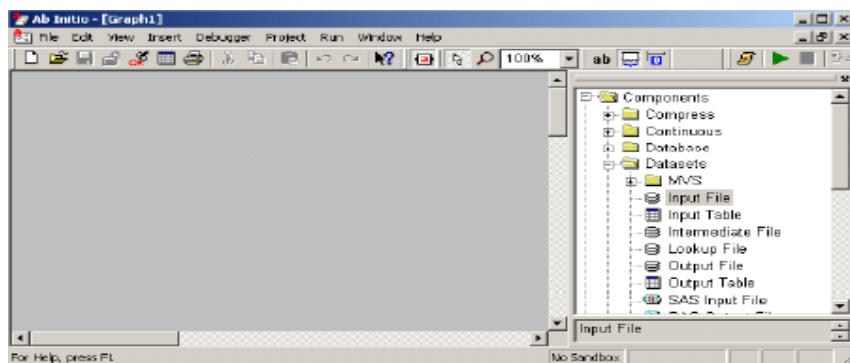
- The name of your sandbox is shown on the status bar at the bottom of the window in which the graph appears:



Identifying the Sandbox in the GDE



- Otherwise, if the graph is not there in the sandbox the status bar reads 'No Sandbox'



CapGemini

Ab Initio Training

10

Saving your Graphs in the Sandbox



- New graphs that have not yet been saved in your sandbox do not recognize the project parameters until after they have been saved.
- If you have to save the graph generated script produced when you ran the graph), check the **Save Script When Graph Saved To Sandbox** option in the EME Datastore Settings dialog box. The script will be saved whenever you save the graph in your sandbox.

What is a Project



- In EME, Project is a collection of graphs and related files that are stored in a single directory tree in EME data store and are treated as a group for purposes of version control, navigation and migration. Logically a project is a self contained and largely independent content area that accomplishes a simple business goal. **Projects are checked out to sandboxes.**

Projects



Every EME is grouped into projects.

- **Physically**, each project in the database is a collection of graphs and related files that are stored in the single directory tree and are treated as a group for purposes of source control, navigation and migration.
- **Logically** a project is fairly a self contained and largely independent content area, a suite of graphs and associated files that accomplishes a business goals.
- To work on a project you check it, out into a sandbox, which is a filesystem directory structure that mirrors the projects datastore structure.
- A project has attributes that describe its behavior ,including parameters ,import list, directory list and an extension list.
- Projects can include other projects so as to share common files and settings.

Sandbox with Project



- A sandbox that *is* associated with a project is by definition under source control in an Enterprise Meta>Environment datastore. The project is the "master copy" of the sandbox contents; you check out copies of files in the project to work on in your sandbox, and check in the altered files to update the project copies under source control. The directory structure of the project in the datastore is exactly the same as the sandbox directory structure in your filesystem.
- The sandbox now contains only some set of particular versions (usually the latest) of the files that make up its contents. The project contains all the versions of the files.

Sandboxes and Projects



- A project and a sandbox are thus essentially the same thing, seen from different points of view. A sandbox is a copy of a project. There can be (and usually are) many such copies. Multiple users can set up sandboxes based on the same project, thus creating multiple copies of it.
- **Each sandbox can be associated with only one project, but a project can be (and usually is) associated with multiple sandboxes.**
- Any user working on the project will have his or her own sandbox in which to work.
- **When you check a sandbox in to the EME, the contents of the sandbox are copied into the EME datastore project area. When you check the project out, the filesystem copies of the project contents are updated in your sandbox.**

Project Parameters



- Project Parameters, unlike graph Parameters, have an effect that is project wide. A project's parameters and values are inherited by all graphs in that project.
- Project Parameters are **checked in or out** with the project. You can lock them in your sandbox, they can be (and often are) called sandbox parameters, but they are properly called Project Parameters.

Order of evaluation of graph and project parameters

When you run the graph, parameters are evaluated in the following order:

- The host setup script is run
- Common (i.e. included) project parameters are evaluated.
- Project Parameters are evaluated
- The project start.ksh script is run
- Graph parameters are evaluated
- The graph start script is run.

Types of Project Parameters



There are 4 types of project parameters:

- Standard Parameters
- Switch Parameters
- Dependent Parameters
- Common Project Parameters

Project Directory Tree in a datastore



- The directory tree of the project contains subdirectories for the transforms, record formats ,graphs and so on. The project structure in the datastore is mirrored by the structure of your file system sandbox directory.
- The following table shows the structure of a typical EME datastore project.



/Projects	Default directory under which specific projects reside. You may edit the name of this directory.
<i>/yourproject</i>	Subdirectory for a specific project called <i>yourproject</i> .
/xfr	Subdirectory for transform files.
/dml	Subdirectory for record format files.
/db	Subdirectory for database interface files.
/mp	Subdirectory for GDE graphs.
/run	Subdirectory for deployed shell scripts.

More on Project



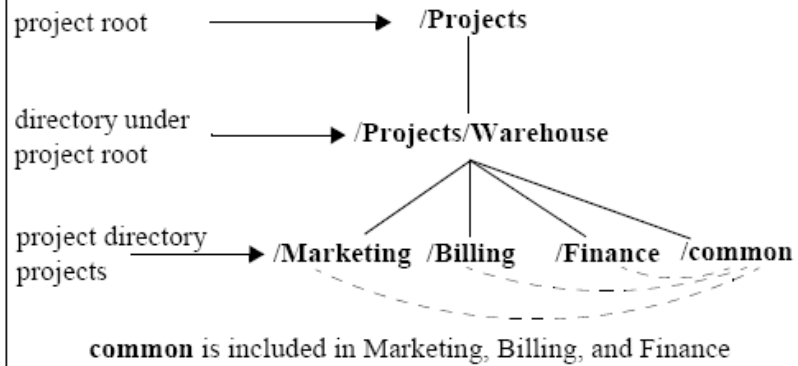
- **Migration:** A project can be parameterized so that you can migrate projects or individual graphs through different work stages from development through test, to production without having to make changes to the graph.
- **Source Control:** The EME assigns a version number to the project and all the files it contains .Because files are versioned to an earlier state or earlier time .
- **Common Projects:** A common project is a directory tree containing information that is shared among multiple projects. When information is shared among multiple Ab Initio applications projects then this information should be placed into a common project, which can be included in other projects.

Example of simple Project Organization



- Suppose you are trying to create a data warehouse which comprises three main projects-Marketing ,Billing , Finance. All of these projects make use of same data; therefore you include the shared data through the common project called common.
- Now if a single team at your company is responsible for the information in one of the constituent projects and responsibility for the common projects is shared by all, then a good way to organize our client projects and common project is shown by following diagram.

A good structure when one team owns one project



Key: The solid lines represent the directory structure. The dashed lines show how common projects are included.

EME



- The **Enterprise Meta>Environment (EME)** is a high-performance object-oriented storage system that inventories and manages various kinds of information associated with Ab Initio applications. It provides storage for all aspects of your data processing system, from design information to operations data.
- The EME also provides rich store for the applications themselves, including data formats and business rules. It acts as hub for data and definitions . Integrated metadata management provides the global and consolidated view of the structure and meaning of applications and data- information that is usually scattered throughout you business .

Benefits of EME



The Enterprise Meta>Environment provides a rich store for applications and all of their associated information including :

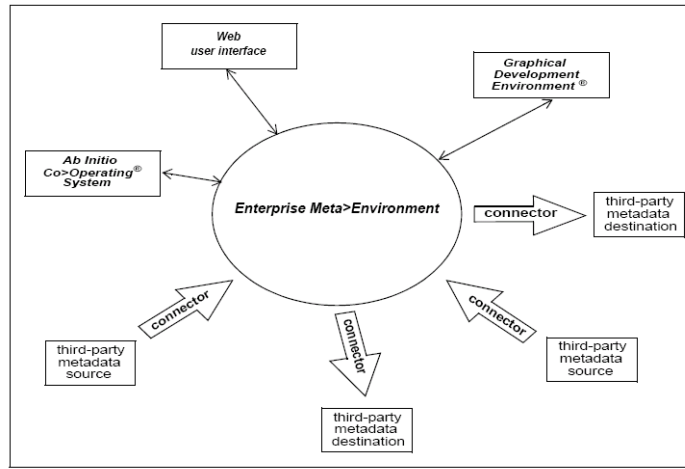
- **Technical Metadata-Applications** related business rules ,record formats and execution statistics
- **Business Metadata** -User defined documentations of job functions ,roles and responsibilities.

Metadata is data about data and is critical to understanding and driving your business process and computational resources .Storing and using metadata is as important to your business as storing and using data.

EME-Ab Initio Relevance



- By integrating technical and business metadata ,you can grasp the entirety of your data processing – from operational to analytical systems.
- The EME is completely integrated environment. The following figure shows how it fits in to the high level architecture of Ab Initio software.





Stepwise explanation of Ab Initio Architecture

- You construct your application from the building blocks called components, manipulating them through the Graphical Development Environment (GDE).
- You check in your applications to the EME.
- The EME and GDE uses the underlining functionality of the Co>Operating System to perform many of their tasks. The Cooperating System units the distributed resources into a single "virtual computer" to run applications in parallel.
- Ab Initio software runs on Unix ,Windows NT,MVS operating systems.



Stepwise explanation of Ab Initio Architecture - continued

- Ab Initio connector applications extract metadata from third party metadata sources into the EME or extract it from the EME into a third party destination.
- You view the results of project and application dependency analysis through a Web user interface .You also view and edit your business metadata through a web user interface.

EME :Various users constituency served



The EME addresses the metadata needs of three different constituencies:

- Business Users
- Developers
- System Administrators

EME :Various users constituency served



- **Business users** are interested in exploiting data for analysis, in particular with regard to databases ,tables and columns.
- **Developers** tend to be oriented towards applications ,needing to analyze the impact of potential program changes.
- **System Administrator** and **production personnel** want job status information and run statistics.

More on EME



- Application specific objects- like graphs, components, record formats and transformation are stored automatically whenever a user **checks-in** a project or graph. Job tracking information will also be stored automatically if a user selects Save Tracking Data to EME Datastore in the run settings dialog prior to running the application.

Working with EME



The basic situation when you work with EME is that you have files in two kinds of location:

- A personal work area which you specify, located in some filesystem.
- A filesystem storage area

This is the EME system area where every version that you save of the files you work on its permanently preserved, organized in projects. Its general name is [EME Datastore](#).

You have indirect access to this area through the GDE.



Features of EME-1

➤The EME stores all your metadata including:

Information Types	Description
Graphs	Ab Initio applications
Datasets	The logical inputs and outputs to your graphs (The physical data itself is stored in the host file system or in an RDBMS)
Record formats	Descriptions of how each field of your data is formatted and what it means
Data transformations	Rules for manipulating data during processing
Business metadata	Historical and high-level descriptions of data
Tracking information	Performance details of applications
Job history	Performance over time, for example, how fast each phase of an application runs
Web-based applications	Shop for Data graphs
Documentation	Supporting information
Categories	Classification information
Version history	Changes to EME objects

Features of EME-2



➤ Versioning:

The EME logs all changes made to the objects stored in it so that you can compare different versions of your graphs for revision history or access earlier versions of your applications. The version control system allows you to:

- Navigate to earlier version of the object
- Navigate it to the version of the graph used in the particular run of the job.
- View the state of the entire EME datastore at a particular point in time.
- Find the differences between the 2 versions of a graph, data transformation, or record format definition.

Features of EME-3



➤ **Source Code Control:**

The EME controls access to objects stored in it, preventing users from interfering with each other as they work on the project. EME source controls allows you to:

- Check in/check out files from a working area (sandbox)
- Apply tags to specific graph and file versions in the datastore
- Migrate versions from one datastore to another.

Features of EME-4



Dependency Analysis:

The EME performs dependency analysis, a process by which it determines the dependencies and relationships among objects stored in the EME.

With dependency analysis you can quickly:

- Find the origin of the data item by locating all the data transformations and data elements used in its production. These comprise the upstream dependencies of the object.
- Determine the possible impact of the program changes by locating the parts of an application that depend on the object being changed – the downstream dependencies of the object.



Features of EME-5

Performance Analysis:

- The EME stores extensive information Ab Initio environment, including which application ran, when they ran, and how long they took.

Universal Data Access through the Web Interface

- All the information in the EME is available at any desktop computer that connects to your internal web server.

EME Interfaces



You create and manage an EME through three interfaces:

Use the **GDE** (Graphical Development Environment) to do major tasks:

- Create an EME project.
- Edit a project's parameters and attributes
- Edit graphs, transformations and record formats
- Check out graphs, files and entire projects from the EME
- Check in to the EME modified graphs, files and projects
- Perform Dependency Analysis.

Out

Checking out Projects, Graphs and Files



In order to work on the project, you must :

- setup a sandbox
- Checking the project out into your sandbox and locking the files you wish to work on.

The main effect of checking an object out from the EME datastore is that your sandbox is updated with the latest datastore version of that object. In order to edit the checked-out object in your sandbox, you must first lock it. While you possess the object's lock, no one else can edit your sandbox's version of the object. When you check the object in the GDE release the lock. Or even you can release the lock explicitly yourself any time. Note that checking an object does not lock the object, it merely allows you to view the object in your sandbox.

Checking out Projects, Graphs and Files



- If the target sandbox does not exist, the Checkout wizard creates and configures the sandbox according to the project parameters, to the sandbox parameters.
- You can check out graphs and files as well as entire projects. When you check out a graph, the wizard also checks out the graphs project's parameters as well as the graphs associated record format ,transform ,and configuration files.
- You must first check out the common projects before you check out the projects that include them.

Locking



You can open any graph in the usual fashion (by selecting file>open). However

If the graph or file has been **checked in, then** it is a datastore object and you must first lock it before you can edit it.

Locking the graph prevents other users from changing it at the same time as you are working on it. Otherwise conflicts will occur between the sandbox and the datastore versions of the files.

The Lock Button



- The Lock button's appearance tells you that what the locking status of the object you are viewing is.
- When a graph or file is unlocked, the lock button on its tool bar appears as an open yellow lock symbol.



The Lock Button



- If an object has never been checked in, there is no lock associated with it. The lock button on the toolbar appears as an open gray lock symbol.
- When the object has been locked by other user then it looks like second diagram.



Checking in Projects, Graphs and Files

- Once you have successfully edited the project files you wish to work on you have to check them in order to put new versions under source control in the EME datastore ,and make them available to other users.



Thank You

End of Session 14

CapGemini

Ab Initio Training

69