BizTalk RFID

Objectives

Why RFID?

BizTalk RFID Architecture

Processes

Devices/Device Groups

EventHandlers

Management

The promise of RFID

- **IP addresses for entities in the physical world**
  - Enabling an internet for the physical world
  - rfid://urn:epc:tag:grai-96.3.0067890.1234.567879
- **Hardware tags are the lowest layer**
  - Like IP addresses
- **ROI realized in software**
  - Web based lower level protocols like IP
- **RFID is the poster child for enterprise connectivity**
  - Many scenarios require cross-enterprise information sharing

The pain of RFID

- **Hardware adoption blockers**
  - Regulatory issues for multi-region tags
  - Reader reliability
  - Device management challenges
- **Software adoption blockers**
  - Reader heterogeneity
  - Edge deployments
  - Privacy / security concerns
  - Mission critical
- **Hardware / software automation**
  - The challenge of real-time response

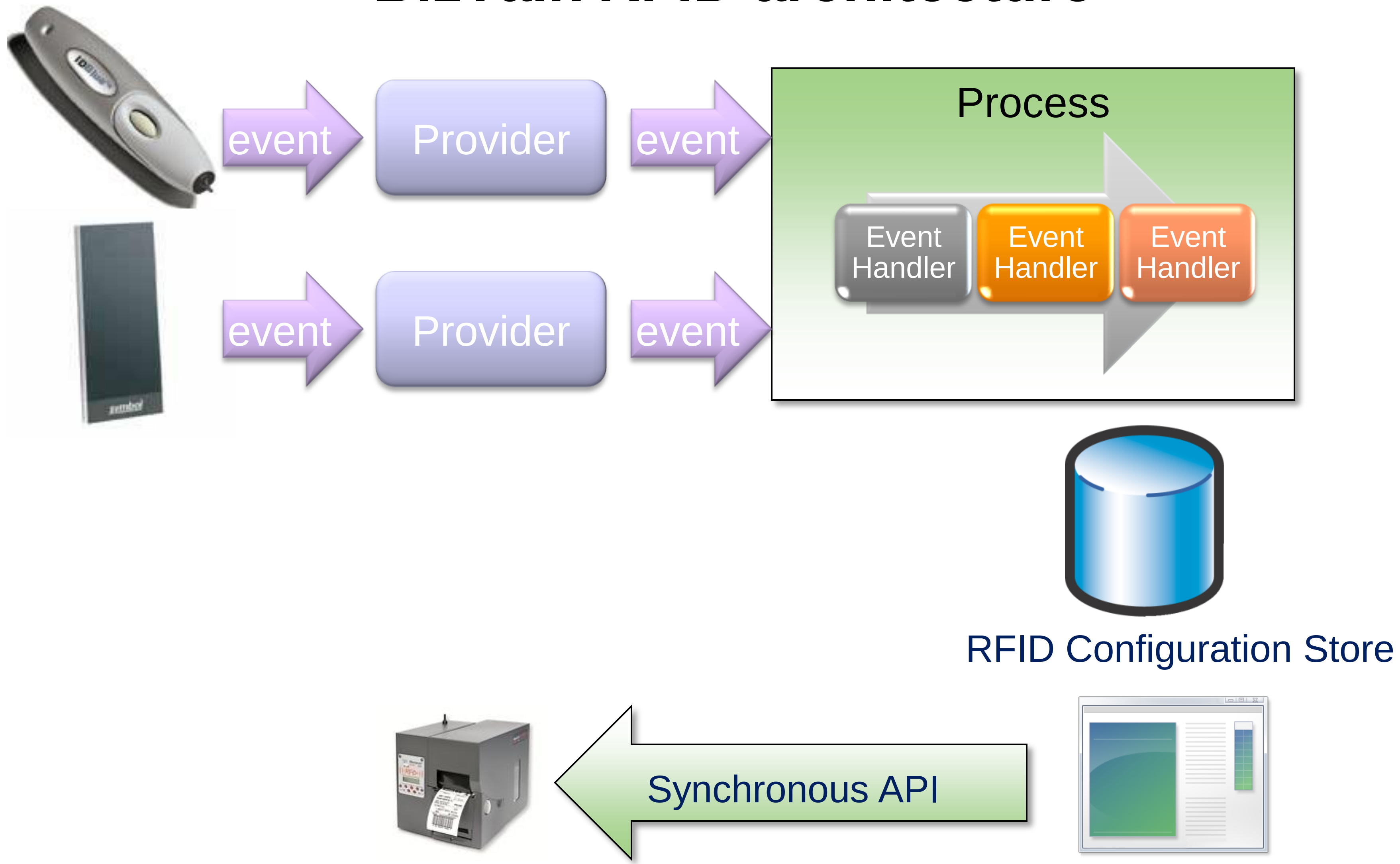
Customer scenarios

- Supply chain visibility
- High value asset tracking
- Document tracking
- Maintenance alerts
- Product verification [theft and returns]
- Cattle tracking
- …

BizTalk RFID

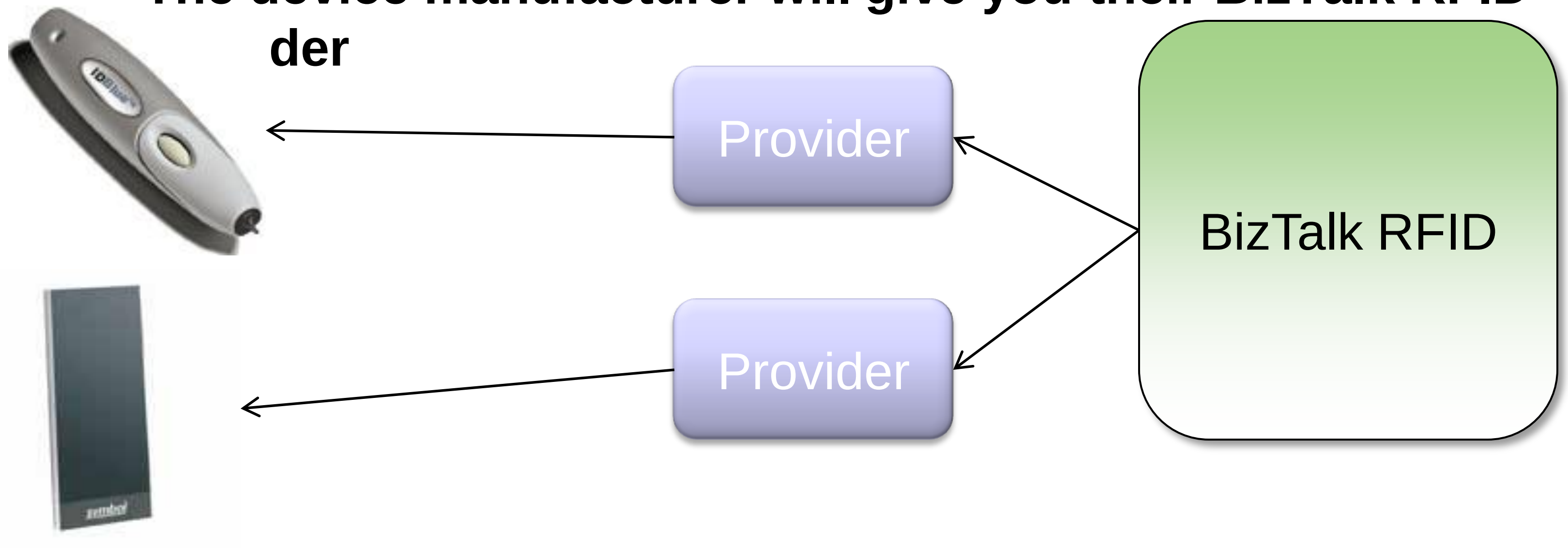
- **Enable applications to work with a rich set of devices in a hardware agnostic fashion**
- **Server and application services for interacting with devices and tag reads**
- **Management tools for devices and RFID business processes**
- **APIs for interacting with devices**
  - Synchronous API for device control
  - Asynchronous API for process creation and event handling

BizTalk RFID architecture



Provider

- **An assembly that each device manufacturer writes that plugs into BizTalk RFID**
  - Similar to the ADO.NET model for databases
  - Encapsulates the differences in device APIs away from BizTalk RFID and your code
- **The device manufacturer will give you their BizTalk RFID**
  der

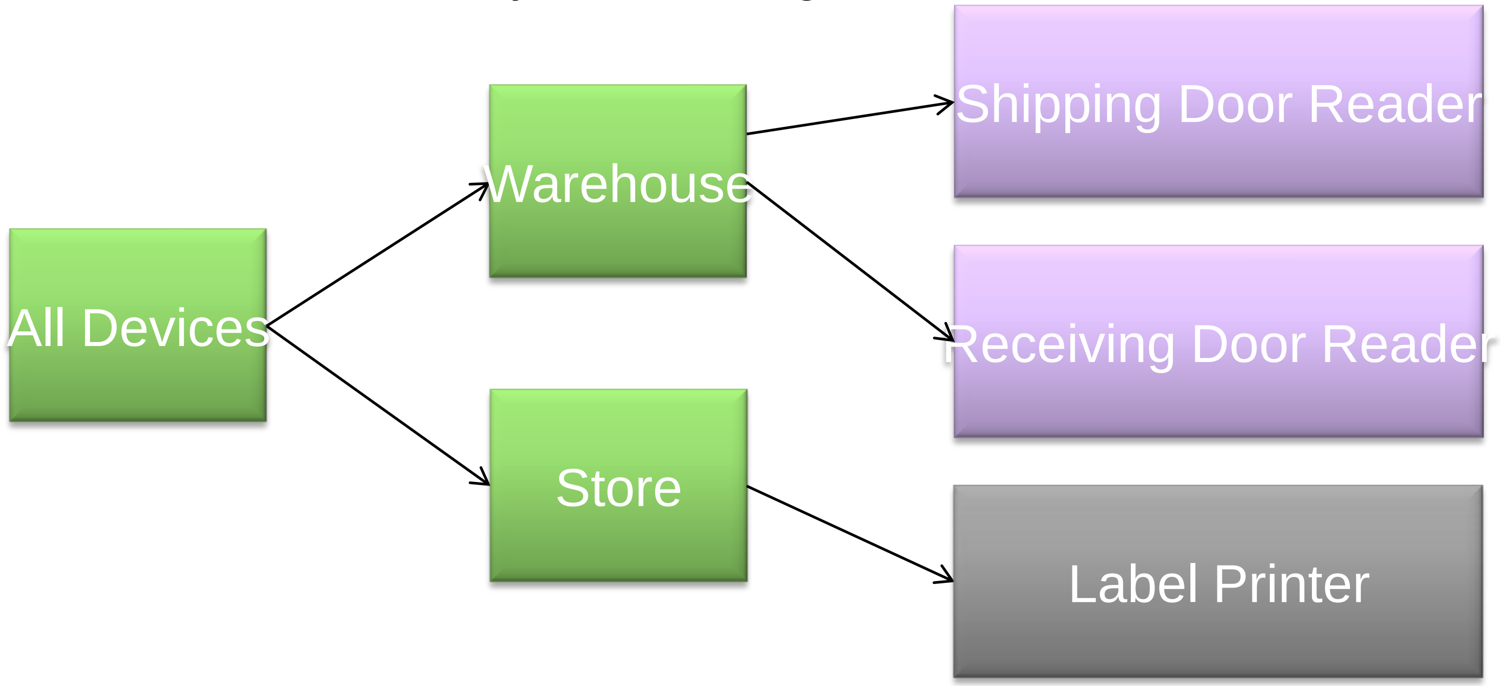


Devices

- **A Device is a named physical RFID device**
  - One to one mapping between a device in BizTalk RFID and a physical device in the real world
- **Readers – readers broadcast RF signals and get back RFID tag reads**
  - Can be small (handheld)
  - Can be large (antennas next to warehouse doors
  - Some readers can also write data
- **Printers – printers write data to RFID tags**
  - Generally prints label (based on template), barcode, and to the RFID tag in a label

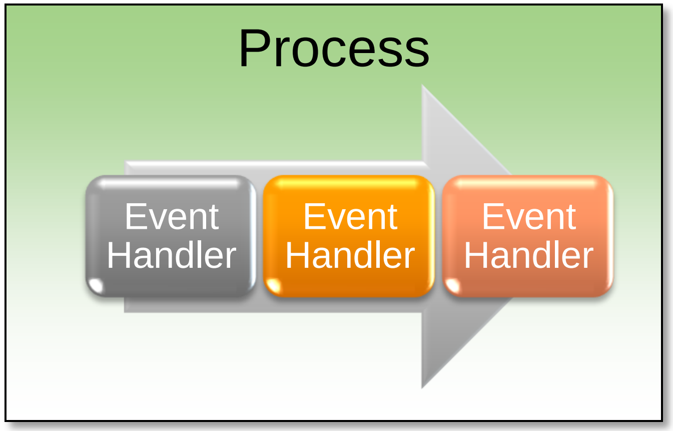
Device Groups

- **Some RFID installations might need to manage hundreds or thousands of devices**
- **Enables organization of devices**
  - Like files on a file system with organized folders



Process

- **A container in BizTalk RFID for mapping devices to software**
- **Maps a logical device to a set of EventHandlers**
- **Logical device enables**
  - Device aggregation
  - Device splitting (multiple antennas)
- **EventHandlers are .NET components**
  - Derive from well-known base class
  - OOB EventHandlers
  - Custom EventHandlers possible (likely?)



Steps to build an RFID Async Application

- **Install BizTalk RFID**
- **Install and configure your provider**
- **Configure a device or device group**
- **Create one or more logical processes**
  - For each process configure the necessary Event Handler components
- **Functionality exposed via MMC**
  - Can script via command-line tools or API

Configuring a provider

- **Right-click on the Device Providers node in the MMC**
  - "New Provider" context menu item
- **Browse to the provider's assembly**
  - Click register
- **Press Ok**
  - Option to start provider
  - Option to add device
- **Starting provider doesn't connect to device(s)**
  - Just configures provider process and opens communication between provider and RFID Service
- **Right-click on provider to discover device(s)**
  - Installation may automate this step as well

Most device manufacturer's provider installation will do this automatically

Creating a device

- **Right-click on Devices node in MMC and select “New Device”**
- **Pick provider**
- **Specify connection**
  - Provider specific
  - Defines connection between device and provider
- **Add Device Groups (optional)**
- **Authentication (options)**
- **Specify properties on device**
  - Device must be open to set properties

Creating a Process

- **Specify name**
  - Must be unique
- **Specify processing mode**
  - Transactional (default) – transactional guarantee if processing fails message goes back into the queue
  - Express (no guarantees)
  - Reliable (same as express)
- **Optional description**

Binding the process

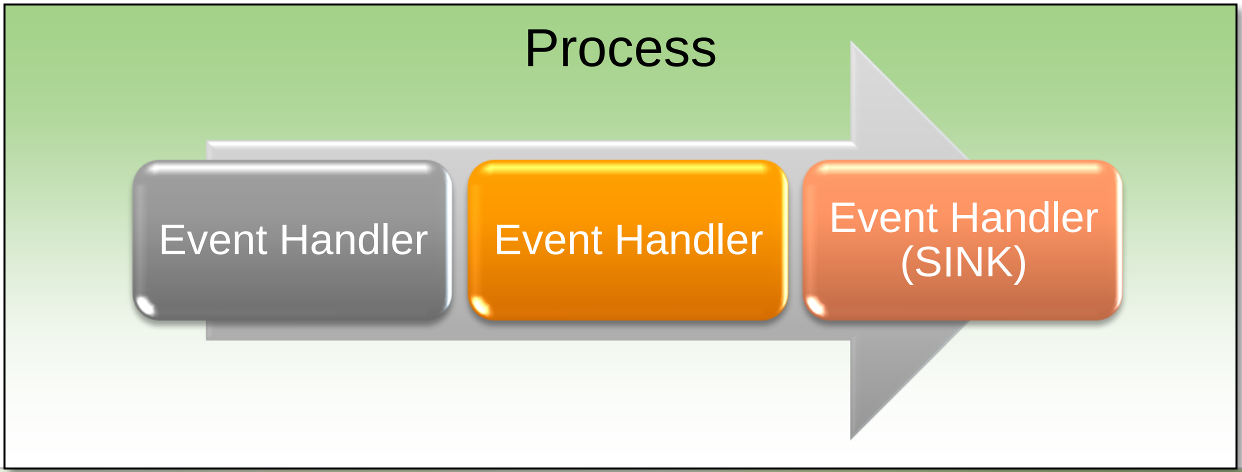
- **The action of mapping a process to devices and components**
- **Step 1 - Create a logical device**
  - Can aggregate devices
  - Can “split” devices
  - Regex can be used to simplify
- **Step 2 - Configure components**
  - Three OOB : RuleEnginePolicyExecutor, SqlServerSink, SqlServerSinkWithChronologicalReads
- **Step 3 – Validate the Process**
- **Step 4 – Start the Process**

EventHandlers

- **An essential part of each RFID process is its EventHandlers**
  - Provide all the application specific functionality
- **EventHandlers process tag events**
- **Can provide multiple types of functionality**
  - Filtering
  - Enrichment
  - Transformation
  - Processing
- **Each Process must have at least one EventHandler component**
- **Each Process can have *N* EventHandlers configured**
  - Each handler must be cooperative

Terminating Sinks

- **Some EventHandlers are “terminating”**
- **EventHandler indicates this through its method signature**
  - void return value advertises that it is a Sink
- **Sink must be the last EventHandler in a Process**
  - Optional – not every process has to have a Sink
  - Sink must be the last component
  - Only one Sink allowed per Process



RuleEnginePolicyExecutor

- **One of OOB EventHandler components**
  - Integrates with BizTalk Business Rule Engine (BRE)
- **Component is configured with a named deployed Policy**
  - Extra facts can also be configured
  - Specific Policy versions can be specified
- **Policy can then implement the EventHandler processing logic**
- **RfidRuleEngineContext object is always an available fact**
  - Contains properties and methods to control and introspect on the RFID events

SqlServerSink

- **One of the OOB EventHandler Components**
- **Pumps tag event data into a SQL Server database**
- **SqlServerSinkWithChronologicalReads is related component**
  - Separate but related component
  - Writes in chronological order
- **Common use cases:**
  - Custom reporting on top of database
  - SSIS package to push data into other database
  - Configure BizTalk Server SQL Server Adapter to poll and send data into BizTalk Server

Creating a custom Event Handler

- **Create a class library project**
- **Add a reference to**
  - Microsoft.Rfid.Design.dll
  - Microsoft.Rfid.SpiSdk.dll
- **Create class that derives from RfidEventHandlerBase**
  - Override the Init method
- **Must have parameter-less constructor**
- **Two static methods to be added**
  - GetEventHandlerMetadata (required)
  - CreateDefaultEventHandlerDefinition (optional)
- **Add appropriate methods for getting events**
  - These must be annotated with RfidEventHandlerMethod attribute

EventHandler Statics

```
//returns Metadata for EventHandler configuration
public static RfidEventHandlerMetadata GetEventHandlerMetadata(bool
vendorEvents)
{
    Dictionary<string, RfidEventHandlerParameterMetadata> parms = new
        Dictionary<string, RfidEventHandlerParameterMetadata>();
    parms.Add("MyProperty",
        new RfidEventHandlerParameterMetadata(typeof(string),
                                    "A Description for the property","",false);
    RfidEventHandlerMetadata md =
                            new RfidEventHandlerMetadata("DebugSink", parms);
    return md;
}


//metadata stored by the RFID process runtime
public static EventHandlerDefinition CreateDefaultEventHandlerDefinition()
{
    Type myType = typeof(DebugSink);
    return new EventHandlerDefinition(myType.FullName,
    new EventHandlerInfo(myType.Assembly.FullName, myType.FullName));
}
```

RfidEventHandlerMethod

- **Attribute can be applied to one or two methods**
- **Methods must take a single RfidEventBase or an array of RfidEventBase**
  - Derived types acceptable
- **Methods must return void, a single RfidEventBase or an array of RfidEventBase**
  - Derived types acceptable
- **Method signatures dictate functionality**
  - void return creates “terminating” EventHandler (Sink)
- **If process was created as transactional call to EventHandlerMethod will have an ambient transaction available**

EventHandlerMethod choices

```
[RfidEventHandlerMethod]
public void TagEventTerminating(RfidEventBase tagEvent)
{
}
[RfidEventHandlerMethod]
public void TagEventTerminating(RfidEventBase[] tagEvents)
{
}
[RfidEventHandlerMethod]
public RfidEventBase[] TagEvent(RfidEventBase[] tagEvents)
{

}
[RfidEventHandlerMethod]
public RfidEventBase[] TagEvent(RfidEventBase tagEvent)
{
}
```

Synchronous Applications

- **EventHandlers and Processes are known as Async RFID Applications**
- **Object model is define that can be used to create Sync Applications**
  - Turning device on or off
  - Printing
  - Tag control (kill)
- **Aysnc and Sync models can be combined**
  - Async event - sync model to be used to turn device off
  - Async event - send tag kill command

Summary

- The RFID industry is reaching maturity of scale
- BizTalk RFID v1 aimed squarely at enterprise scenarios (Mission Critical RFID)
- Providers provide an abstraction layer on top of hardware
- Processes enable pluggable event handlers to be combined together to provide software solution on top of RFID hardware