Attend a *live event* at your computer.

Microsoft BizTalk Server 2004 Rules Engine

Rajan Dwivedi
Consultant, Premier Support
Microsoft Corporation

Session prerequisites

- Familiarity with .NET
- Experience with XML, XPath, and ADO.NET
- Basic orchestration knowledge
- Working knowledge of C#
- Awareness of Microsoft® Visual Studio® .NET IDE
- Basic relational database management system (RDBMS) skills

Agenda

- Introduction to business rules
- Developing business rules
- Invoking rules from orchestration
- Advanced concepts
- Rules and orchestrations
- Q&A

What are business rules?

Business rules

- Statements that govern conduct of business processes
- Provide separation of rules from application code
- Give business users control of business logic

Benefits

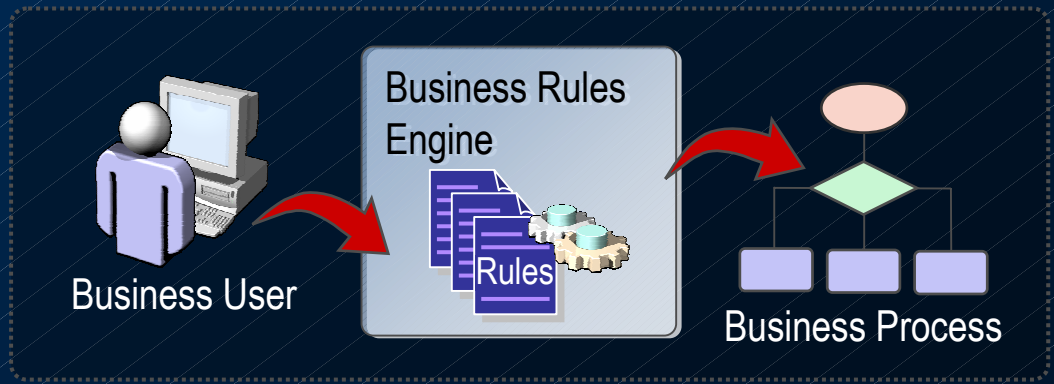
- Faster response to change
- Reusability of work
- Lower total cost of application development and ownership

Business rule examples:

- Today's interest rate = ***6.5%***
- Apply ***20%*** discount to all sales, this week only
- If order over ***$250k***, send to credit department for approval

Remember

- Business processes stay the same over time, but business rules are volatile!



Business rules (continued)

Information workers can use rules to:

- Define and own business policies
- Change policies in real time

No coding or rebuilding required

Use business rules to:

- Classify requests
- Trigger notifications
- Automate approvals
- Escalate issues
- Reroute documents

Business Event	Business Decisions	Rules
Customer Purchase Order Request	• Is customer known? • Is product known? • Is product available? • Is product low on inventory?	• If customer ID exists, then customer is known • If customer is preferred and quantity exists on shelf, then product is available • If customer is not preferred and quantity + 10 exists on shelf, then product is available • If quantity on hand < reorder quantity, then product is low on inventory • If product is low on inventory, then reorder product

BizTalk rules engine in brief

- Lightweight .NET engine
  - Integrated with Microsoft BizTalk® Server
  - Host-agnostic for in-proc scenarios
- Powerful tools and framework
  - Supports developers, information technology admin, and information worker
  - Complete application programming interface (API) for design and execution
- Rich modeling capabilities
  - Declarative statements
  - Logical expressions (AND/OR/NOT/custom)
  - Data sources: .NET objects, XML, DB
- Enterprise-ready
  - Security, performance, versioning

Business rules concepts

Term	Description
Policy	• Logical grouping of rules
Rule	• Contains a condition and a set of actions
Condition	• Contains predicates applies to facts
Facts	• Data upon which the rule operates
Actions	• Functions called when the condition is evaluated to true
Vocabulary	• Contains set of definitions (business-specific terminology)
Definition	• User-friendly name/string for technical data
Rule Engine	• Evaluates condition based on facts and executes actions if condition evaluates to true
Rule-based application	• Invokes policies and submits facts
Rule store	• Repository for rules and policies

Tools for business rules

- Business Rule Composer
  - Develop and deploy policies
- Microsoft Visual Studio .NET
  - Integrate policies and processes
  - Build supporting code
    - Fact Retrievers – long-term facts
- Rule Engine Deployment Wizard
  - Deploy/export/import policies and vocabularies
- Health Activity and Tracking (HAT)
  - Tracking configuration
  - Policy execution monitoring

Business rules personas

- Information worker
  - Design and test policies
  - Use business vocabularies
- Developer
  - Integrate policies and process
  - Bind vocabularies
- IT pro
  - Security, deployment, and migration of policies
  - Track policy execution

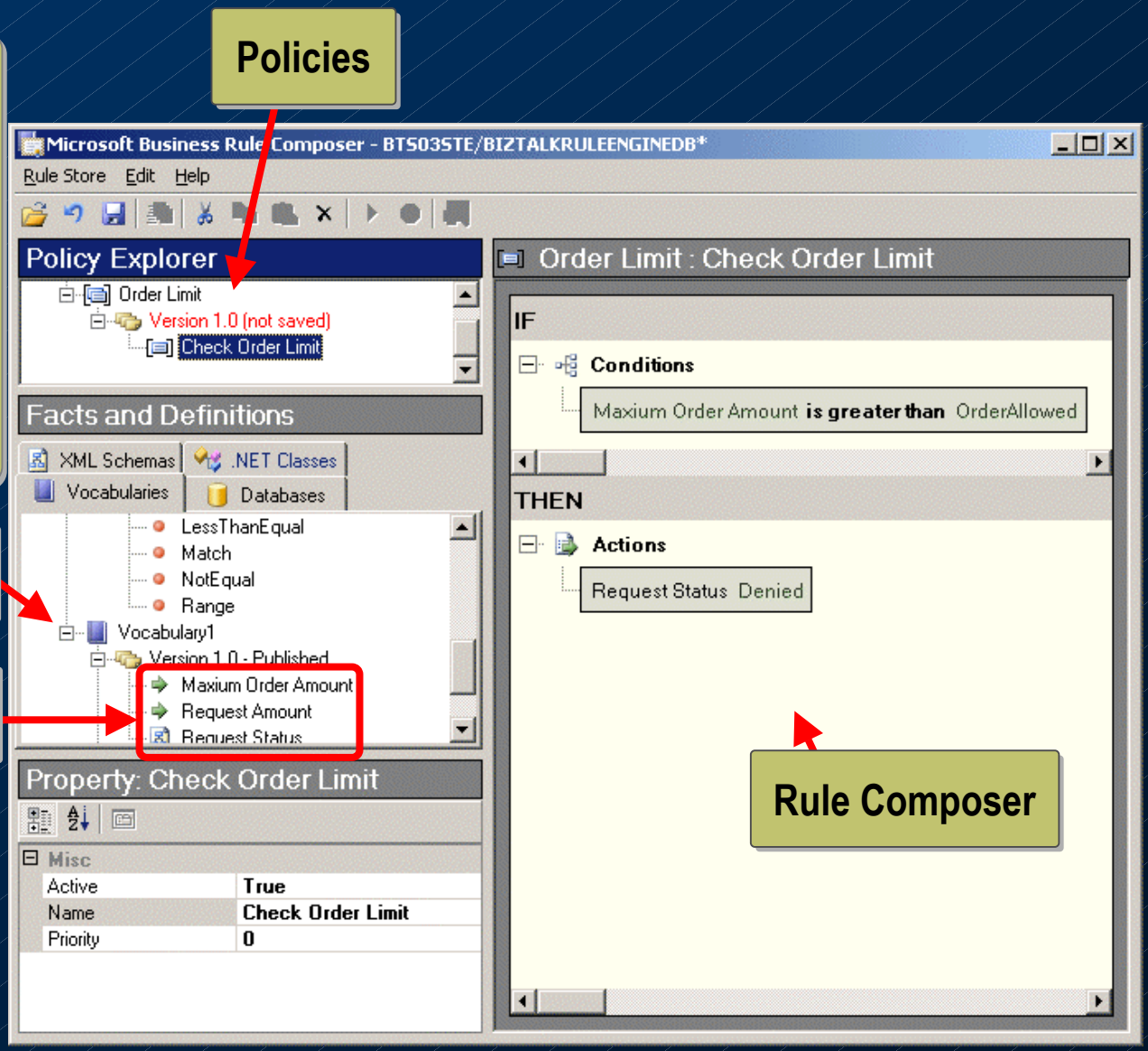
- Composer: stand-alone app
- Version policies
- Use policies through InfoPath™

- Inside Visual Studio .NET
- Composer: for policy bindings
- .NET APIs

- Inside HAT
- Deployment wizard
- Manage security

Developing business rules using Business Rule Composer

- Create versions of vocabularies
- Create vocabulary definitions
- Compose policies
- Publish vocabularies and policies



Developing vocabularies

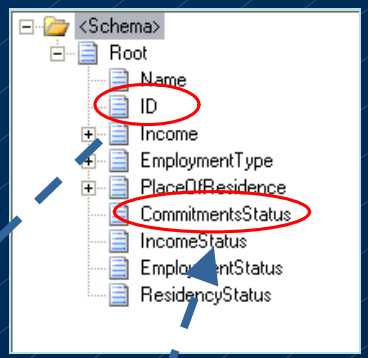
Steps:

1. Add new vocabulary version.
2. Add definitions to vocabulary based on data from XML document, database, or a .NET class.
3. Save vocabulary version, it can be modified later.
4. Publish vocabulary; cannot be modified, can be used in developing rules.

Design-time features

- Vocabulary elements to associate technical implementation to business terms

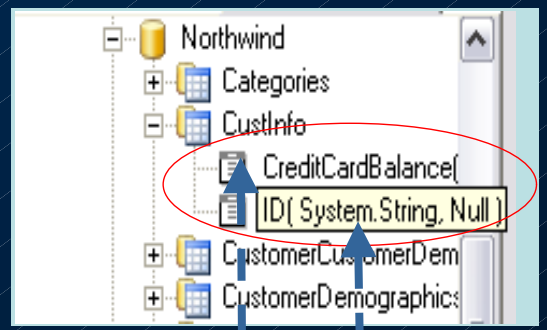
XML message



.NET component

LoanData.CreditBalanceLimit

Fact store



IF

The Loan ID *is equal to* a valid ID in customer database

AND

Credit Card Balance *is greater than* Maximum Limit

THEN

Set Commitments Status in the loan to "Compute "

Developing policies and rules

Steps:

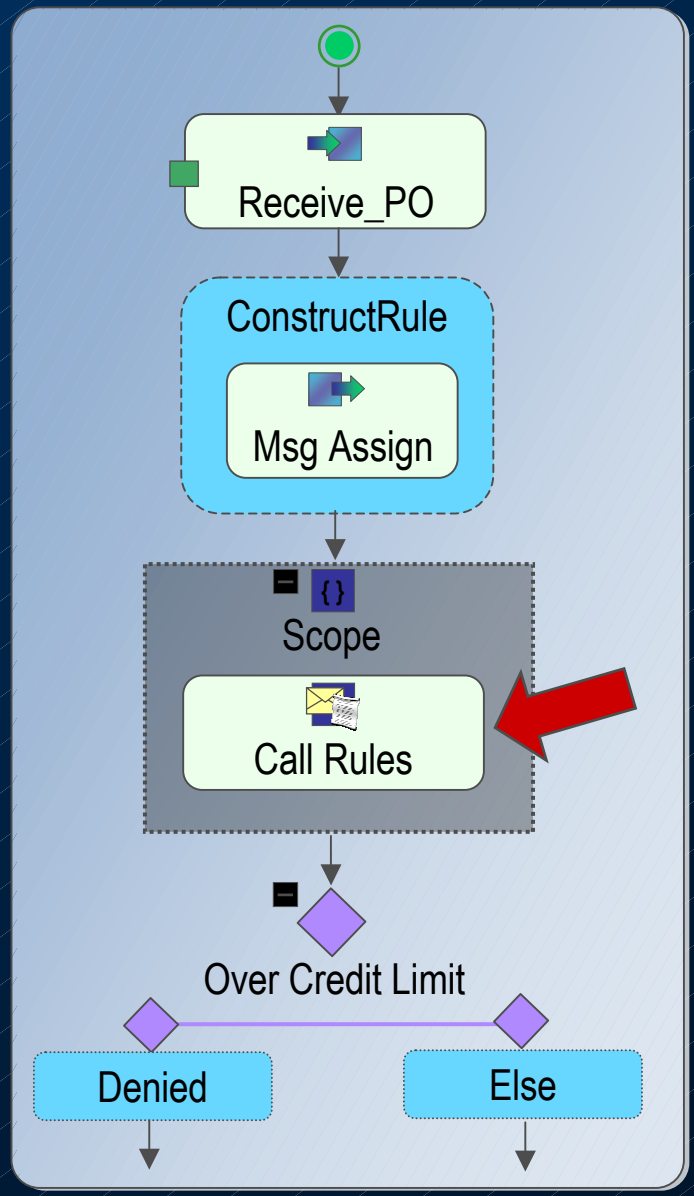
1. Add new policy version.
2. Add rules to policy version.
3. Define condition based on facts and predicates.
4. Specify actions for the rule.
5. Save and test the policy.
6. Publish policy; cannot be modified later.
7. Deploy policy so that it can be used from client.

Invoking rules from orchestration

- Call rules shape
- Programmatically invoking rules
  - Use “Expression” shape in orchestration
  - Use .NET components

Call rules shape

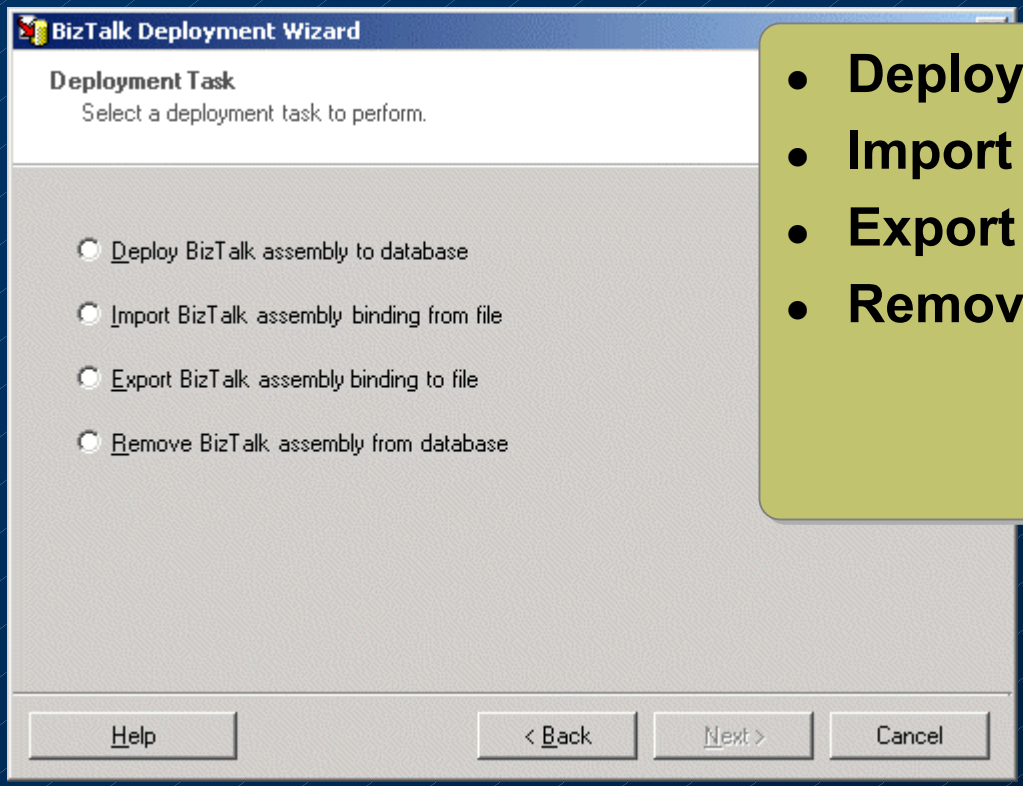
1. Add reference to rule engine assembly.
2. Insert a scope shape and set transaction type to atomic.
3. Put Call Rules shape inside the scope.
4. Configure the shape to specify a policy.



Programmatically invoking rules

- Add a reference to rule engine assembly
- If using Orchestration, then
  - Insert scope shape (atomic)
  - Insert expression shape
- Create a Microsoft.RuleEngine.Policy object
- Call Execute method to execute the policy
- Add facts as needed

BizTalk rules deployment wizard



- Deploy policies
- Import from XML file
- Export to XML file
- Remove policies

Advanced concepts

- Forward chaining execution
- Short-term facts vs. long-term facts
- Fact creators
- Rule engine architecture
- Rules and orchestrations
- Enterprise features/scalability

Forward chaining execution

- Facts and data inserted into working memory
- Matching stage
  - Evaluates rules conditions based on facts
  - Selects candidate rules for execution
- Conflict resolution stage
  - Highest-priority rule wins
- Action stage
  - Actions are executed
  - Can assert facts into working memory, causing more rules to be fired

Forward chaining example

Policy

- Rule 1 – “Auto Approve”
  - If A.Quantity <= 500
  - Then B.Status = “Auto Approved”
- Rule 2 – “Promotion”
  - If A.TotalSale > 10000
  - Then A.Quantity = A.Quantity +10
- Rule 3 – “High Volume”
  - If A.Quantity > 500
  - Then B.Status =“High Volume”

Rule Engine Agenda

B.Status = Auto Approved

A.Quantity = A.Quantity+10

A.Quantity = 495;

A.TotalSale = 11000

Forward chaining example (continued)

Policy

- Rule 1 – "Auto Approved"
  - If A.Quantity <= 500
  - Then B.Status = "Auto Approved"
- Rule 2 – "Promotion"
  - If A.TotalSale > 10000
  - Then A.Quantity = A.Quantity +10
- Rule 3 – "High Volume"
  - If A.Quantity > 500
  - Then B.Status ="High Volume"

Rule Engine Agenda

A.Quantity = A.Quantity+10

- ~~B.Status = Auto Approved~~

B.Status = High Volume

A.Quantity = ~~495~~ 505

B.Status = "High volume"

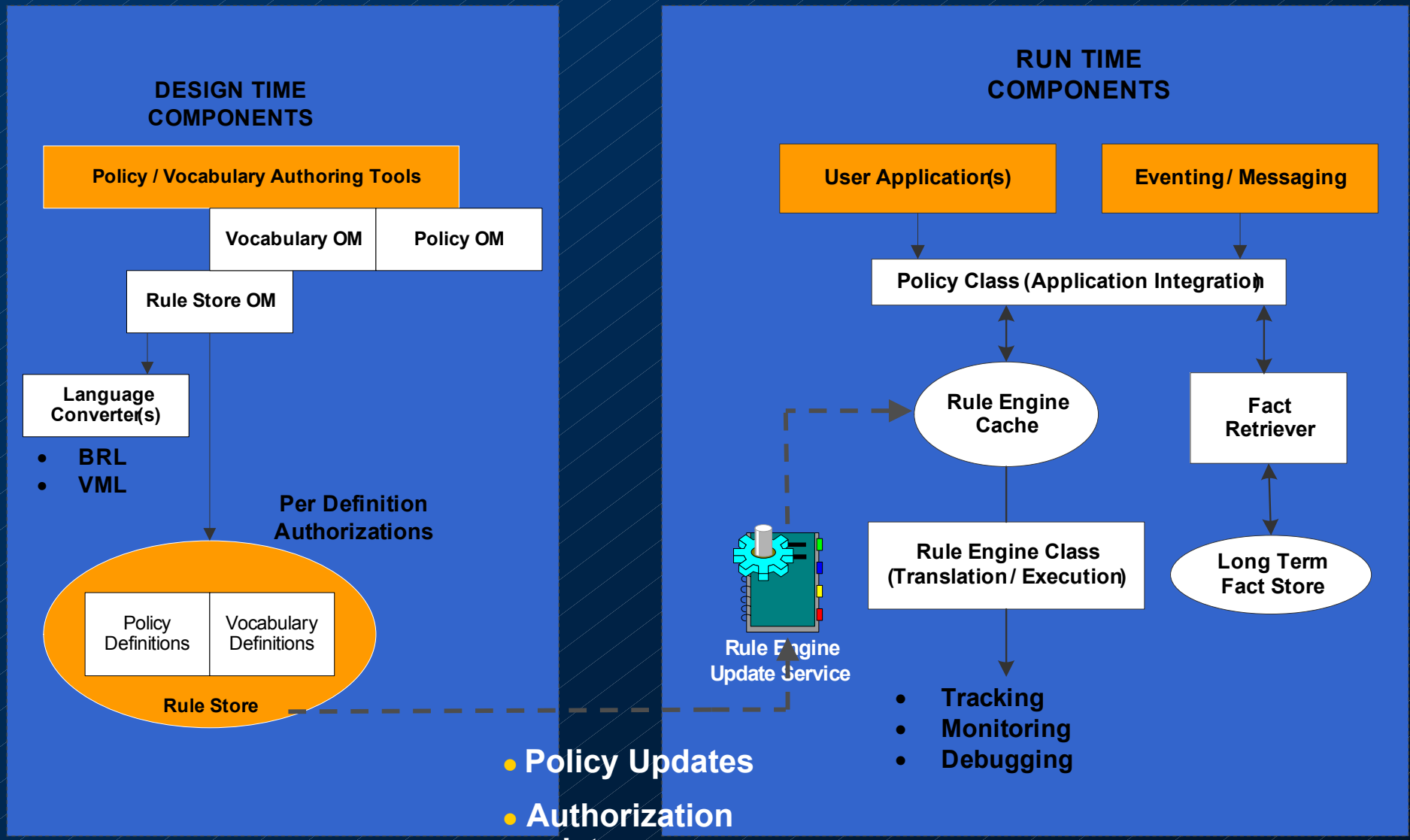
Short-term facts vs. long-term facts

- Short-term facts are available only for one execution cycle of the rule engine.
- Long-term facts are available for multiple execution cycles of the rule engine.
- Fact Retriever:
  - Provides plumbing mechanism for loading facts from the custom store.
  - .NET component implementing IFactRetriever interface.
  - Associated with a policy version.
  - UpdateFacts method is called by rule engine.
  - Retrieves long-term facts from the fact store and asserts them into working memory of rule engine.

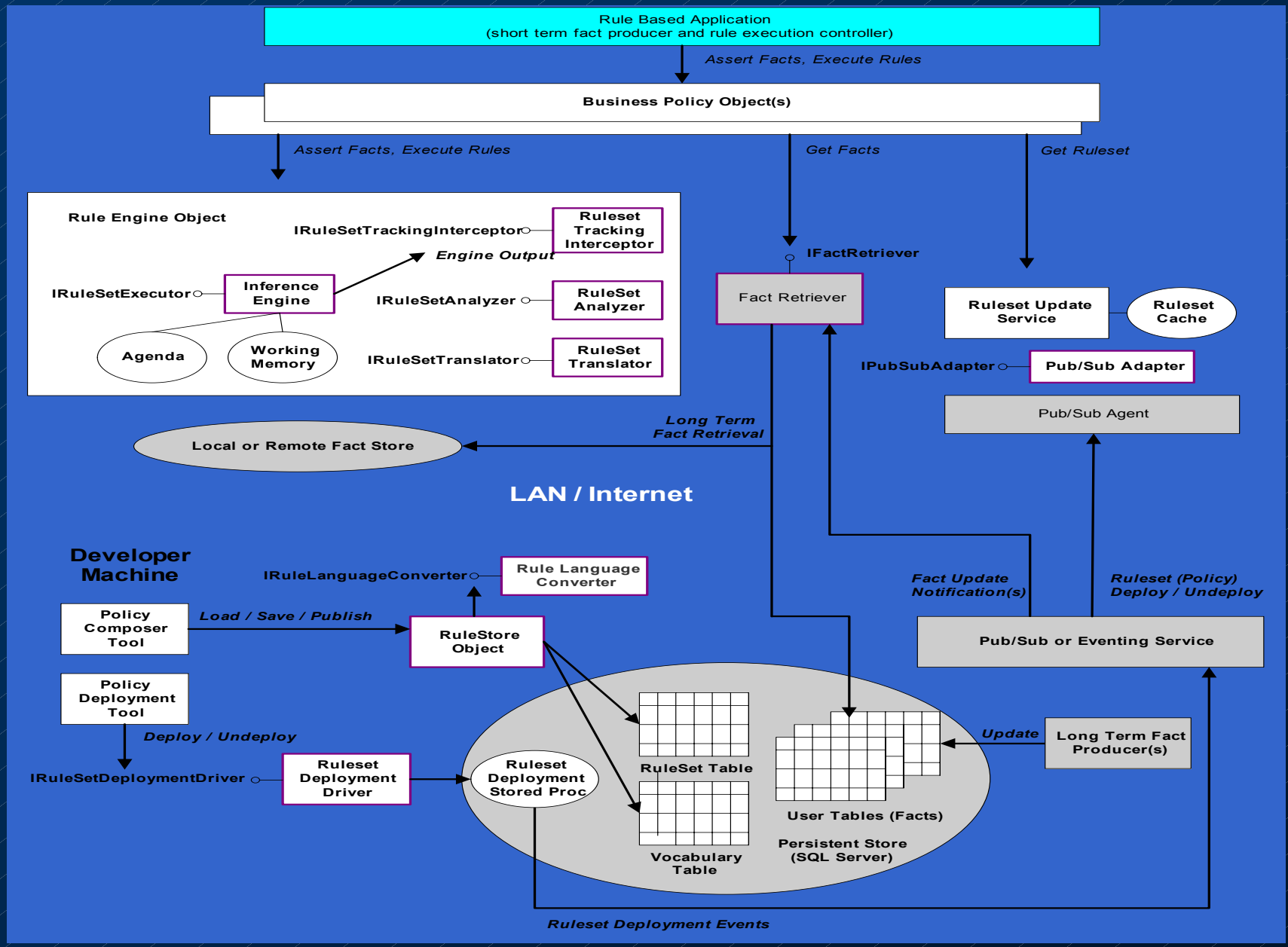
Fact creators

- Fact creator class is mainly used for policy testing
- Creates instances of facts
- .NET component implementing IFactCreator interface
- CreateFacts method is called

Rules engine framework architecture



Rules engine under microscope



Engine control functions

- Assert
  - Add facts to working memory
- Retract
  - Remove facts from working memory
- RetractByType
  - Remove all instances of the specified type
- Update
  - Instruct engine to reevaluate relevant facts/conditions
- CreateObject
  - Exposed as constructors in Composer
- Halt
  - Stop engine processing
- Clear
  - Clear facts from memory and clear actions from agenda
- Executor
  - Handle to the rule engine instance

Rules or orchestration

Where do I use them?

Use rules for:

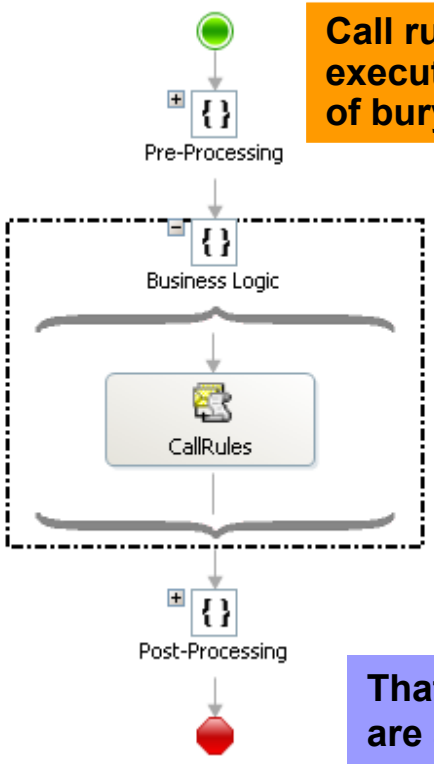
- Discrete, point-in-time evaluations and calculations
- Large number of permutations to encode in a control flow
- Fact-based inferencing where control flow cannot be predefined

Use orchestration for:

- Formal workflows that require:
  - Long-running semantics
  - Transactions/compensations
  - Messaging
- Known control flow that must be rigorously managed for performance/scale
- Visibility and tracking are critical

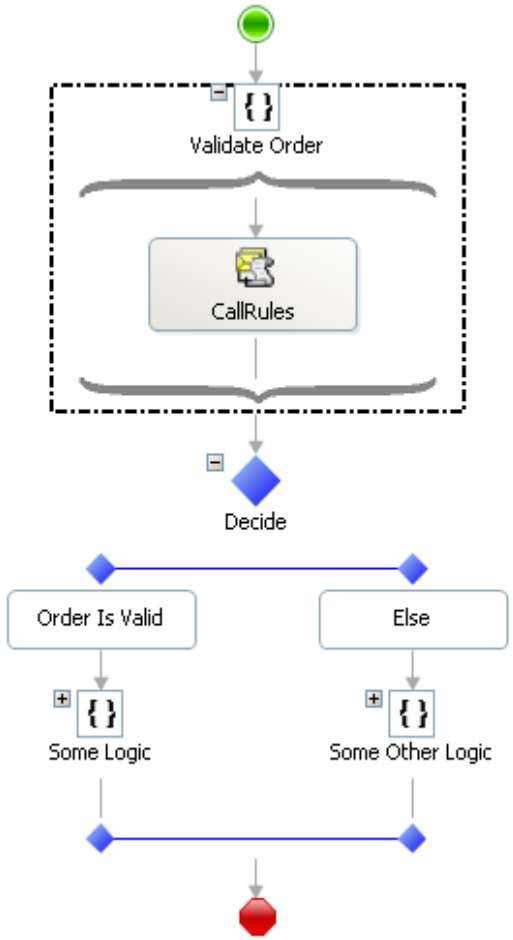
Rules and orchestration

Illustrated patterns



Call rules engine to execute logic, instead of burying it in code.

That's what rules are all about!

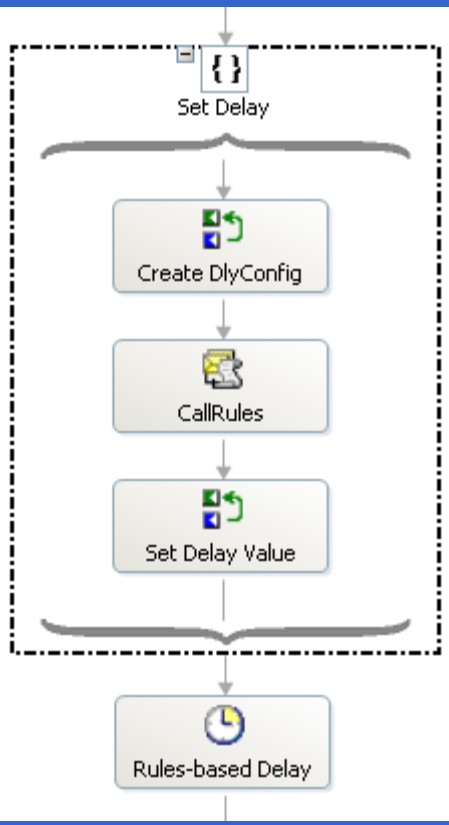


Call rules engine and determine path.

Set the field values and use them for decision making!

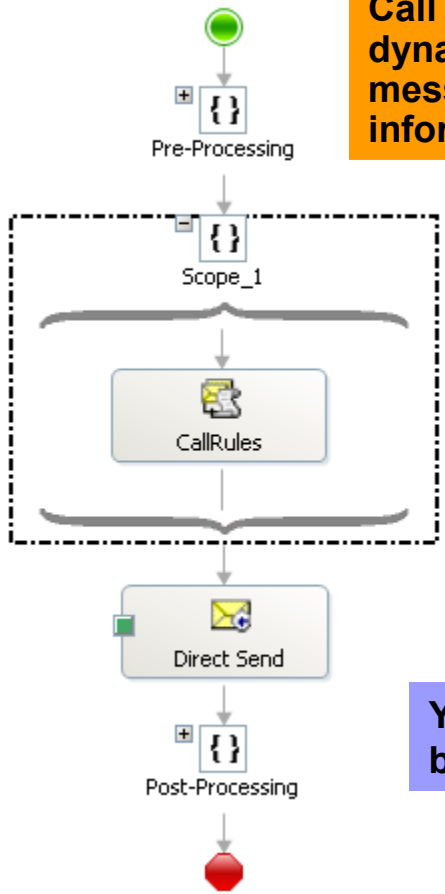
Rules and orchestration

Illustrated patterns (2)



Use the rules engine for dynamic or variable delays.

Think of rules as configuration mechanism!



Call rules engine to dynamically populate messaging routing information.

You can use rules based routing!

Using rules for calculating certainty factor

- Typical scenarios may require finding a risk or certainty factor for a rule set in a policy
  - For an insurance policy example, business setup/acquisitions, bank loans, stock analysis, and more
- You can implement this in an action of each rule
  - When Rule1 is fired Risk = Risk + 10%
  - When Rule3 is fired Risk = Risk – 15%
  - When Rule9 is fired Risk = Risk * 2
- .NET components can be leveraged for sophisticated algorithm
- If final Risk > 67% then reject it!
  - Like loan, acquisition, and more

Enterprise-ready framework

- Authorization
  - Role-based access to the policies/vocabularies through logical authorization groups
- Versioning
  - “Latest version” or “specific version” with roll-back
- Rule Engine Update Service
  - Dynamic updates with configurable caching
- Performance and scalability
  - Optimized eager-match algorithm for smaller rule sets (RETE) and Microsoft algorithm for larger rule sets
  - Scales to large rule sets and fact bases

Rule engine scalability

Exit criteria tested for Voyager (beta):

- Max rule set size: 50,000 rules
- Max number of rule sets: No upper bound
  - Tested against SQL Server™–based rule store
- Concurrent engine instances (scale up): 8
- Tested 8 parallel engine instances (8 CPU computer)

Conclusion

- BizTalk Rules Engine
  - When combined with orchestration technology, enables truly agile process platform TODAY.
  - Enables business users to fully participate in business processes.
  - Increases business performance management platform return on investment through policy consistency across the organization.

Attend a live event at your computer.

Thank you for joining us for today's event.

- For information about all upcoming Support WebCasts, and access to the archived content (streaming media files, PowerPoint® slides, and transcripts), visit the Support WebCast site at http://support.microsoft.com/WebCasts/
- We sincerely appreciate your feedback. Please submit any comments or suggestions about the Support WebCasts on the "Contact Us" page of the Support Web site at

  http://support.microsoft.com/servicedesks/webcasts/feedback.asp.