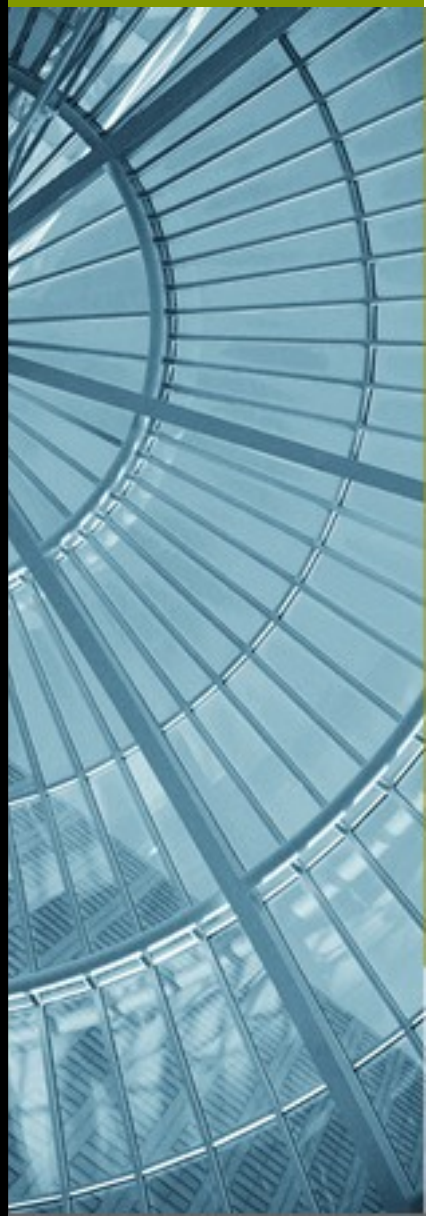
SYBASE®



STP Deployment Framework

Rob Cernich – April 6, 2006

Introduction

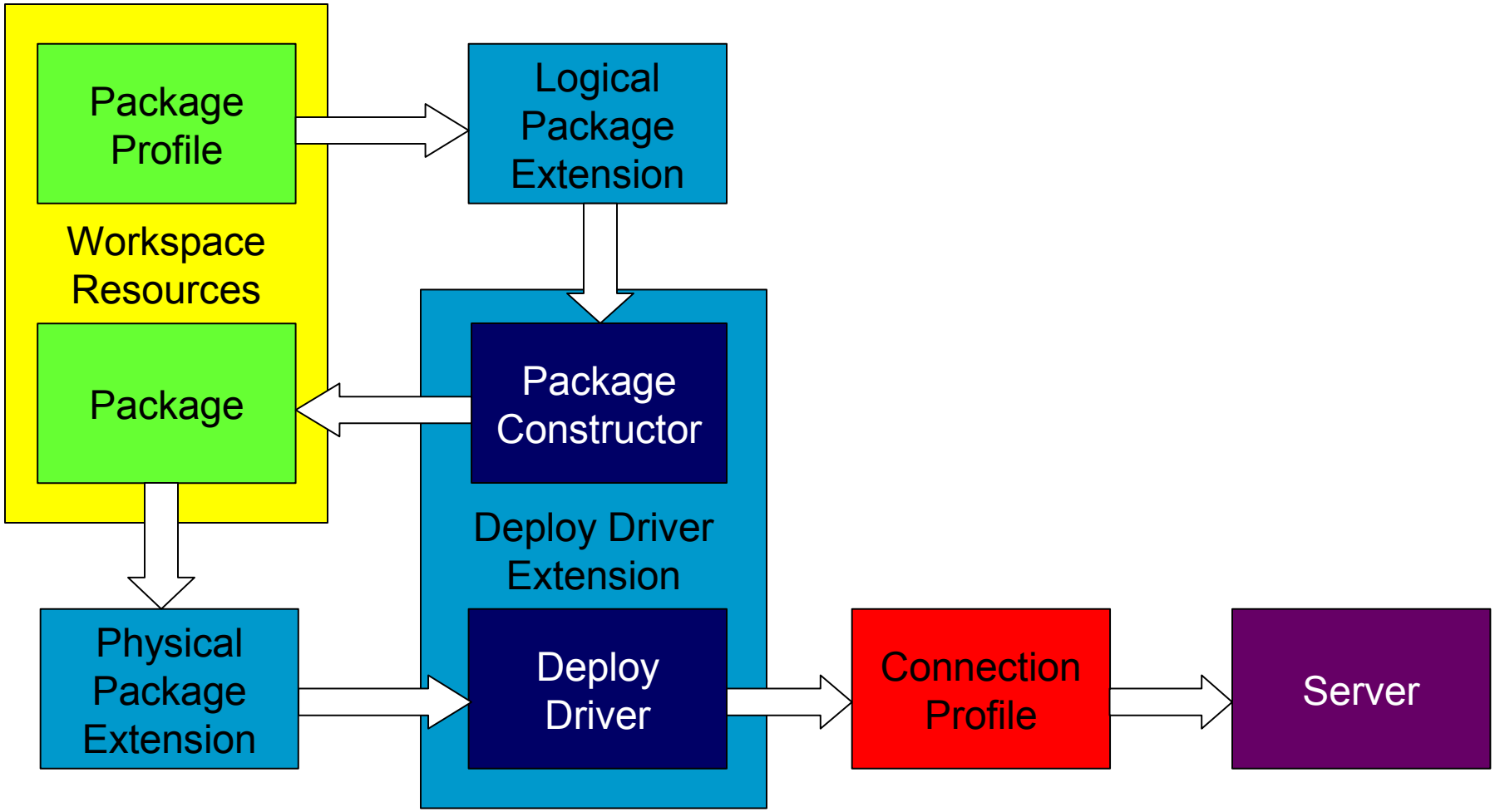
- This framework defines a set of API's and extension points for the definition of profiles that aid in the building, packaging, configuration and deployment of artifacts to runtime environments.

- This framework builds on top of the DTP connection profile framework that defines connection definitions for server environments.

Introduction

- **Deployment Framework**
  - Developers define extensions for identifying and working with packages within the workspace
  - Developers register deployment extensions against connection profiles
  - Deployment extensions may define package constructors for creating deployable packages from logical or abstract package definitions
- **Deployment Profile Editor**
  - Allows users to target packages for deployment to specific servers
  - Allows users to tailor package configuration for deployment to specific servers
  - Provides a mechanism for creating repeatable deployments

Overview



Overview (cont'd)

- User creates a package profile (a file describing the contents, etc. for a package)
- A logical package extension identifies the package profile to the framework. This information is used to help locate a package constructor.
- A package constructor is used to create a deployable package
- A physical package extension identifies the deployable package to the framework. This information is used to help locate a deploy driver.
- A deploy driver is used to identify connection profiles to which the package may be deployed and to deploy the package to a server represented by a connection profile.

Package Support

- **Logical Package Extension**
  - Might also be considered “abstract” or non-deployable packages
  - Identify items to be included in a package for deployment
  - May also include configuration details related to packaged items, including global settings
  - Present framework with a technology type identifier (for use in locating package constructors)
- **Physical Package Extension**
  - Might also be considered “concrete” or “deployable” packages
  - Present framework with a server type identifier (for use in locating deployment drivers)

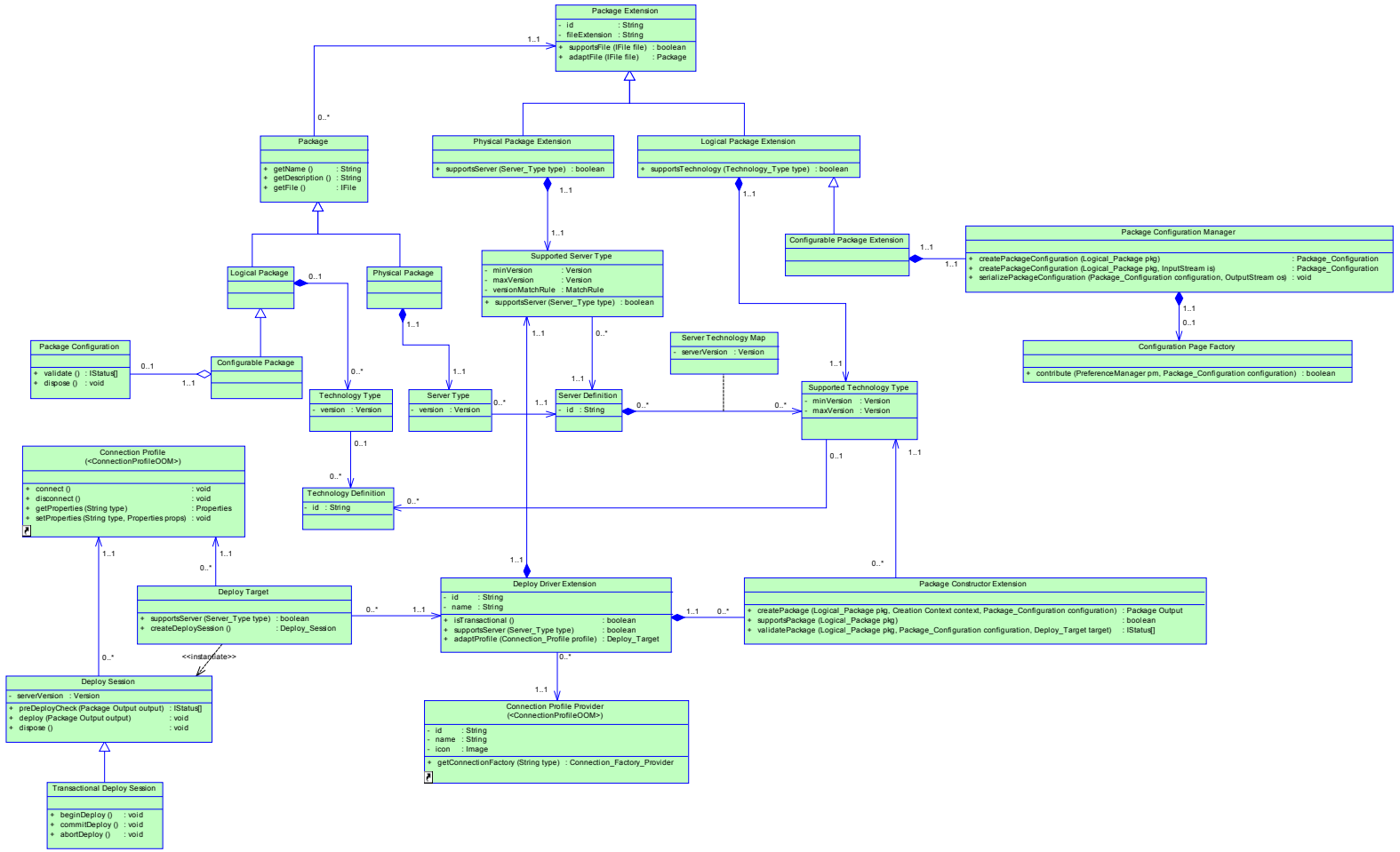
Deploy Support

- **Deployment Driver Extension**
  - Adds deployment capabilities to a connection profile
  - Presents framework with a server type identifier (e.g. Tuscany, JBI-SCA)
  - May include a list of package constructors
  - Package Constructors
    - Used for constructing deployable packages from logical package definitions (identified through technology type)
    - Typed to specific technology type and version (i.e. package constructors that support specific versions of an underlying technology; e.g. SCA v. 0.9, SCA v. X.x, SCA v. 0.9-X.x, etc.)
    - Also provide validation of logical packages based on a targeted server (e.g. does the specified target support BPEL services)

Typing

- **Technology Types**
  - Allows packages to be typed as belonging to a specific technology type (e.g. SCA v. 0.9) {Note, in practice, these are actually used to identify logical package types.}
- **Server Types**
  - Types physical packages to be typed to a specific version and class of server (e.g. Tuscany WAR, JBI-SCA)
  - Types deployment drivers as supporting a specific version and class of server
- **Map**
  - Allows specific version and class of server to be mapped as supporting a specific version and class of technology
  - Used by framework to tie package constructors (identified through a server type) to logical packages (identified through technology types); e.g. constructing a Tuscany WAR from an SCA 0.9 assembly {See above, technology type ~ logical package type}

Object Model



User Interface

- **Deployment File Editor**
  - Allows packages to be targeted to specific servers for deployment
  - Allows logical packages to be configured specially for deployment to specific servers
  - Allows packages to be targeted to more than one server
  - Allows for deployment of multiple packages
  - Restricts available targets based on package type and server support
- **Actions**
  - Create package action
    - Available for workspace files that can be identified as logical packages
    - Allows user to create a physical package from logical package definition
    - If multiple package constructors exist, user is prompted to select one (based on the name of the server type; e.g. Tuscany WAR, JBI-SCA, etc.)
  - Deploy package action
    - Available for workspace files that can be identified as logical or physical packages
    - Prompts user for server (list is comprised of servers applicable to the package)
  - Execute deployment action
    - Available for deployment files within workspace
    - Executes the deployment defined within the file (creates deployable packages; deploys packages to target servers)

Eclipse Integration

- **Plugins delivered**
  - …deploy.core
- **Extensions**
  - org.eclipse.ui.editors
  - org.eclipse.ui.newWizards
  - org.eclipse.ui.popupMenus
- **Extension Points**
  - ….deploymentExtension
    - logicalPackage
    - configurablePackage
    - physicalPackage
    - deployDriver
    - technologyDefinition
    - serverDefinition
    - technologyMap
- **Classes Available**
  - ....core
    - Interfaces for interacting with extensions
    - Package extension helper classes
  - ....core.adapters
    - Adapter classes for converting between model elements and deployment objects
    - Adapter classes for converting between resource objects and deployment objects
  - ....core.operations
    - Objects for executing common deployment operations

External Components

- **Other Eclipse Components:**
  - Connection profile framework from Data Tools Platform project (DTP)

Basic API

- **How is it used?**
  - Extension points
    - ....deploymentExtension
      - logicalPackage
      - configurablePackage
      - physicalPackage
      - deployDriver
      - technologyDefinition
      - serverDefinition
      - technologyMap
  - Classes
    - ....core.DeploymentExtensionManager
    - ....core.adapters.DeployAdapterFactory
    - ....core.operations.CreateDeployPackagesOperation
    - ....core.operations.ExecuteDeploymentOperation
    - ....core.ui.editors.DeployEditor

I18N

- **Standard string/image resource handling. Nothing exciting.**

Current Status

- **Functionality Summary**
  - Provides a UI for creating and managing deployment profiles.
  - Provides a platform that can be easily extended by other components (e.g. JBI, JEE, etc.)
  - Strengths
    - Allows users to create repeatable deployments.
    - Supports deployment of heterogeneous applications.
- **Shortcomings**
  - Does not allow overridden package configurations to be restored to their default values
  - Packages are created for each execution
  - Credentials used for connecting to servers are stored in the connection profile
  - Should support execution in a headless environment (e.g. Ant, Maven)

Future Functionality

- **New**
  - Reuse previously created packages when executing deployments (determine which packages need to be rebuilt)
  - Enable users to restore default package configurations in editor
  - Implement rollback capability for failed deployments
  - Prompt for UID/PWD when connecting (currently, profiles must be configured using credentials with a deployment role)
  - Support incremental deployment
  - Support synchronizing deployed packages with workspace resources
  - Support “undeploy”
- **Cleanup**
  - Server-Technology mapping needs to be cleaned up/consolidated
  - Technology type should be renamed; these actually correspond with package definitions
  - Package validation techniques need to be consolidated

Testing Strategy

- **Current Testing Strategy**
  - Basic automated tests
  - Manually driven minimal acceptance tests
- **Needs work…**
  - Needs more automated tests ***

Bug Fixing

- **Suggestions for bug fixing**
  - Code should be pretty straightforward (there haven't been many bugs logged against either the editor or framework)
- **Debug tips**
  - Operation classes are the main entry points for most deployment actions
  - Adapter classes are responsible for adapting objects to/from their respective deployment objects (e.g. from IFile to IPackage, IDeployTarget to IConnectionProfile)