Sam Sriramadhesikan
14 August 2012

Scripting

Tivoli software

Agenda

- Architecture
- Automation Scripts application
- Launch Points
- Variables
- Deep Dive into Launch Points
- Logging
- Resources

User Need

- Ability to rapidly extend packaged applications
- Typical user challenges
  - Limited to no Java skills; avoid Java-based development
  - Reduce system downtime across the product environments
- Primary customization areas
  - Business object extensions
  - Field validations
  - Workflow / Escalation actions
- Driven by customer and exploiter requirements
  - Maximo Advisory Council
  - IBM Service Management product family

What is a script?

- Short pieces of code
- Simplified programming model
- Usually used to glue or extend applications
- Usually interpreted (vs compiled Java or C/C++)
- Example – *add all asset spare part quantities together and set total into ASSET.SPAREQTY*:

```
spPartSet = mbo.getMboSet('sparepart')
partCount = spPartSet.count()
totalQty = 0.0
for i in range(partCount):
     partMbo = spPartSet.getMbo(i)
     totalQty += partMbo.getDouble('quantity')
mbo.setValue('spareqty',totalQty)
```

Why scripting?

- Java skills and implementation increase IT costs
  - Java/JEE developers
  - API knowledge/compatibility
  - Performance/functionality issues
  - Build process for WAR, EAR, JAR files
  - Server shutdown / re-starts
- Scripting promotes simplified programming model
- Scripting is completely dynamic (no server re-starts)

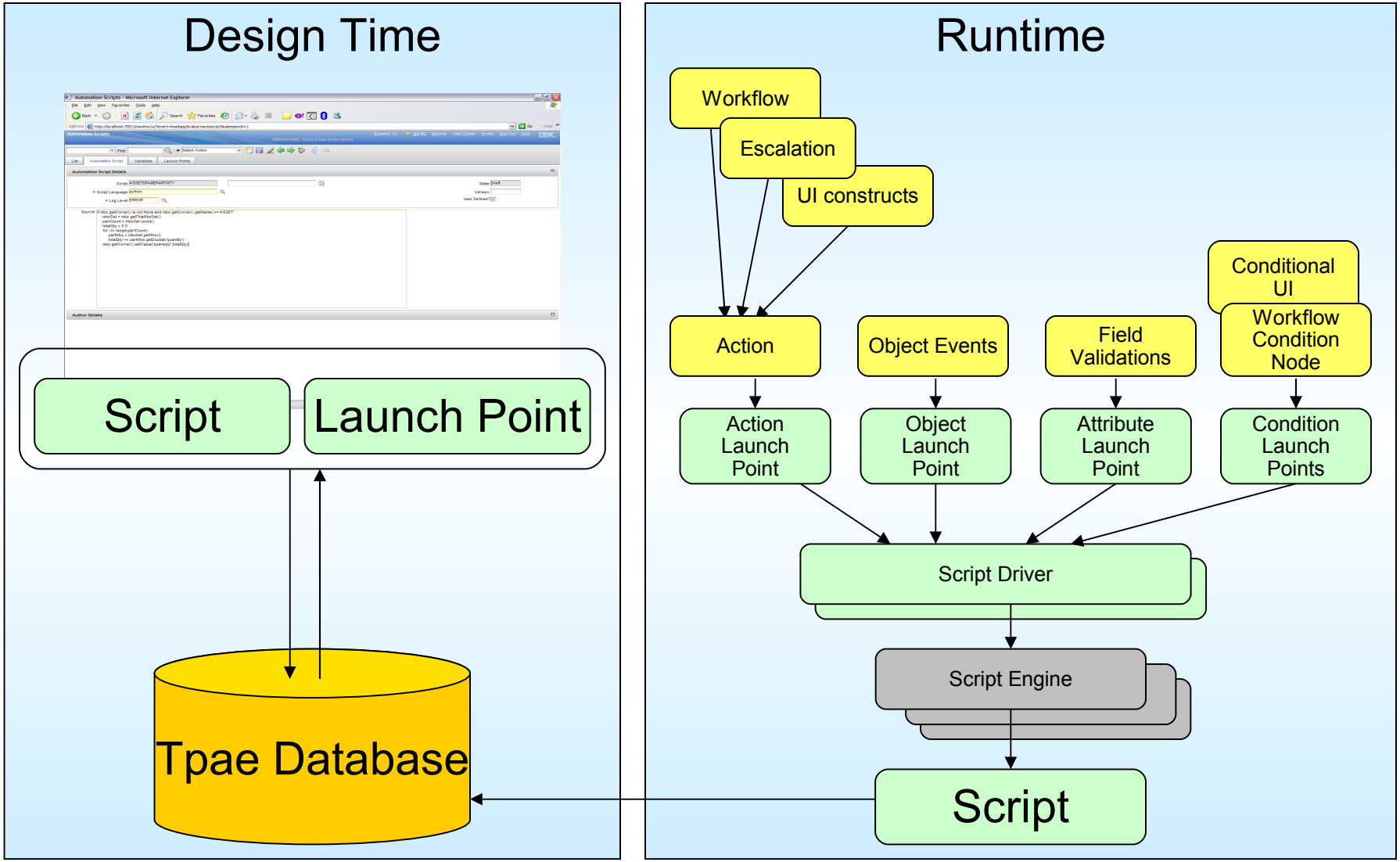
Scripting Strategy for 7.5

- Tpae 7.5 exploits scripting API that is part of JDK 6
  - JSR 223 – standardized scripting API for Java
- Supports two script engines out of the box
  - Rhino JavaScript (embedded with JDK 6, compliant with JSR 223)
  - Jython 2.5.2 (newer version compliant with JSR 223)
- Other JSR-223 compliant script engines can be seamlessly plugged in
  - Provides programming flexibility to clients and practitioners
  - Example: Jacl, JRuby, Groovy, Jawk
  - Place JARs in application server classpath and re-start server

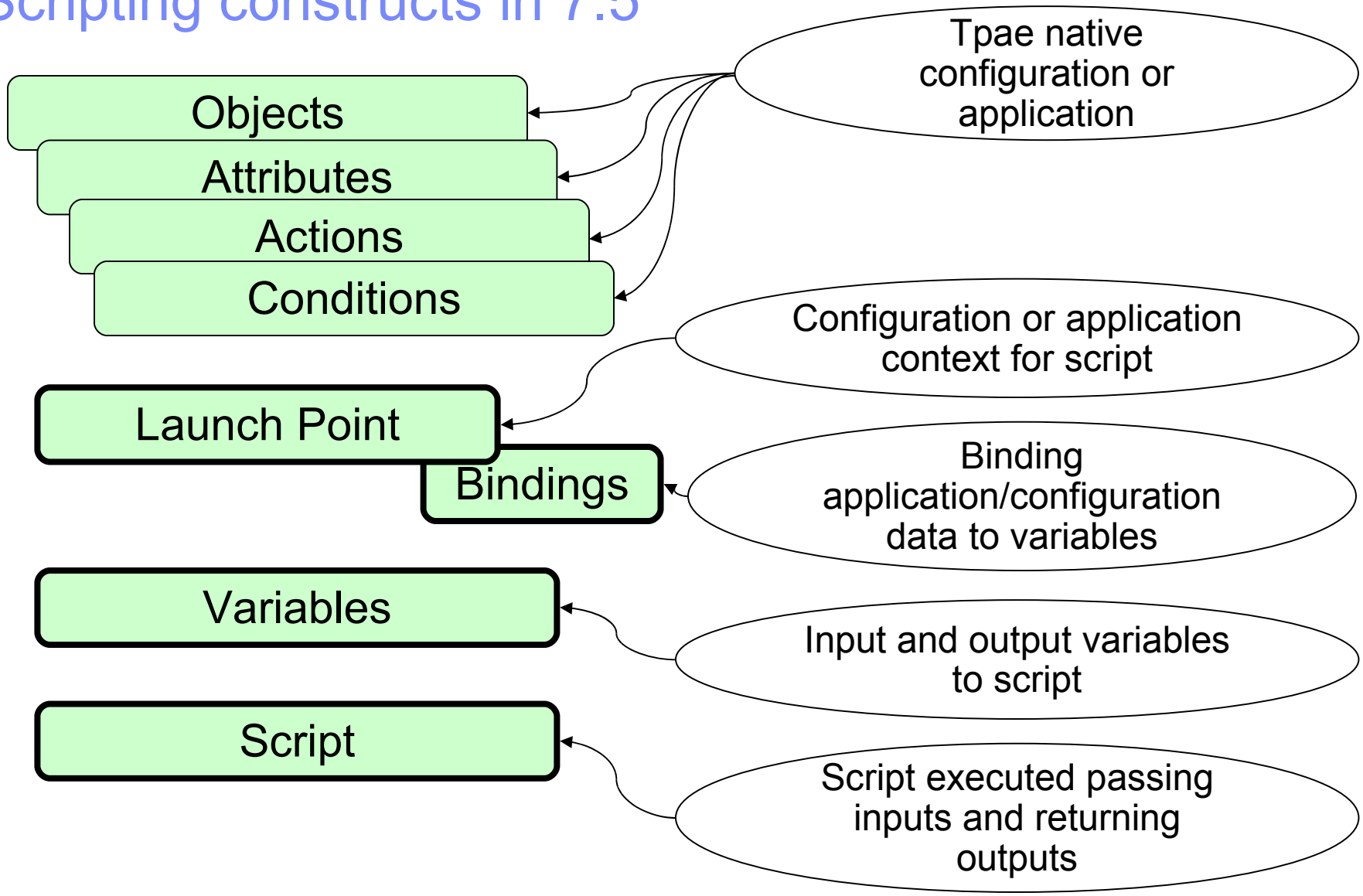
Scripting Strategy for 7.5

- Tpae scripting can be enabled for many different configurations
  - No longer limited to just actions
  - Script creation and management remains the same
- Tpae scripting supports simple coding approach
  - Launch points offer closer alignment with Tpae applications and configurations
  - Script variables and bindings to pass in data and return results
  - Detailed knowledge and experience of Maximo APIs not a pre-requisite

Scripting architecture in 7.5



Scripting constructs in 7.5



Automation Scripts application

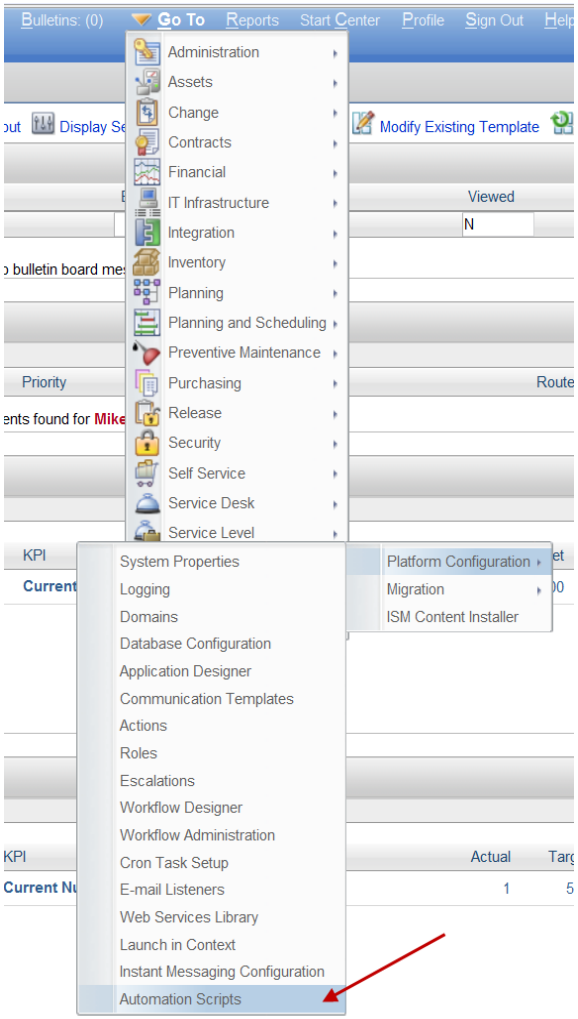
- Creation and maintenance of launch points, scripts and variables
- Wizards to create launch points
  - Object Launch Point
    - Execute scripts on MBO events such as init, add, update, or delete
    - Execute scripts conditionally (based on criteria)
  - Attribute Launch Point
    - Execute scripts during field validations
  - Action Launch Point
    - Execute scripts in the context of workflow/escalation actions
  - Custom Launch Point
    - Workflow conditions and conditional expressions can be enabled

Automation Scripts application

- Declare input and output variables
- Bind variables to MBO attributes, system properties, MAXVARs or literals
- Share script among multiple launch points
- Import existing script files created externally into application
- Promote scripts and launch points from development to production with Migration Manager
- Specify logging level and place log statements within script

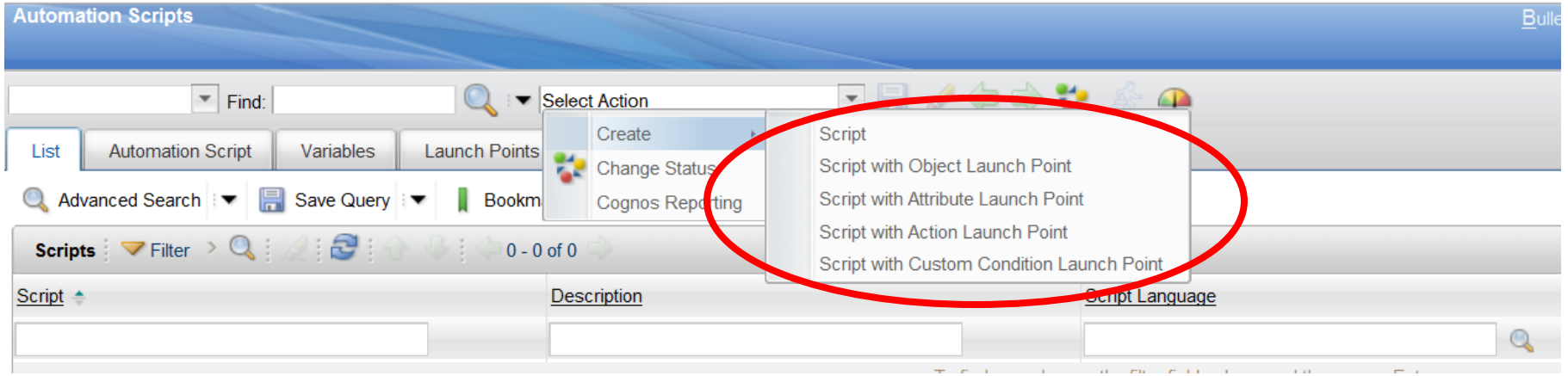
Automation Scripts application

- Go To->System Configuration->Platform Configuration->Automation Scripts
- Security Group MAXADMIN granted access to the application out of the box
- Standard power application with four tabs
  - List
  - Automation Script
  - Variables
  - Launch Points



Automation Scripts application - creation

- Create raw script
  - Script can be subsequently associated with launch point
- Create launch point associated with variables and scripts
  - Wizards to guide users through creation sequence

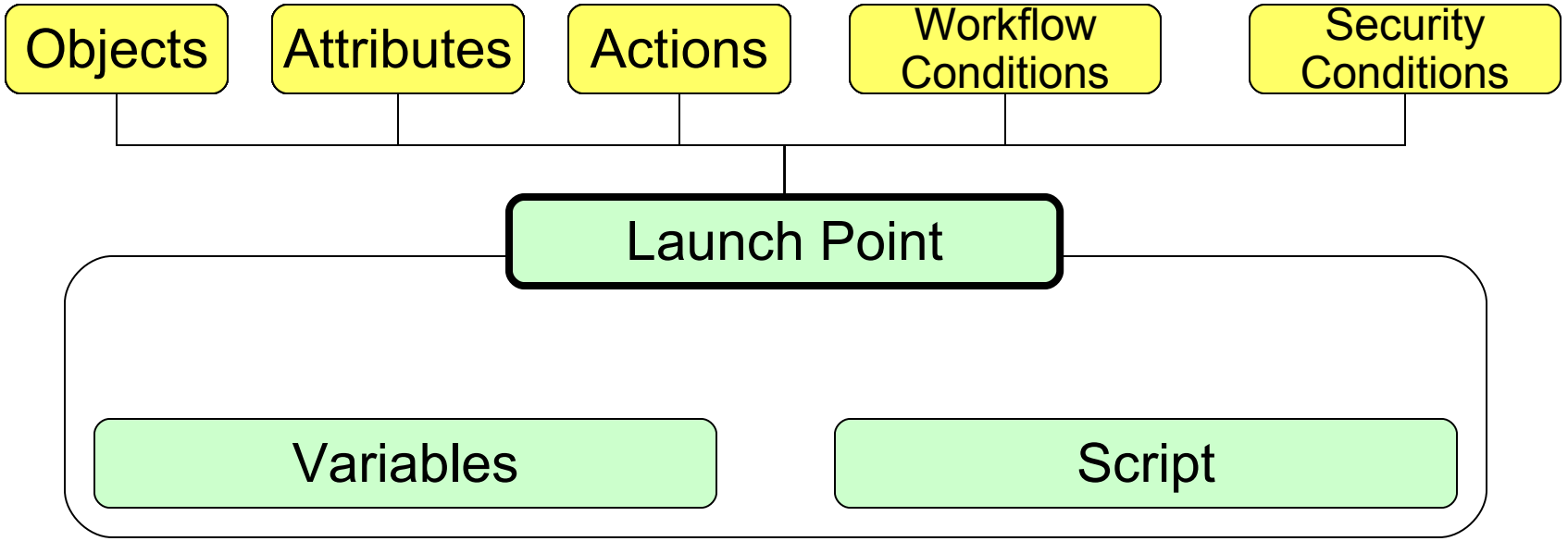


Automation Scripts application - maintenance

- Maintain existing launch points, variables and scripts from main tabs
  - Modify script code
  - Add or modify variables
  - Reconfigure launch point
  - Deactivate or activate launch point
  - Delete existing launch points, variables and scripts

Launch Points

- Launch Point is a complete script configuration
- Launch point configuration consists of three parts
  - Target application or context the script should execute on
  - Body of the script
  - Variables to be passed between application and script



Variables

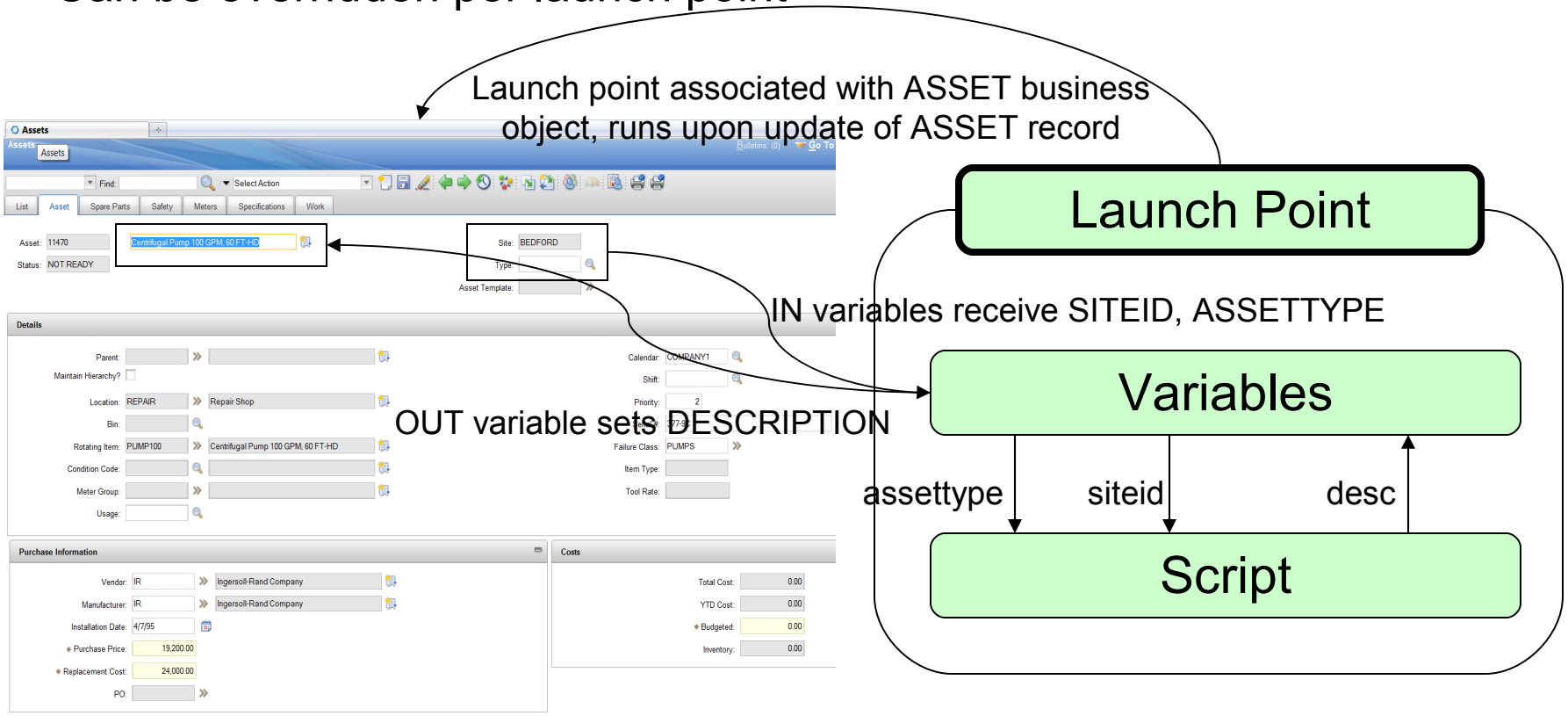
- Variables enable data to be passed to and returned from script
- Exploiting variables simplifies script code
- Variables have these characteristics:
  - Variable Type: IN (passed to) or OUT (returned from) or INOUT (both)
  - Variables of type OUT are set back into a business object
  - When setting back to business object the following can be controlled:
    - Allowing or suppressing validation
    - Allowing or suppressing access control
    - Allowing or suppressing action

Variables

- Variables have these characteristics:
  - Variable Binding: Variables can be bound to one of:
    - Business object attribute (MAXATTRIBUTE)
    - System property (SYSPROP)
    - MAXVAR
    - Literal value
  - Variable Binding: Variables can receive a global binding value that can be overridden per launch point
    - Global binding value is common to all launch points associated with the script
    - Global binding value overridden for a particular launch point, if needed

Variables

- Variables are always bound to the source or target of data
- Variable binding is declared at the script level
  - Can be overridden per launch point



Implicit Variables

- Script code made simpler by providing implicit variables
- Implicit variables are automatically available to a script
- No need to declare these separately in Automation Script application

Implicit variable	Description
user	IN variable only; provides currently logged in user's ID
app	IN variable only; provides application name that script is executing against
evalresult	OUT variable only; workflow condition and security condition scripts return true / false values always
errorgroup	OUT variable only; script code sets Tpae error group configured in Database Configuration application
errorkey	OUT variable only; script code sets Tpae error key configured in Database Configuration application
params	OUT variable only; script code sets an array of parameters that populate the error message specified by errorgroup, errorkey

Additional implicit variables

- These implicit variables are associated with a primary variable
- For example, if an OUT variable called siteid is specified for a script, then siteid_readonly will cause the associated SITEID attribute to become read-only

Implicit variable	Description
_readonly	Retrieve or set the read-only flag for a business object attribute
_required	Retrieve or set the required flag for a business object attribute
_hidden	Retrieve or set the hidden flag for a business object attribute
_internal	IN variable only; provides the Tpae internal value for a SYNONYMDOMAIN entry
_initial	IN variable only; provides the initial value for an attribute as retrieved from the MBO
_previous	IN variable only; provides the previous value for an attribute as retrieved from the MBO
_modified	IN variable only; provides a flag indicating if the value for an attribute has been modified

Variables bound to attributes

- Variables that are bound to business object attributes can be configured in a number of ways

Variable Binding	Description
Attribute	Variable receives its value from the business object's attribute
Relationship.Attribute	Variable receives its value from a related business object's attribute (multiple relationships can be traversed using Tpae standard 'dot' notation)
Arrays	Variable receives an array of values from a business object's attribute (multiple relationships can be traversed)

Array Notations

- Array notations only allowed on IN variables with MAXATTRIBUTE bindings
- Existing 7.1.x support:
  - POLINE.POCOST.costlinenum

From PO business object, traverse the 'POLINE' relationship and retrieve the first POLINE record, traverse the relationship 'POCOST' and retrieve first POCOST record and return the value of COSTLINENUM

  - POLINE[i].POCOST[j].costlinenum

From PO business object, traverse the 'POLINE' relationship and retrieve the ith POLINE record, traverse the relationship 'POCOST' and retrieve

Result of these expressions is a single value

Array Notations

- Support in 7.5:
- Existing 7.1.x features carried forward
- Array of values can be obtained in a qualified manner
  - POLINE.POCOST

  Return all POCOST records for all POLINE records

  - *POLINE[linecost>100].POCOST[percentage<100].loadedcost**

  Return an array of loadedcost values for those POCOST records where percentage is less than 100 for all POLINE records where linecost is greater than 100

  - *POLINE[linecost>100].POCOST[cond:COSTCONDN].loadedcost**

  Return an array of loadedcost values for those POCOST records that satisfy the Condition Expression specified by COSTCONDN for all POLINE records where linecost is greater than 100

Array Notations

- Business objects can be traversed in the following ways:
  - By index (example, POCOST[j])
  - By SQL filter (example, POLINECOST[linecost>100]
  - By condition expression (example, POCOST[cond: COSTCONDN])
- Results of traversal always end with a single attribute value or array of attribute values
  - POLINE[i].POCOST[j].costlinenum
  - *POLINE[linecost>100].POCOST[percentage<100].loadedcost**

Variables with binding to business object attributes can be initialized using the scripting framework array notations rather than coding with Tpae API within the body of the script.

Object Launch Point

- Defines a script configuration executed during business object events
  - Initialize
  - Insert
  - Update
  - Delete
- Supports both persistent and non-persistent objects
- Business object events can be filtered to meet specific criteria
- Three-step wizard to create configuration

Object Launch Point Demo

Requirement:

When a new asset record is created with the Assets application, the asset number should adhere to the following naming convention: asset number should be prefixed with the type of asset being created.

Asset Type	Prefix
FACILITIES	FT
FLEET	FL
IT	IT
PRODUCTION	PR

- Scripting ingredients to implement this requirement:
  1. Define an Object launch point that executes whenever a newly inserted asset record is saved
  2. Define the script that associated prefix with and associate with the Object launch point
  3. Test the Object launch point and script immediately and activate or de-activate as desired

Attribute Launch Point

- Defines a script configuration executed during field validations
  - Script executes against the validate() method of MboValueAdapter
- Supports both persistent and non-persistent attributes
- Script can trigger display of informational, warning or error messages based on validation logic
- Three-step wizard to create configuration

Attribute Launch Point Demo

Requirement:

If Purchase Price on an asset is more than $100, Vendor field must be populated. If Purchase Price on an asset is less than $100, Vendor field is not required. For any Purchase Price, Replacement Cost should be half the Purchase Price.

- Scripting ingredients to implement this requirement:
  1. Define an Attribute launch point that executes whenever Purchase Price field is tabbed out of
  2. Define the script that configures the fields and associate with the Attribute launch point
  3. Test the Object launch point and script immediately and activate or de-activate as desired

Action Launch Point

- Defines a script configuration executed as part of workflow process or escalation point
  - Action record is generated during launch point creation
  - Action record is of type ‘CUSTOM’
  - Tpae 7.5 scripting provides out of the box Java class callable from action to execute script
  - Out of the box Java class accepts three parameters:

Param 1	Script name
Param 2	Launch Point name
Param 3	Action name

  - Parameters are auto-populated during launch point creation

Action Launch Point Demo

Requirement:

As part of a workflow process operating on a Service Request record, system initiated processing computes target contact and target start dates and also creates a work log entry against the Service Request.

- Scripting ingredients to implement this requirement:
  1. Define an Action launch point that computes target contact and start dates and generates work log entry
  2. Define the workflow process that incorporates the Action launch point
  3. Test the workflow process, action launch point and script immediately and activate or de-activate as desired

Custom Condition Launch Point

- Conditions are commonly configured in Tpae
- Two types of conditions are enabled for scripting in Tpae 7.5
  - Worflow Conditions (as defined in Condition Node)
  - Security / Conditional UI (as defined in Condition Expression Manager
- Two steps to fully enable scripted conditions:
  - Define the Custom Condition Launch Point
  - Associate the Launch Point with workflow Condition Node or Condition Expression Manager
- Condition Node and Condition Expression Manager configuration must be performed in native applications
  - Workflow Designer (define CUSTOM condition in workflow canvas)
  - Condition Expression Manager (define CUSTOM expression)
    - Type=CLASS, Class=com.ibm.tivoli.maximo.script.ScriptCustomCondition
    - Expression should hold launch point information: <script name>:<launchpoint name>
- Tpae 7.5 scripting provides out of the box Java class callable from either type of condition
- Script must always return true or false (evalresult OUT variable)

Custom Condition Launch Point Demo

Requirement:

For all users in the MAXADMIN security group, for assets that have status of DECOMMISSIONED, Assets application must not display the Spare Parts tab.

- Scripting ingredients to implement this requirement:
  1. Define an Custom Condition launch point evaluates the status of an asset record
  2. Define the condition expression that will invoke this script
  3. Define the signature option that will controls access to the Spare Parts tab
  4. Associate the condition expression and signature option with MAXADMIN security group
  5. Test the complete configuration immediately and activate or de-activate as desired

Skills

- Scripting is targeted at exploiters and implementers
- Exploiters may deliver out of the box script content for their product capabilities
  - TSRM Service Catalog shopping cart scripts
  - CCMDB / RBA script to trigger provisioning workflows in TPM
- Implementers create scripts to accelerate production roll out
- Experience with Tpae configurations is a pre-requisite
  - Data model and relationship knowledge extremely valuable
- Knowledge of scripting language syntax and operations is a pre-requisite
- Knowledge of Tpae business object API is an advantage, though not required

Deployment approach

- Scripts should be created and tested in development environments
- Application behavior with and without scripts can be measured – simply activate or de-activate the script
- Scripts can be packaged using Migration Manager and distributed to pre-production and production environments
  - Migration group SCRIPTCFG
  - DMSCRIPT object structure
  - DMLAUNCHPOINT object structure
  - Launch points that were active in source will be migrated and set active in target

Script Logging

- *autoscript* logger set to ERROR level out of the box
- All log statements from scripting framework and individual scripts processed by *autoscript* logger
- Logging for individual scripts
  - Place print statements inside the script
  - Print output redirected to *autoscript* logger
  - Each script can configure the log level of its print statements
  - All print statements inside body of script redirected to same log: no support to generate mixed logs (example, ERROR and DEBUG)
- Recommendation is to redirect scripting logs to dedicated log file

Script Logging

- Log statement indicates execution time
  - Irrespective of log level
  - System property *mxe.logging.CorrelationEnabled* must be enabled

```
15 Mar 2011 22:45:40:333 [INFO] [MXServer] [CID-MXSCRIPT-102] Correlation started.

15 Mar 2011 22:45:40:423 [INFO] [MXServer] [CID-MXSCRIPT-102] The total time taken to execute
  the CONCATSR script for the CONCAT  launch point is 16 ms.

15 Mar 2011 22:45:40:423 [INFO] [MXServer] [CID-MXSCRIPT-102] Correlated data: BEGIN
  evalScriptTime:89 app:SR Script:CONCATSR MboId:288 MboName:SR LaunchPoint:CONCAT
  evalINParamsTime:1 UserName:SR evalOUTParamsTime:0 ElapsedTime:90 ms  END
```

- Specify DEBUG level in autoscript logger to obtain additional execution details on a script
  - Prior to executing, print the binding values for script variables
  - After executing, print the updated values
  - If errorkey and errorgroup being set, indicate if the message successfully retrieved

Script Compilation and Execution

- Upon save, script is compiled and cached
  - A pre-compiled script can be executed multiple times without the need to reparse or recompile
  - Pre-compilation can be done only with those scripting engines that support compilation
  - Pre-compilation and cacheing make script execution more efficient
- Any syntax errors found reported in application as popup message
- Jython / JavaScript standard interpreter executes script
- If additional imports are used, then the corresponding libraries must be present in application server CLASSPATH

Resources

- Maximo 7.5 Information Center for Automation Scripts application help
- Jython scripting
  - http://www.jython.org
- Javascript (Rhino) scripting
  - http://www.mozilla.org/rhino/
- JSR-223 scripting for Java specification
  - http://jcp.org/aboutJava/communityprocess/pr/jsr223/
- Tpae 7.5 Scripting Cookbook
  - http://ibm.co/pPI32E
- Service Management Connect blogs
  - https://www.ibm.com/developerworks/servicemanagement/am/index.html