Metreos Communications Environment Developer Training

August 2004

innovate integrate develop deploy™

Agenda

1. Overview
2. MCE Architecture
3. MCE Applications
4. Administration
5. Developing MCE Applications
6. Examples and Walkthroughs

METREOS COMMUNICATIONS

Training Objectives

- Educate you on the Metreos Communications Environment (MCE) architecture and operation.
- Train you on how to use the Metreos Visual Designer to build MCE applications.

Primary Goal:

Leave you with the necessary materials and understanding to effectively utilize your MCE installation to build and deploy innovative IP communications applications.

METREOS COMMUNICATIONS

Overview

History and the MCE

METREOS COMMUNICATIONS

History Repeats Itself

- IP telephony as a technology has matured but development and deployment tools have not.
- The first web applications were:
  - CGI-based (first generation technology)
  - Hard to build
  - Hard to manage
  - Hard to deploy
- Lack of a robust framework, toolset, and platform hinders adoption until…
  - Frameworks and toolsets like ASP, ColdFusion, etc.
  - Web application servers like WebLogic, WebSphere, etc.

METREOS COMMUNICATIONS

Metreos Communications Environment (MCE)

- Presents a new model for developing and deploying integrated, media-rich IP telephony applications.
- Eliminates the mystery of telephony and the underlying communications platform enabling developers to leverage convergence for innovation.
- Uniquely blends ease-of-use and power by allowing developers to quickly build applications without taking anything away from developer freedom.
- Delivers a secure, stable and scalable runtime platform.

The MCE allows you to focus on delivering value without the worrying about the details of how the telephony components work.

METREOS COMMUNICATIONS

MCE Architecture

System Components

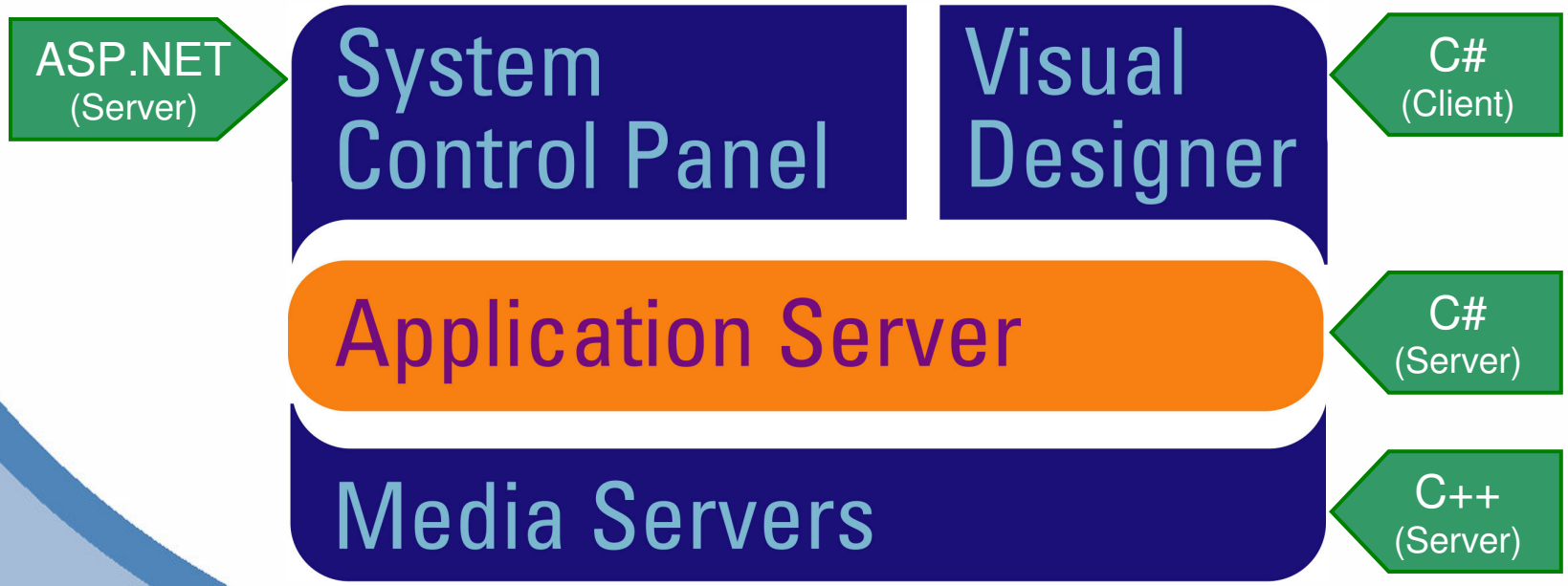
MCE Application Lifecycle

Language Architecture

METREOS COMMUNICATIONS

System Overview

- Application Server is the heart of the system.
- Media Servers provide audio stream control.
- Visual Designer enables developers to build applications.
- System Control Panel is the web administration portal.



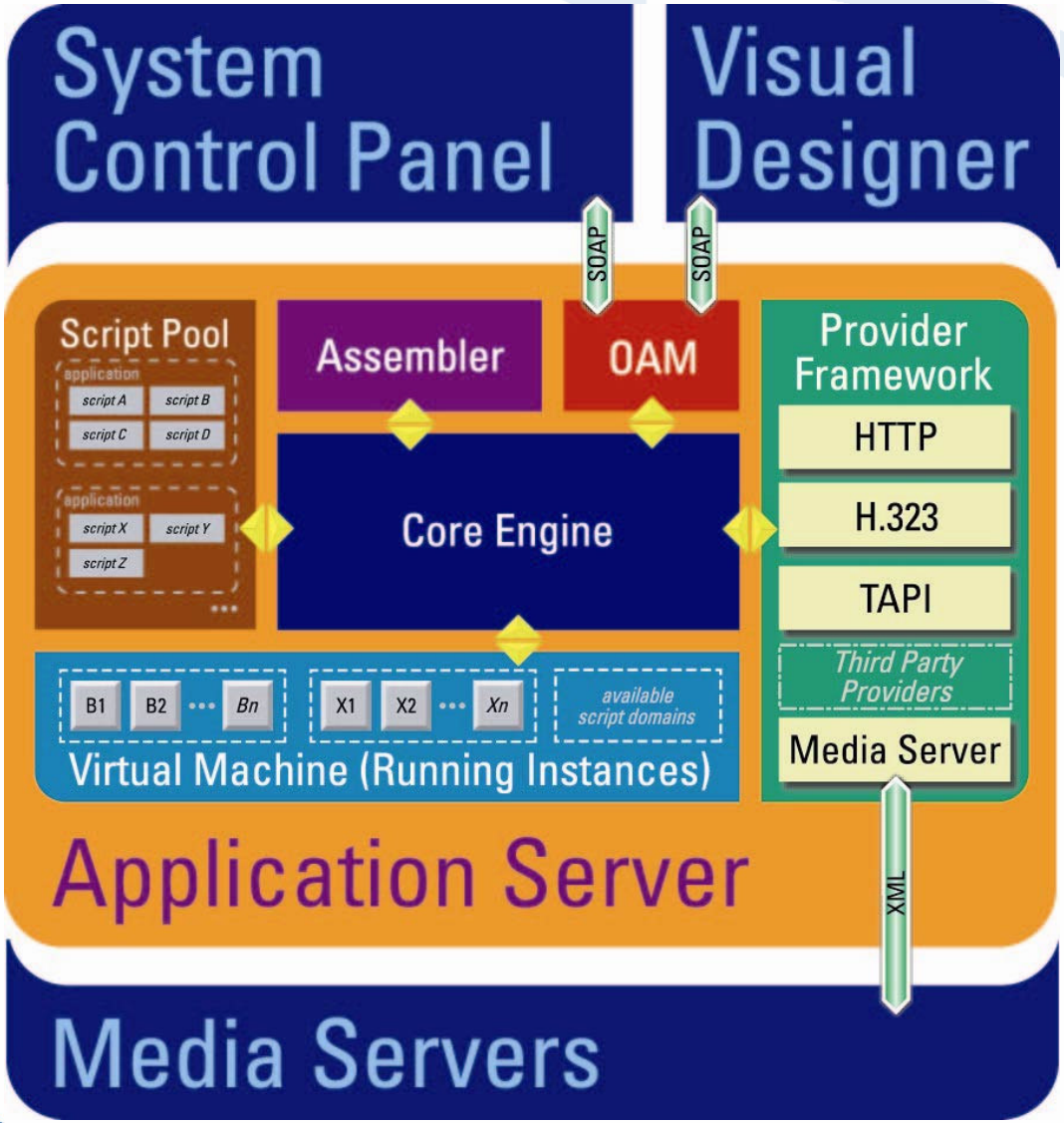
METREOS
COMMUNICATIONS

Metreos Application Server (MAS)

- Core of the MCE system.
- MCE applications are stored and execute within the application server.
- Applications scripts are assembled from an XML intermediate language into in-memory compiled applications ready for execution.
- A virtual machine inside the application server manages the execution of the application.
- Facilitates communication with external systems through "protocol providers".

METREOS COMMUNICATIONS

Metreos Application Server – How it Works



METREOS COMMUNICATIONS

MAS – Protocol Providers

- Provide the "glue" between the MAS and external systems.
- Functional equivalent of a Unix daemon process or Windows service.
- Execute within their own virtual process space.
- Execute in a dedicated thread and have their own message loop.
- Expose Events and Actions to MCE applications.
- Able to persist data and maintain state.

METREOS COMMUNICATIONS

Metreos Media Server (MMS)

- Pure software media processor.
- Originates and terminates voice media for applications running within an MCE.
- Tightly coupled with the Metreos Application Server via an XML RPC interface.
- Capable of processing up to 120 distinct media streams.
- Scalable independent of the application server.
- **All media processing capabilities exposed in an easy to use manner inside of Metreos Visual Designer.**

METREOS COMMUNICATIONS

MMS Feature Highlights

- **Scalability**
  - Up to 120 concurrent connections per box.
  - Scales independent of application server nodes.
- **IVR**
  - Play and Record in VOX and WAV format.
  - Play multiple prompts in sequence.
  - DTMF detection; in-band (including RFC2833) and out-of-band.
  - "Half-Connect" capable.
  - Complex "termination" conditions.
- **Conferencing**
  - Up to 120 participants per conference.
  - Individual conferee mute and kick.
  - Conference recording. Up to 64 concurrent conferences being recorded per box.
- **Advanced Features**
  - Multicast support for IVR.
  - Speech support using Nuance or Speechworks.
  - Low bit-rate coders (G.723 and G.729a).

METREOS COMMUNICATIONS

MCE System Control Panel

- Web-based management console.
- Consumes the application server's web services API.
- Provides access to all necessary administrative functions:
  - Add/Remove/Configure applications
  - Add/Remove/Configure protocol providers
  - Invoke protocol provider extensions
  - Add/Remove media servers
  - Configure system logging
  - License management
  - User management
  - Diagnostics

METREOS COMMUNICATIONS

Metreos Visual Designer

- Visually construct Communications Business Logic.
  - Extensive catalog of communications actions.
- Single tool for building complete IPT apps.
- Targeted at the network admin and software developer.
- Integrated with the Metreos Application Server for on-the-fly deployment.
- Comprehensive reference designs.
  - Accelerate developer learning curve.

METREOS COMMUNICATIONS

Visual Designer - Core Feature Set

- Graphical Application Definition
- Application Integrity Checks
- Extensible Toolbox
- Embedded Code
- One Click Deployment
- Runtime Debugging

METREOS COMMUNICATIONS

MCE Architecture

System Components

MCE Application Lifecycle

Language Architecture

METREOS COMMUNICATIONS

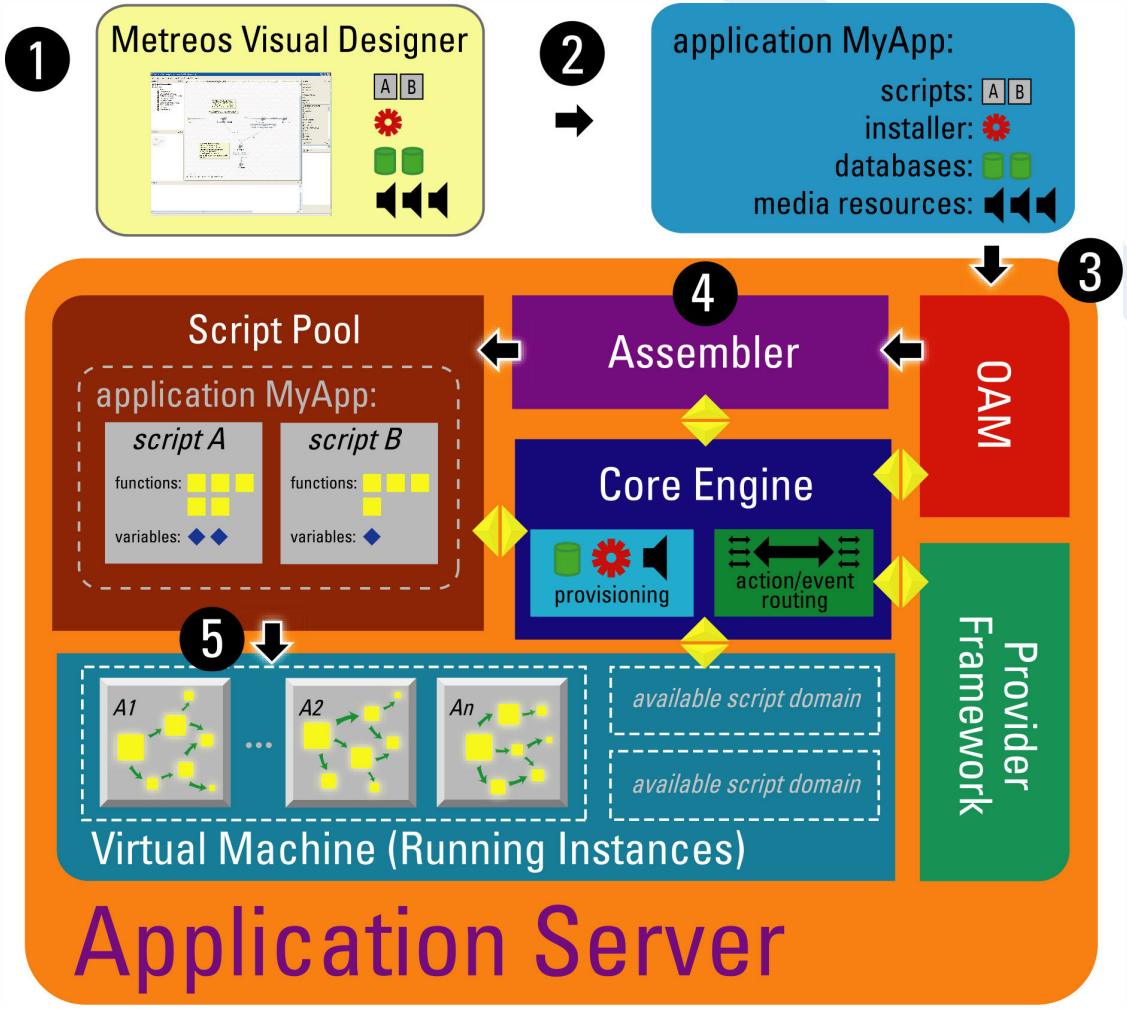
MCE Application Lifecycle

- Describes the flow of applications within the MCE from development to deployment.
- Five primary steps during which error checking is done.

METREOS COMMUNICATIONS

MCE Application Lifecycle (continued)

1. Development
2. Build
   - Compilation
   - Packaging
3. Deployment
4. Installation
   - Extraction
   - Provisioning
5. Execution



METREOS COMMUNICATIONS

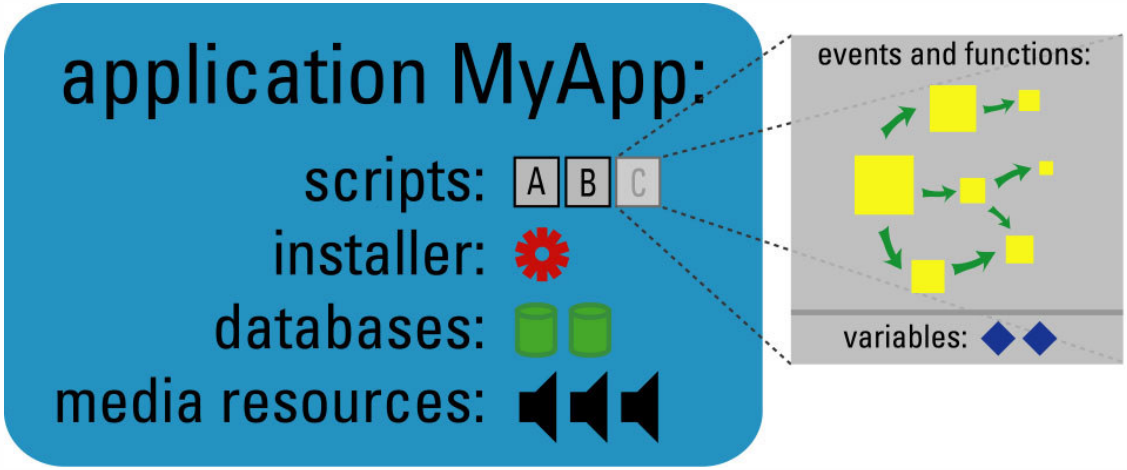
MCE Architecture

System Components
MCE Application Lifecycle
Language Architecture

METREOS
COMMUNICATIONS

Application Structure

- Four types of components:
  1. **Scripts** contain all application logic and define application flow.
  2. **Installer** defines the configuration required for the application.
  3. **Databases** are SQL creation scripts for database tables.
  4. **Media** resources are audio prompts utilized by the application.



METREOS COMMUNICATIONS

MCE Application Packages

- All elements of an MCE application required to deploy and run it are packaged into a single file.
- The file is known as a "Metreos Communications Archive" and has a .MCA file extension.
- The MCA package uses the TAR archive format and is the functional equivalent of a Java JAR file.
- The command line tool 'mca.exe' can be used to manually build and extract MCA archive packages.
- The Metreos Visual Designer handles the packaging automatically for the developer.

METREOS COMMUNICATIONS

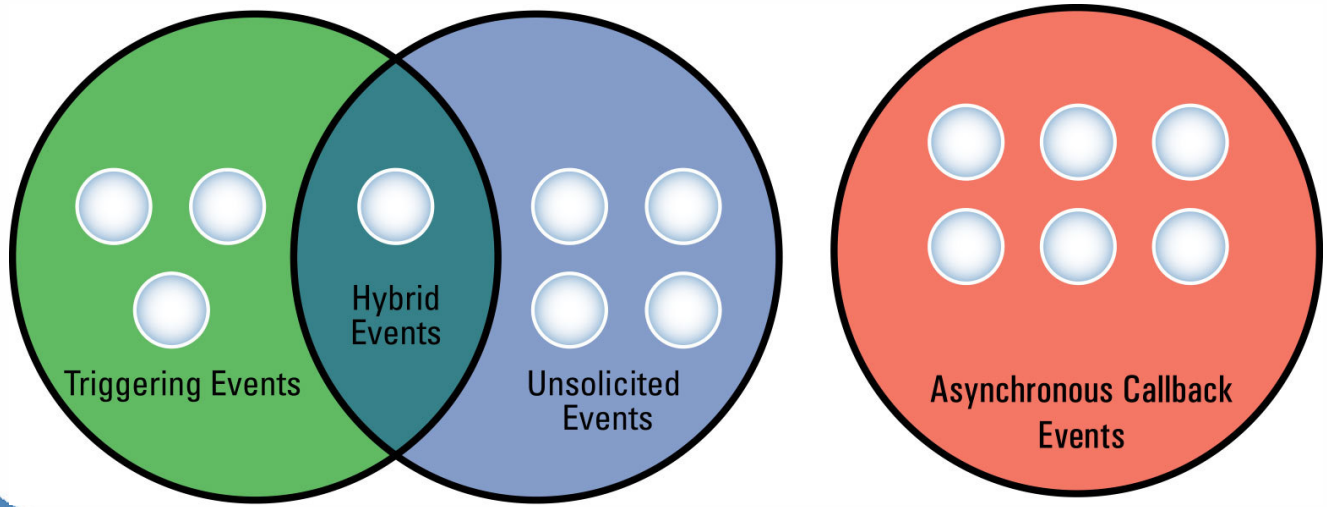
Action-Event Model

- MCE Applications are driven by a concept known as the **action-event** model.
- Applications communicate with the application server through actions.
- The application server sends information to the applications using events.

METREOS COMMUNICATIONS

Events

- Three types of events within the MCE:
  - Triggering
  - Unsolicited
  - Asynchronous Callback
- Hybrid events are those which may either be triggering or unsolicited (e.g., "Http.GotRequest").



METREOS COMMUNICATIONS

Events (continued)

- An event signature is the unique identifier of an event handler based on the event type and event parameters.
- All event handlers have event signatures.
  - Triggering event signatures indicate when an application script starts.
  - Unsolicited and asynchronous callback event signatures indicate which event handler function to use.
- Triggering event signatures must be unique across the entire application server.
- Unsolicited and asynchronous callback event signatures must be unique within a script.

METREOS COMMUNICATIONS

Actions

- MCE application scripts are constructed by linking actions together.
- Actions allow scripts to send data to the outside world or carry out specialized application logic.
- Three types of actions:
  - Core
  - Native
  - Provider
- Each type of action executes in a slightly different manner which will be covered later.

METREOS COMMUNICATIONS

MCE Applications

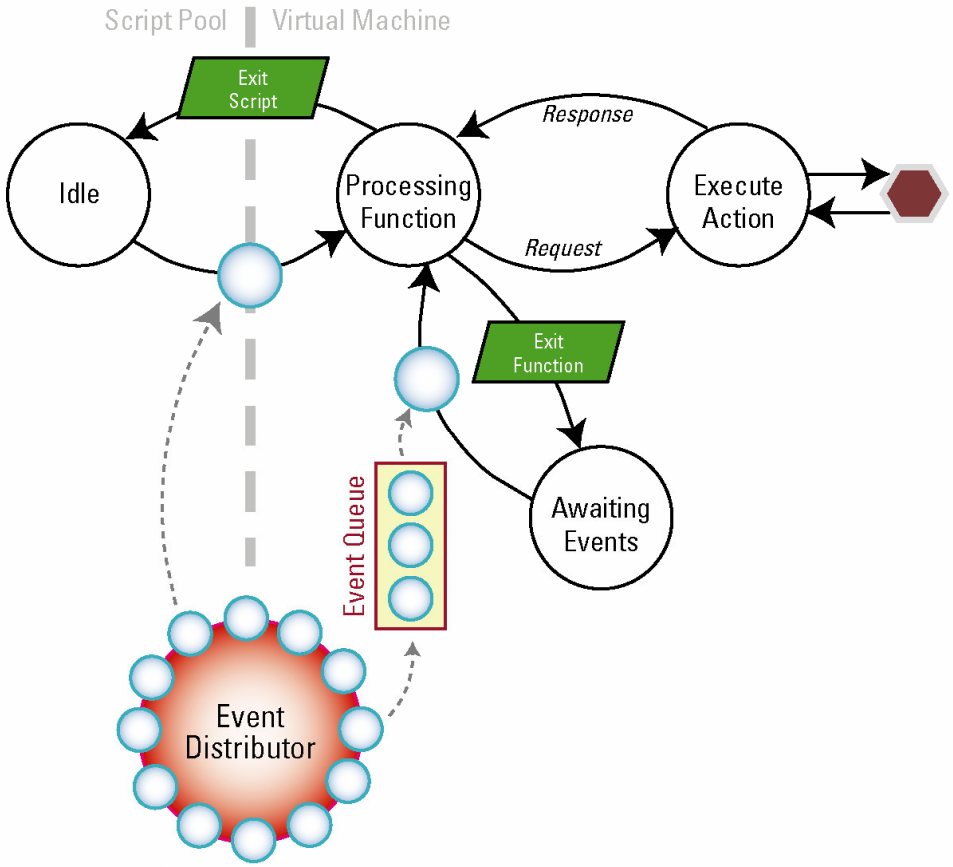
Execution Model

Application Script Elements

METREOS COMMUNICATIONS

Execution Model

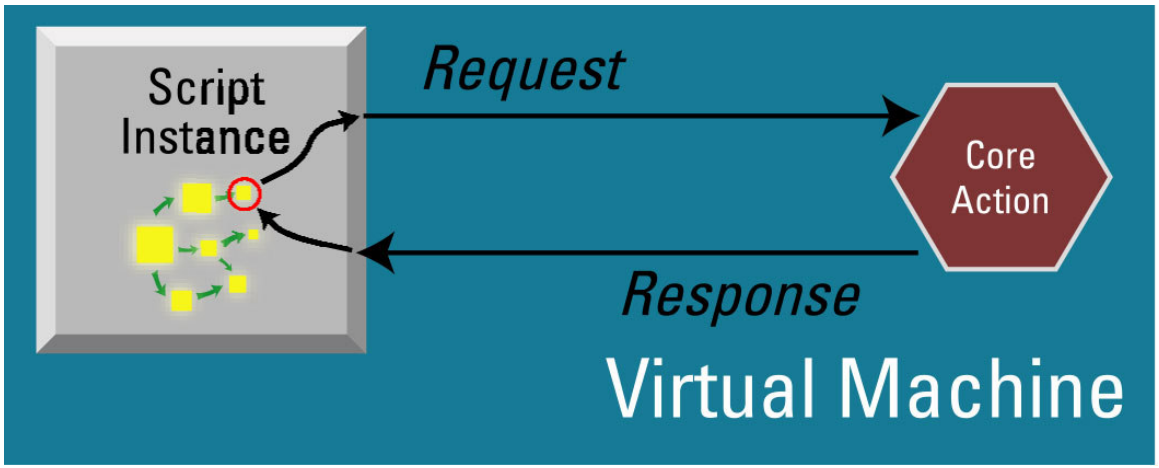
- Recall that the MCE utilizes an "action-event" execution model.
- Each type of action executes differently:
  - Core
  - Native
  - Provider



METREOS COMMUNICATIONS

Core Actions

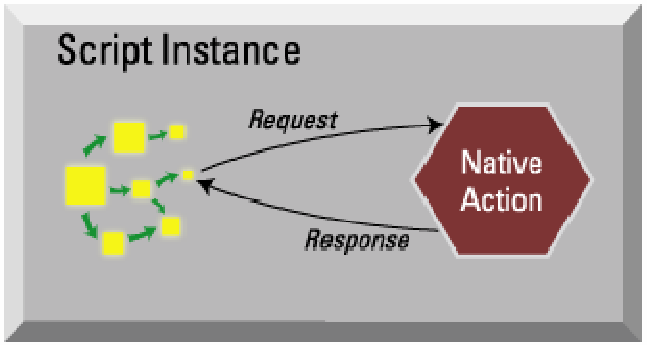
- Execute synchronously.
- Handled natively by the virtual machine.
- Represent fundamental functionality:
  - Exit, Exit Function, Call Function
  - Send Event, Forward All Events



METREOS COMMUNICATIONS

Native Actions

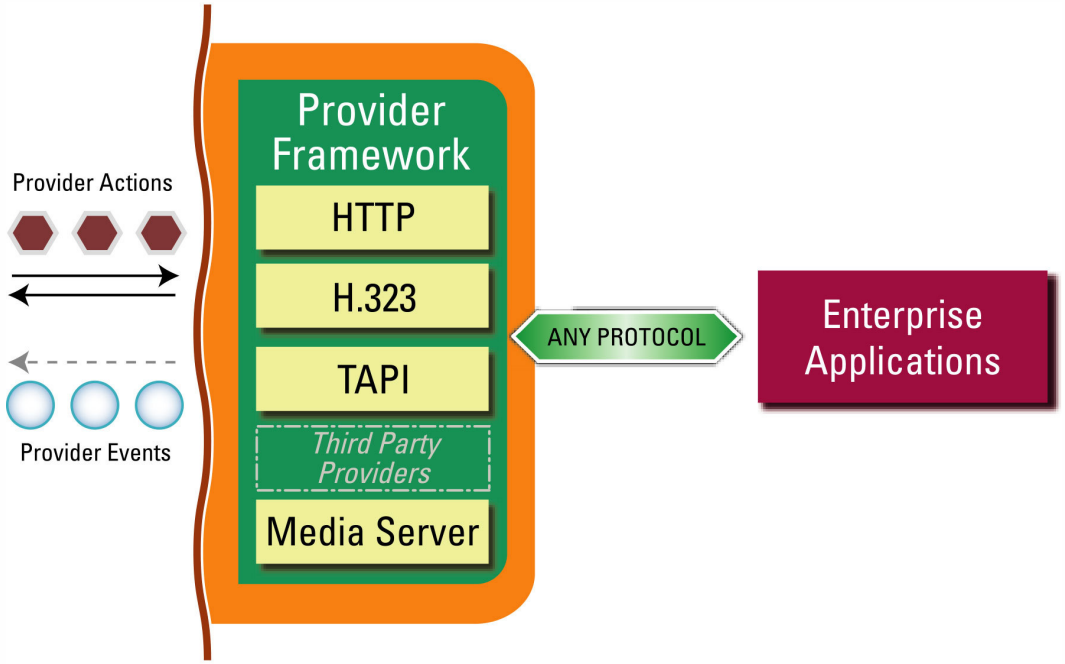
- Execute synchronously.
- Execution never leaves the script instance's context.
- Execution logic provided by external .NET assembly:
  1. Virtual machine recognizes native action.
  2. Virtual machine uses reflection and invokes the "Execute()" method of the native action.
  3. Execution leaves the virtual machine and executes code within the native action assembly.
  4. "Execute()" method returns and virtual machine continues.



METREOS COMMUNICATIONS

Provider Actions

- Recall that a "protocol provider" facilitates communication between external systems and MCE applications.
- Providers execute within their own virtual process space and are entirely separated from the virtual machine.
- Two types of provider actions:
  - Synchronous
  - Asynchronous

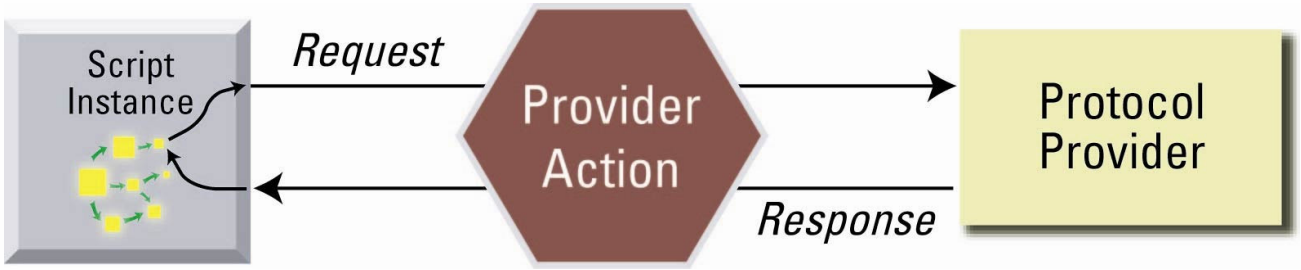


METREOS COMMUNICATIONS

Provider Actions – Synchronous

1. Virtual machine constructs a provider message.
2. Virtual machine sends provider message to core engine.
3. Core engine routes provider message to the right provider.
4. Provider sends a final response.

Application script blocks until a response is received.



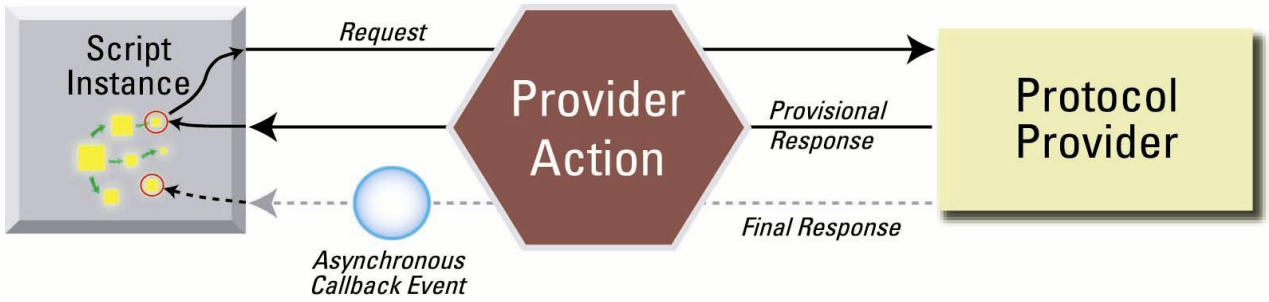
METREOS COMMUNICATIONS

Provider Actions – Asynchronous

1. Virtual machine constructs a provider message.
2. Virtual machine sends provider message to core engine.
3. Core engine routes provider message to the right provider.
4. Provider sends a provisional response.

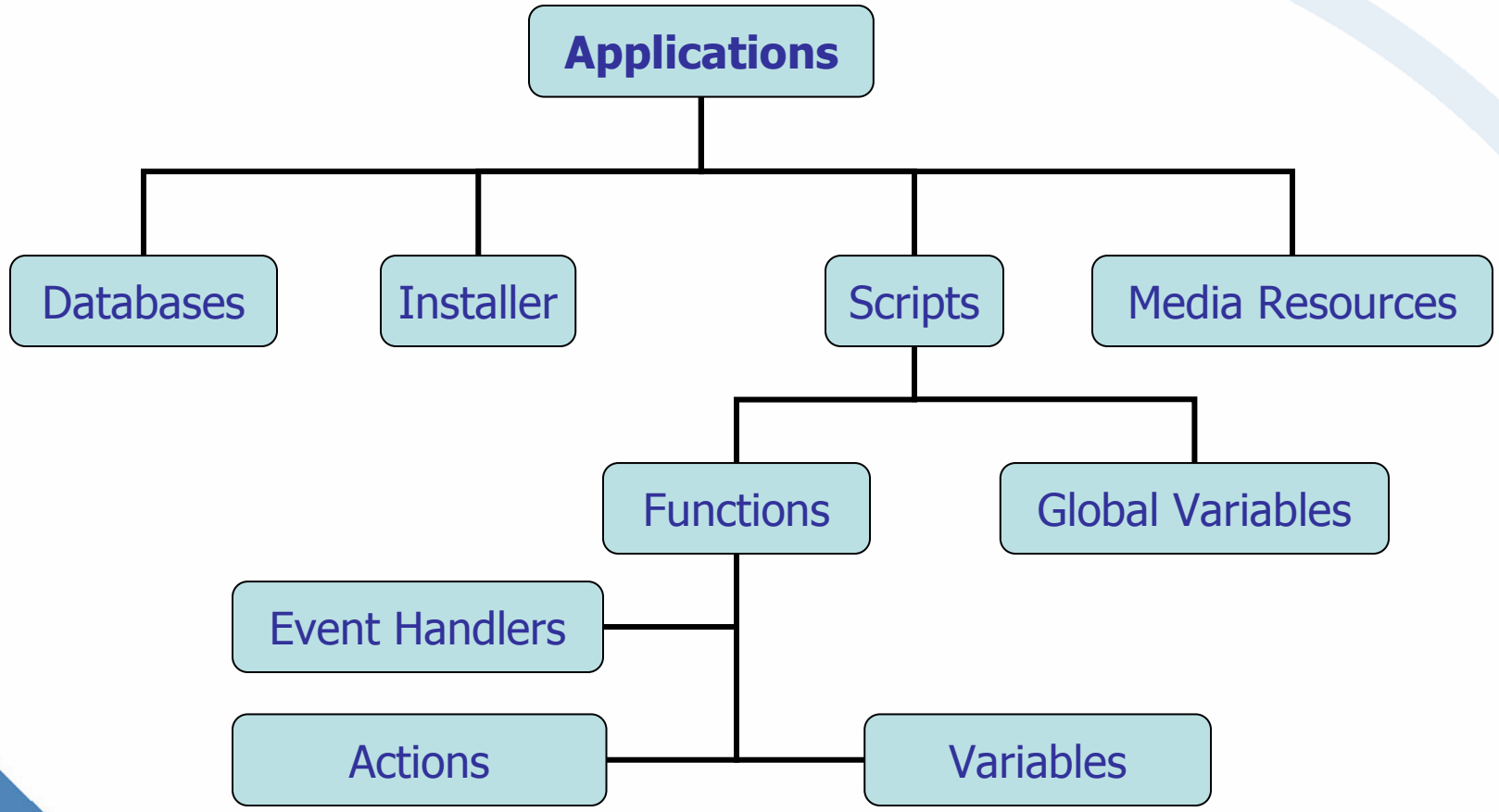
Application script blocks until a response is received.

5. Provider finishes processing the action request and sends a final response in the form of an asynchronous event callback.



METREOS COMMUNICATIONS

Application Elements



METREOS COMMUNICATIONS

Application Script Elements

- Application scripts contain these primary elements:
  - **Trigger**: Indicates when a new instance of an application script should begin.
  - **Functions**: Logical grouping of functionality defined using actions. Optionally, functions may be flagged as event handlers.
  - **Variables**: Encapsulate data within the script and may be scoped locally to the function or globally to the script.
  - **Actions**: Execute finite pieces of logic on behalf of the script (e.g., make a phone call).
- More details on all of these elements will be demonstrated using the Metreos Visual Designer.

METREOS COMMUNICATIONS

Administration

Using the System Control Panel

METREOS COMMUNICATIONS

Live Demonstration

METREOS COMMUNICATIONS

Developing MCE Applications

Using the Metreos Visual Designer

MCE Extensibility

Building Native Actions

Building Native Types

Building Protocol Providers

METREOS COMMUNICATIONS

Live Demonstration

METREOS
COMMUNICATIONS

Developing MCE Applications

Using the Metreos Visual Designer

MCE Extensibility

Building Native Actions

Building Native Types

Building Protocol Providers

METREOS COMMUNICATIONS

MCE Application Development

- MCE applications can be constructed entirely within the Metreos Visual Designer.
- If the developer wishes to extend the capabilities of the platform there are three options:
  - Native Actions
  - Native Types
  - Protocol Providers
- Extensions may be built using .NET enabled languages:
  - C#, C++, VB, Python, Java, etc.
- The choice of which type of extension to build is based on the general requirements of what that extension must provide.

METREOS COMMUNICATIONS

Native Actions vs. Protocol Providers

- Native actions:
  - Are light-weight and easy to build.
  - Execute within the same context as the script instance itself.
- Protocol Providers
  - Can generate events.
  - Can maintain state.
  - Can create threads.
  - Are more complex to build.

- **Developers should build a protocol provider when:**
  - They need to monitor external systems for events.
  - They need to maintain state independent of script instances.

METREOS COMMUNICATIONS

Native Types

- Allow developers to extend the type system provided by the MCE.
- Native types provide the functionality behind variables.
- A native type is responsible for serialization and de-serialization of a variable to and from string format.
- Native types are easy to build.

METREOS COMMUNICATIONS

Action/Event Package Definition

- Describe the inputs and outputs of native actions, types and providers.
- Meta-data is used by the Metreos Visual Designer in populating the toolbox.
- The action/event package meta-data is embedded into the native action, type or provider assembly using .NET attributes.

```
[Action("OpenDatabase", false, "Open Database", "Create a DB connection.")]
public string Execute(
    LogWriter log,
    SessionData sessionData,
    IConfigUtility configUtility)
{
    // Do Something Useful
}
```

A Standard Native Action Execute Method

METREOS COMMUNICATIONS

Adding MCE Extensions to the Designer

- Action/Event Package Generator tool, 'pgen.exe'.
- Examines Native Action, Native Type, or Provider assembly files for Action/Event package meta-data.
- Output is an XML file that can be loaded by the Metreos Visual Designer describing the capabilities of the extension.

- **Walkthrough:**

  Using pgen.exe from the command line.

METREOS COMMUNICATIONS

Developing MCE Applications

Using the Metreos Visual Designer
MCE Extensibility
Building Native Actions
Building Native Types
Building Protocol Providers

METREOS
COMMUNICATIONS

Implementing a Native Action

- Key namespaces:
  - **`Metreos.Interfaces`**
  - **`Metreos.ApplicationFramework`**
  - **`Metreos.PackageGeneratorCore.Attributes`**
- 5 Basic Steps:
  1. Create a new class that implements **`INativeAction`**
  2. Decorate the class with a **`PackageDeclAttribute`**
  3. Add **`set`** properties for any input parameters and decorate those properties with **`ActionParamFieldAttributes`**
  4. Override the **`Execute`** method with custom code
  5. Decorate the **`Execute`** method with an **`ActionAttribute`**

METREOS COMMUNICATIONS

Live Demonstration

METREOS COMMUNICATIONS

Developing MCE Applications

Using the Metreos Visual Designer
MCE Extensibility
Building Native Actions
Building Native Types
Building Protocol Providers

METREOS COMMUNICATIONS

Implementing a Native Type

- Key namespaces:
  - **`Metreos.ApplicationFramework`**
  - **`Metreos.PackageGeneratorCore.Attributes`**
- 4 Basic Steps:
  1. Create a new class that implements **`IVariable`**
  2. Decorate the class with a **`SerializableAttribute`**
  3. Implement the **`Parse`** and **`Reset`** methods
  4. Override the **`ToString`** method with custom code

METREOS COMMUNICATIONS

Live Demonstration

METREOS COMMUNICATIONS

Developing MCE Applications

Using the Metreos Visual Designer
MCE Extensibility
Building Native Actions
Building Native Types
Building Protocol Providers

METREOS
COMMUNICATIONS

Implementing a Protocol Provider

- Key namespaces:
  - `Metreos.Interfaces`
  - `Metreos.Messaging`
  - `Metreos.LoggingFramework`
  - `Metreos.ProviderFramework`
  - `Metreos.PackageGeneratorCore.Attributes`
- Implementation is more complex than either a Native Action or Type, but follows a very straight forward pattern.

METREOS COMMUNICATIONS

Implementing a Protocol Provider (cont)

1. Create a new class that derives from `ProviderBase`
2. Decorate the class with a `ProviderDeclAttribute` and `PackageDeclAttribute`
3. Override the `Initialize`, `RefreshConfiguration`, `OnStartup`, `OnShutdown` and `Cleanup` methods with custom code
4. Implement action handling methods and decorate those methods with `ActionAttribute`, `ActionParamAttributes`, and `ResultDataAttributes`

METREOS COMMUNICATIONS

Live Demonstration

METREOS COMMUNICATIONS

Examples and Walkthroughs

METREOS
COMMUNICATIONS

Live Demonstration

METREOS COMMUNICATIONS

Metreos Communications Environment Administrator and Developer Training

August 2004

Thank You!

METREOS
COMMUNICATIONS

innovate integrate develop deploy™