Why WebSphere Operational Decision Management?

A Case for Business Users of Information Technology

Contents

Introduction

To the business user of information systems, the relationship with the IT personnel who build and maintain those systems can be a contradictory experience: on the one hand, IT delivers significant productivity benefits by automating routine tasks and improving access to information; on the other, the process of building automated systems often requires embedding business policies and automated decisions into software systems, where they become difficult to maintain and adapt.

The relationship between business and IT frequently consists of miscommunication, misunderstanding and frustration. Part of the disconnect between the IT organization and business groups is a natural disparity between the work cycles of the two groups. To the business, the IT system development cycle appears to be a long, drawn-out process. Further, once the policy managers define the requirements, it is often a leap of faith that the implementation will fulfill their needs. It is not easy for policy managers to follow their requirements through the implementation process once they are transformed into technical code. In the implementers' defense, what they receive as requirements is often not sufficiently detailed and precise; many requirements are determined once the development process has started, sometimes with the concurrence of the business people and other times at the discretion of the development team.

More importantly, though, once the work is accomplished, the process for changing business decision logic requires going through the same complex consultation and translation process for every revision. Thus, policy and decision changes become entangled with a software development cycle that is driven by a host of technology and resource issues, and

is insufficiently responsive to the business sponsors' needs. In effect, the owners of the business policy lose control over its expression and evolution when it is handed off to be embedded in a software solution—and they never regain control. The traditional code-based approach for building business systems limits the business sponsor's flexibility to adapt their operations to dynamic market conditions, individual customer demands, or changing regulatory environments.

However, implementing decision logic using *IBM WebSphere Operational Decision Management* enables a giant leap forward in bridging the gap between business people and IT. Using this offering, the line-of-business defines their policies and automated decisions in a form that is both understandable to the non-technical user and consumable by the systems that will use them, providing clear communication between the policy managers defining the requirements and the developers implementing the business system solution.

This approach allows line-of-business personnel to directly define how their business policies are being executed, and, perhaps more importantly, the system facilitates change in an easy, safe and reliable manner. In traditional information systems, the business policies get hard-coded into the application. When changes need to be made, the entire software development life cycle must restart—understand the requirement, design it into the system, make sure it doesn't adversely impact anything else, implement the change, test and then deploy it (with several iterations in many cases before getting to deployment). IT rarely wants to go through this process for a single change, so requirements must be bundled into reasonable work efforts, further slowing down the process.

With WebSphere Operational Decision Management (WODM), the business decision logic (again, the part of systems most likely to require change) is separated out and can be changed without impacting the remainder of the application. An assessment of the impact of the change on other decision logic must still be done, but this is much simpler than a full system impact assessment and WODM provides comprehensive capabilities to do this. A single change can be assessed, implemented and tested in a very short timeframe (often hours). You do not have to wait until you have additional changes to make the effort worthwhile.

Once the problem is stated in this way, the solution becomes clear: the expression of business policies and the ability to change them need to be decoupled from the systems that use them, enabling:

- Business policy experts to manage and evolve decision logic using the methods and vocabulary with which they are most familiar
- Technology experts to manage and evolve systems using the methods and vocabulary most suitable to their tasks.

WODM enables this separation of concerns—it provides an integrated set of capabilities, called WebSphere Decision Center, which policy managers and software engineers use to build flexible applications in which decision logic is abstracted out of software code, where it can be directly authored, modified and managed independently of the underlying software system.

Business policies, business rules and business events

In order to explain how WebSphere Operational Decision Management accomplishes this feat, we need to agree on some nomenclature.

Every organization has people responsible for setting the policies by which they do business. In this context, a *business policy* is a statement of guidelines governing business decisions. An insurer may have an underwriting policy, for example, that says *"an 'underage applicant' for insurance on high-powered sports cars is not eligible for coverage."*

A policy statement like this is not sufficient to form the basis of an automated decision. Someone has to translate the policy into more specific statements that specify the details of how the policy is enforced. We call that a person a policy manager. In this insurance example, the policy manager is a subject matter expert for the automobile underwriting domain.

The specific statements that enforce the policy are *business rules*. Business rules are translations of the policies into detailed conditions and actions that unambiguously enforce decision outputs. For this to be possible, we start with an understanding of the data, interactions and terminology included in the specific domain of the business policy. The understanding of this information will lead to the definition of a vocabulary that will be used in writing of the rules. These rules will be used by various business systems that require them through an object model, which is developed by IT and mapped to specific data sources within the software infrastructure; but to the policy manager, their interaction with the object model will be through the non-technical vocabulary, allowing them to author and maintain rules using a business syntax which is described in more detail later this paper.

Business rules expand upon policies, by stating the detailed circumstances under which it is applicable and the actions that enforce it. One policy can translate into many business rules. In the insurance underwriting policy described above, for example, business rules need to define the terms of the policy (what is an "underage applicant and what is a "high-powered sports car") and probably also need to specify variants of the policy enforcement. Different regional regulations may require that "underage" be defined differently from one location to another, and the notion of "high-powered sports car" may change over time, as underwriting experience shows specific automobiles to incur more or less underwriting risk.

Another concept relating to policies is *business events*, which focus on occurrences of significance in a process or transaction that result in a change of state (e.g. in the auto insurance scenario, the submission of an application into the underwriting system, or the change of an application from "pending" to "accepted"). Definitions describing patterns of interest across multiple business events can be defined to enforce business policies or other organizational objectives. These definitions also take the form of conditions and actions, although the conditions tend to be based on the aggregation of multiple occurrences or correlating event patterns that span a period of time (compared to business rules which tend to be based on an individual occurrence at a specific point in time). Since both the definition of business rules and business event patterns use a common condition-action form, we will refer to both as "rules" in this paper, and use the terms *business rules* and *event rules* when there is a need to distinguish between them.

Even one policy domain, such as personal auto insurance underwriting in our example, can require hundreds or thousands of rules, frequently changing over time, and differing throughout jurisdictions (or customers, products, channels or any other partition of a business policy domain). We call the process by which a company manages and governs changes to policies, the *rule life cycle*.

WODM provides policy managers with tools to efficiently define rules and manage change through the rule life cycle. Specific applications are built using the technical tooling that is part of the offering. These applications execute data against rules and output the intended decision action to business processes and systems. For the insurance underwriting scenario, the underwriting application might invoke business rules to make a decision between rejecting an application, accepting it automatically, or referring it to a human underwriter for special consideration; or it may utilize event rules to determine if an

excessive number of a certain type of applications have exceeded a defined threshold (for instance, a number of applications for a type of high-risk vehicles during a specified time period), requiring a special review. In either case, Decision Center provides business users with the ability to participate directly in controlling the definition and governance of rules that enforce their business policies.

How WebSphere Operational Decision Management benefits policy managers

From a policy manager's perspective, the benefits of WODM stem directly from its ability to accomplish three separate things:

1. Implement rule changes separately from the application development life cycle
2. Author rules in business language
3. Manage rules throughout their life cycle, from creation through testing, deployment and retirement

Each of these is explained in detail in the following sections.

Implementing rule changes separately from the application development life cycle

Software is developed and deployed through a cycle of activities that is driven partly by business requirements, but also by technical and engineering demands (such as product upgrades or changes in other software systems with which it must integrate) not directly related to those requirements. The rule life cycle, on the other hand, can be driven by a variety of non-technology demands. For example:

- **Market dynamics:** Successful businesses need to be able to change their policies in response to market demands, economic conditions or competitive actions.
- **Regulatory change:** In financial services, insurance, and other highly regulated industries, rule changes can be demanded by the requirements of regulatory agencies, or by changes from legislation and court rulings. Government agencies administering benefit programs or complex laws must rapidly implement policy changes dictated by executive fiat or legislative acts.
- **Customization and personalization:** More and more businesses are enabling their applications to behave differently for different classes of customers or for individual customers. This may involve the same basic application that applies different rules based upon each customer's contract terms, or it may involve much deeper customization of entire functions for each individual customer. Third-party logistics providers, for example, are experts in the art of moving material around the world. They provide these services to other businesses, but the provision of these services will be "tuned" differently for each customer: an automobile manufacturer with a just-in-time manufacturing philosophy will want its service levels skewed toward absolute predictability and reliability, over and above cost, whereas a discount merchandiser may wish to accept to delays in favor of cost savings. The third-party logistics provider customizes its prices and service levels for each of these customers quite differently.

All these demands have one thing in common: they require a much higher degree of flexibility and responsiveness in translating business policies into actionable rules. The result is that it can be very difficult to align the rapidly evolving rule life cycle to a traditional software release schedule. A rule-based application mitigates this problem, by separating the cycles for software development and rule management, each with its own demands and timing.

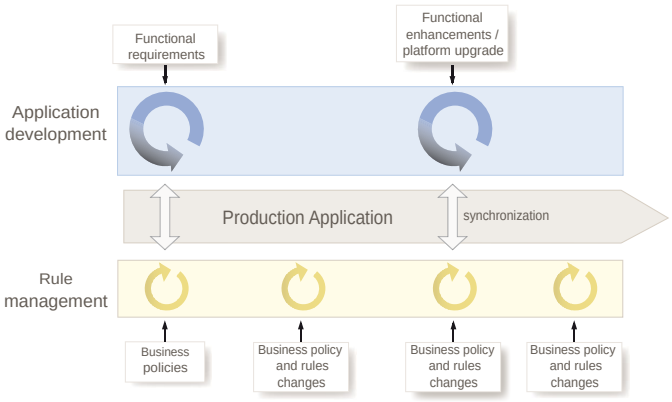

Figure 1: Separating the rule and application development cycles

Note that the application development cycle, represented at the top of the diagram, is driven over fairly long timeframes, by major functional enhancements or platform upgrades. New software releases are major engineering efforts, characterized by traditional design, construction and testing methodologies, and which are relatively infrequent. Policy changes on the other hand, represented in the bottom half of the diagram, are driven by business requirements which necessitate rule changes. Changes to these business requirements may be very frequent (e.g., in creating cross-sell or promotion offerings in an online store), or more periodic (as in the case of negotiated contract terms for a specific customer), but are not, in either case, on the same schedule as changes to the overall application. Using WODM, so long as the rule changes represent variations of decision-making based on data already managed by the application, they can proceed according to the business demands of the policy maker, not the technology demands of the application development process.

Author rules in business language

Separating the rule life cycle from the application development cycle enables more timely response to business change. However, to also empower the policy manager to enact these changes directly, the software technology has to provide another important capability: a means of expressing business rules in a format and language that is familiar to the policy managers, and easily manipulated by them.

There are really two elements to meeting this requirement:

- Specification of a business language that can unambiguously express the policy manager's intent, yet remain familiar and easy to manipulate.
- Translation of this language into something "executable" by the software application.

Business action language

In many cases, these two elements can be met by means of a straightforward mapping between business vocabulary and technical software artifacts.

Software developers typically express their understanding of the information required to run business systems in terms of an object model. An object model is a formal means for expressing concepts (such as Customer, Order, and Item, for example), their relationships (a Customer can have multiple Orders, but an Order belongs to exactly one Customer), their content (a Customer is described by a Name, Street Address, City, State and a Customer Class—such as Platinum, Gold, or Silver), and the operations that can be performed on them (e.g., providing a discount or adding a message to an Order). The object model defines how a software program manipulates data, and WebSphere Operational Decision Management allows all the terms in the object model to be translated into a vocabulary that

policy managers can use when writing rules. Combined with a simple if-then syntax (IF some set of conditions are true THEN a certain action should be taken), the policy manager can express rules in statements that are still in business vocabulary but are precise and automatable. We call this combination of a natural language syntax which uses a domain-specific business vocabulary a *business action language*.

The object model discussed above might be verbalized by a vocabulary as shown in the following table.

Object model concept, data or operation	Vocabulary
Customer	The customer
Name	The customer's name
State	The customer's state
CustomerClass	The customer's priority level
Order	The order
Discount	The order discount
Amount	The order amount
Date	The order date
AddMessage	Add the message <message> to the order
Item	The item
SKU	The catalog number
Quantity	The item quantity
Price	The item price

Table 1: Mapping the business object model to a vocabulary

Now suppose that a policy manager needs to enforce a rule that says that all "gold" customers in Minnesota who place an order in January receive a 10 percent discount. The policy manager would write a business rule like that in Figure 2.

```
IF
   the customer's state is "MN" and
   the customer's priority level is "Gold" and
   the order date is between "January 1, 2010" and "January 31, 2010"
THEN
   set the order discount to 10 percent and
   add the message "As a gold customer you receive a
   10 percent discount on today's order." to the order
```

Figure 2: A rule in a business action language for computing a discount

Note the generic if-then structure of the rule, the use of the established domain-specific vocabulary (mapped to the object model) to define the rule's conditions and actions, and the easily readable, though precise, language of the rule.

Now, consider a rule example which is based on a pattern of business events. In this example the object model will contain two additional objects that are associated to the Order called CustomerCart and CartState, which will be mapped to the vocabulary terms "an order cart" and "the status of the order's cart".

```
IF
    the customer has an order cart and
    the status of the order's cart is "check-out" and
    the customer has an order cart within the previous 30 days and
    the status of the order's cart is "abandoned"
THEN
    add the message "Use code 'Free123' for free shipping on any order over
    $15 in the next 7 days." to the order.
```

Figure 3: An event rule in a business action language

Again, the conditions and actions of the rule can be easily understood by the non-technical business user, but in this case the condition is using comparing multiple order occurrences to make a determination.

Decision tables and decision trees

Sometimes it makes sense to express a business rule as an if-then statement, but often it is more useful to express a set of business rules that involve the same conditional terms in either a table or tree form. A *decision table* is defined by the set of conditional terms as column headers with each row representing a single rule and the result of each rule in the final column.

Building on the order example above, a decision table like that in Table 2 might be used to determine the customer's priority level for shipping product based on their total order amount and state.

	If	**and**	**Then**
	the customer's total order value	**the customer's state**	**the customer's priority level**
1	Between $0 and $500	Any	Bronze
2	Between $500 and $2000	NY, CA	Gold
3	Between $500 and $2000	[not] NY, CA	Silver
4	Between $2000 and $2500	Any	Gold
5	Greater than $2500	Any	Platinum

Table 2: Decision table for customer class

A *decision tree* also combines a set of business rules into one graphical display, but rather than each rule being a row, each rule is a path through the tree, branching at each conditional term. The density of the tree depends on how many possible values there are for each condition—with a few (e.g. true/false), a tree is a good display choice; with many (e.g. each of 50 US States), a table is often more readable. A decision tree is also a good way of expressing related rules with asymmetric outcomes—for example, if some types of customers have more levels of classification than others.

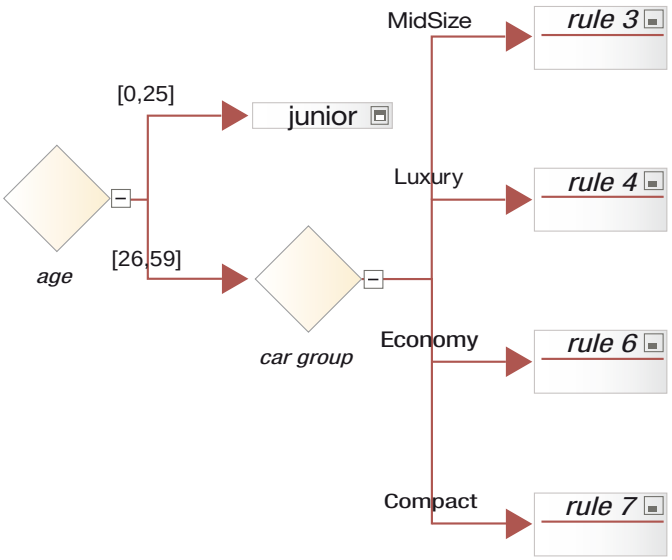

Figure 4: A ruleset in a decision tree format

Both decision tables and decision trees are extremely useful for viewing multiple related rules at once, to ensure that you have addressed all scenarios and that the rules makes business sense as a group. In the current release, only business rules can be expressed in the decision table or decision tree formats.

Manage rules throughout their life cycle, from creation through testing, deployment and retirement

The ability to quickly identify which rules need to change due to a change in business policy, simulate proposed changes prior to going into production and then implement rule changes in the timeframe the business needs is a competitive advantage for organizations. Rules define how the business operates and are valuable assets that should be managed accordingly. This means:

- Organizing rules in a way that is logical to the policy manager
- Making rules accessible to the policy manager and searchable by relevant criteria
- Versioning rules and maintaining an audit trail
- Analyzing rules for consistency and completeness
- Testing the implementation of rules to ensure it is faithful to the business intent
- Simulating rule changes to understand the business impact before implementation

Rule organization

A typical enterprise has many policy managers who are responsible for thousands of rules. To make this efficient, WebSphere Operational Decision Management provides simple, transparent capabilities for organizing rules. Rule organization takes place in the Decision Center repository, providing a centralized environment to manage all rules across various decision management projects. Typically, rules are grouped by high-level content areas which are then broken down further by tasks or decisions. For example, a property and casualty insurance company might start with lines of business (Fire, Auto, Property, etc.) and then have Rating Rules, Pricing Rules and Claim Processing Rules under each. Business rules and event rules are grouped separately in their own projects, but both can be easily navigated within the Decision Center Console, which is the primary interface for accessing the repository.

Search

To effectively manage policies implemented as large numbers of rules, the Decision Center Console provides the ability to search the rule repository for specific rules, based upon criteria other than organizational structure of the repository. WODM also enables the assignment of additional metadata properties such as who authored the rule, when the rule was last modified or the effective date range for the implementation of the rule. It may also be necessary to search by more application-specific criteria such as the geographic location or jurisdiction to which the rule applies. These properties are in addition to the ability to search based on the vocabulary terms used in rules and the actions taken as results.

The more accessible the information, the more effectively business people throughout the enterprise can use and build upon the rules already written. For example, a regulatory compliance officer for an insurance company using the physical organization and metadata mentioned above, could collect and inspect

all the rules for California in effect for a given date, throughout all lines of business and coverage types, by issuing a query for those specific rules.

Versioning and auditing

One key requirement of policy management is the ability to track individual rules and sets of rules as they change over time and to verify the definition of a rule as of any given transaction. WODM facilitates this type of audit, by keeping each version of a rule in the rule repository and tracking when each version was deployed and active. The versioning system may also track the identity of each individual who changed a rule, as well as require annotation of the changes. In this way, an automated audit trail is created for tracking policy changes in the organization.

Analysis of rules

Business policies can be quite complex—the centralized management and formalized expression of the rules that implement a policy enables more systematic analysis of dependencies, consistency and faithfulness to the business intent. Having rules in a formal syntax, WODM can highlight where the result of an action in one rule is used in the condition part of another rule—since the second is dependent on the first, you know changing the first will impact the second. If you have long dependency chains, the ability to highlight the dependencies further down the chain is crucial to understanding the total impact of changes. WODM can also identify inconsistencies and redundancies in your rules:

- Whether a rule is inconsistent (because it specifies conditions that cannot be true, for example).
- Whether a rule conflicts with another rule.
- Whether a rule is redundant (because it is equivalent to another rule, for example).
- Whether a set of rules has gaps in the scenarios addressed

Finally, having rules organized and formally expressed enables the policy manager to review sets of rules to ensure they really do result in the appropriate set of actions. This determination cannot be fully automated for the policy manager, but the information is presented in a manageable way, along with the tools necessary to help the business user to validate rules meet their requirements.

Testing

Policy managers authoring rules using Decision Center have tools that permit them to verify that the rules they write actually work as required. In its simplest form, this means capabilities to answer the question "if I run my rule on these specific values, will it fire and produce the expected results?" This is "unit testing" of rules.

It is also necessary to test rules in aggregation, to determine if, collectively, they implement fully and correctly the intended policy. This requires the ability to specify a complete rule set and a complete set of input data and again verify that the results are as expected. This is "functional testing" of rules.

Finally, because even with constantly evolving policies, businesses require stability in most of their functions, there needs to be a mechanism for verifying that a proposed new rule set satisfies standing constraints on its functionality. This may require running many complete input data scenarios against a candidate set of rules and evaluating the aggregate results. This is "regression testing" of rules.

All three testing types are supported, providing confidence that changes will meet the stated policy requirements and assurance that decision management applications will function properly when they are deployed into the production environment.

Simulation

One benefit of WebSphere Operational Decision Management is that, because policies are given a precise operational form, as rules, a proposed set of related rules (a ruleset) can be evaluated rigorously to determine if it yields the desired results. One version of that is the regression testing of rules mentioned above, but it is also possible to do more business focused verification of rules.

Suppose, for example, an insurer's underwriting goal is to accept 80 percent of tendered risks, without exceeding a specified aggregate exposure index. WODM's "simulation" capability facilitates this as follows: underwriting managers define detailed policies with the intent of meeting this goal and translate these into a "candidate" rule set. This candidate rule set is then be run against all insurance policy applications received in a given time period (for instance, the previous six months), with the results of each evaluation being recorded. Finally, key performance indicators, including the percentage of yield and the aggregate exposure index would be computed on the results, permitting the underwriting manager to discern whether the particular policies written into the rules would have met the overall business goal.

What to look for in a Decision Management Solution

From the perspective of a policy manager or line-of-business subject matter expert, evaluating potential solutions can be a daunting task: a decision management solution delivers its benefits by enhancing the function of enterprise applications, not by replacing them. As a result, because of the shared responsibilities between policy managers and software developers, both perspectives need to be accommodated when evaluating different software offerings. The important features to look for include:

- **Addresses the needs of business and IT:** A solution should provide an environment and tools designed specifically for policy managers that isolate them from the concerns of application developers and have at the same time, an environment and tools that support and simplify the developers' task of integrating with the business systems that need to interact with rule-based applications.
- **Provides domain-appropriate rule artifacts:** As discussed above, there are different ways to express a business rule. A solution should support those that are most natural for a given business, or provide a means to extend and alter its default rule languages to accommodate the business' specific needs.
- **Address the need to manage the entire rule life cycle:** Typically, a successful rule application will eventually encompass hundreds to perhaps tens of thousands of rules, which will undergo constant modification and extension as policies change. To be fully effective, a solution must support the policy manager at every step in the rule life cycle, in order to make large collections of rules intelligible, verifiable and manageable.

Conclusion

Managers responsible for policy implementation in decision-rich enterprises have benefited enormously from information technology. But as the pace of change has increased and the reach of automation has been extended, traditional software applications have not provided the flexibility and agility these managers require. Lines-of-business and IT need to eliminate the traditional communication and collaboration barriers that limit an organization's ability to effectively respond to changing demands from the market, competitors and regulators.

Applications constructed with WebSphere Operational Decision Management can deliver flexibility to business systems and agility to the organization by enhancing the ability to automate decisions with a high degree of precision and personalization. And they provide a set of integrated environments specifically designed to meet the needs of various business and technical users who participate together in the creation, deployment and ongoing maintenance of business rules.

For more information

To learn more about the IBM WebSphere Operational Decision Management, please contact your IBM marketing representative or IBM Business Partner, or visit the following website: **ibm.com**/decision-management

Additionally, financing solutions from IBM Global Financing can enable effective cash management, protection from technology obsolescence, improved total cost of ownership and return on investment. Also, our Global Asset Recovery Services help address environmental concerns with new, more energy-efficient solutions. For more information on IBM Global Financing, visit: **ibm.com**/financing

Highlights
Implementing decision logic with IBM WebSphere Operational Decision Management (WODM) enables a giant leap forward in bridging the gap between business people and IT.

WODM provides a suite of tools which policy managers and software engineers use to build flexible applications in which decision logic is abstracted out of software code.

Business rules are translations of the policies into detailed conditions and actions that unambiguously enforce decision outputs.

Business events focus on occurrences of significance in a process or transaction that result in a change of state.

Both the definition of business rules and business event patterns use a common condition-action form; collectively these are referred to as "rules" in this paper, and the terms business rules and event rules are used when there is a need to distinguish between them.

WODM provides policy managers with capabilities to efficiently define and manage rules through the rule life cycle (WebSphere Decision Center).

WODM has to provide another important capability to enable more timely response to business change: a means of expressing rules in a format and language that is familiar to the policy managers, and easily manipulated by them.

A typical enterprise has many policy managers who are responsible for thousands of rules. To make this efficient, the Decision Center Console provides simple, transparent capabilities for organizing rules.

WODM also enables the assignment of additional metadata properties such as who authored the rule, when the rule was last modified or the effective date range for the implementation of the rule.

Decision Center provides the tools necessary to help the business user to validate rules meet their requirements.

One benefit of WODM is that, because policies are given a precise operational form, as rules, a proposed set of related rules (a ruleset) can be evaluated rigorously to determine if it yields the desired results.

WebSphere Operational Decision Management delivers its benefits by enhancing the function of enterprise applications, not by replacing them.

About the author

Brett Stineman
Product Marketing, Application and Integration Middleware
Software, IBM Software Group

IBM Corporation
Software Group
Route 100
Somers, NY 10589 U.S.A.

Produced in the United States of America
October 2011

Please Recycle

WSW14061-USEN-02