Eric Edeen

IBM Content Manager Storage Team

October 22, 2014

Product Implementation Training (PIT)

FileNet Content Manager 5.2.1

Custom Sweep Support

Introduction

- Course Overview
  - This course will describe what a custom sweep is and how to write one
- Target Audience
  - Anyone designing, authoring or supporting Content Engine Custom Sweep handlers
- Suggested Prerequisites
  - Familiarity with IBM FileNet Content Platform Engine Sweep Framework
  - Familiarity with IBM FileNet Content Platform API
  - Familiarity with writing Java or Java Script code
- Version Release Date October, 2014

Course Objectives

After this course you will be able to:

- Describe a Sweep in conceptual terms
- Identify the 3 basic types of Custom Sweeps and the type of use cases each supports
- Identify the basic components of a Custom Sweep
- Describe the SweepActionHandler interface
- Describe what the onSweep method is for and the general pattern for implementing it
- Describe how to use the SweepItem interface in a custom handler to retrieve target objects and to return outcomes

Course Roadmap

- ➔ Feature Description
- Sweep Framework Overview
- Basic Components of a Sweep
- Creating a Custom Sweep Handler
- Source Code Example
- Coding Tips
- Demonstration
- Security Considerations
- Course Summary

Feature Description

- Previous releases of the Content Engine provided a mechanism for extending server functionality by way of Action Handlers. Such Action Handlers include: Event Actions, Document Lifecycle Actions, Document Classifiers, and Change Preprocessers
- Custom Sweeps are a new kind of Action Handler available in the 5.2.1 release that provides a way to create Sweeps with user defined behavior
- Sweeps are user initiated background tasks that are invoked by the Sweep Framework in order to process large numbers of objects
- Three flavors of Custom Sweeps are supported:
  - Custom Sweep Jobs
  - Custom Sweep Policies
  - Custom Queue Sweeps

Course Roadmap

- Feature Description
- ➔ Sweep Framework Overview
- Basic Components of a Sweep
- Creating a Custom Sweep Handler
- Source Code Example
- Coding Tips
- Demonstration
- Security Considerations
- Course Summary

Sweep Framework Overview

- The 5.2 release introduced several new features that were all implemented using a new subsystem that we called the Sweep Framework
- Existing features based on the Sweep Framework include:
  - Disposal Policies for automatically disposing content that has expired
  - Retention Update Policies for automatically applying and updating retention dates on retainable objects
  - Bulk Move Content Jobs for moving content from one storage area to another for large numbers of documents or annotations

What is a ‘Sweep’?

- A sweep is a user initiated background process that examines a large set of candidate objects and applies an action to the subset that satisfies a rule
  - Sweeps are initiated when an user creates a Sweep object
  - The set of candidate objects is defined by the SweepTarget property, which points to a ClassDefinition object
  - The rule is defined by a FilterExpression property
  - For all OOTB sweeps the action is defined by the sweep type or the policy type that is associated with the sweep
  - Example: the action associated with a Disposal Policy is to dispose (delete) objects selected by the rule
  - Action is arbitrary from the sweep framework’s point of view
  - For Custom Sweeps, the action is defined by the user

Simplified Definition of a Sweep

- From a conceptual standpoint, a sweep could be described as doing the following:

```
objectSet = SELECT object FROM <TargetClass> WHERE <filter condition(s)> == TRUE
FOR EACH obj IN objectSet DO:
    APPLY Action(obj)
END
```

- We refer to each execution of the query and the loop, as a ‘Sweep Iteration’
- Some sweep types perform a single iteration, while other types perform repeated iterations

Types of Sweeps

- Sweeps come in three basic types :
  - Sweep Jobs
  - Policy Controlled Sweeps
  - Queue Sweeps
- Each of these basic types have sub-types that define the specific behavior or action performed by the sweep
- Each of these general types also support subclasses that allow user defined custom behavior
  - Custom Sweep Job
  - Custom Sweep Policy
  - Custom Queue Sweep

Sweep Jobs

- Perform a single Sweep Iteration
- Have a definite start and completion
  - Starts when first candidate object is visited
  - Ends when each candidate object is visited exactly one time (unless explicitly terminated earlier)
- Once executed, Sweep Jobs cannot be re-executed
- Sweep Jobs are useful for executing a one-time-only operation on many objects
- Example:
  - Bulk Move Content Job

Policy Controlled Sweeps

- Perform multiple Sweep Iterations
- Have a definite start - indefinite completion
  - Policy Controlled Sweeps continue to execute until they are either disabled or deleted
- A Policy Controlled Sweep is related to one or more Sweep Policies
- Sweep Policies define the action(s) that will be performed by a Policy Controlled Sweep
- Policy Controlled Sweeps cannot be created directly. They are indirectly created by creating Sweep Policy
- Policy Controlled Sweeps are useful for performing operations that need to be triggered by calendar based events
  - E.g. automatically dispose objects when their retention has expired
  - This cannot be done with Events Actions and Subscriptions because there is no object update to trigger the event

Queue Sweeps

- Although Queue Sweeps share much of their implementation with Job Sweeps and Policy Sweeps, what they do is conceptually very different.
- Queue sweeps provide a generic queuing service that can front any slow or resource intensive process
- Pizza delivery analogy
- Queue Sweeps only process Queue Items, that is, concrete instances of some subclass of Abstract Queue Item
- Queue Items are short lived objects whose only role is to carry a message to the Queue Sweep handler
- Once a Queue Item is “processed”, that is, once the message has been successfully delivered to the handler and the handler has responded with a success code, the Queue Item no longer serves any purpose so it is removed from the queue by deleting it.
- Other Sweep Actions typically perform arbitrary operations on the items being swept, but Queue Sweep Actions only read them and return a status
- Queue Sweeps always have their own dedicated database table

Course Roadmap

Sweep Handler Component

- The Sweep Handler is the executable code that implements the action
- Custom sweep handlers can be written in Java or Java Script
- Sweep Handlers are not Content Engine API objects, they are Java class files or Java Script files
- Sweep Handlers are referenced by Sweep Actions

Code Module Component

- A Code Module is an container for executable code that be stored in an object store
- Executable code stored as a Code Module can be loaded and executed by the Content Platform Engine server without modifying the class path
- Code Modules can be versioned just like documents
- Use of Code Modules for Sweep Handlers is optional (but highly recommended)

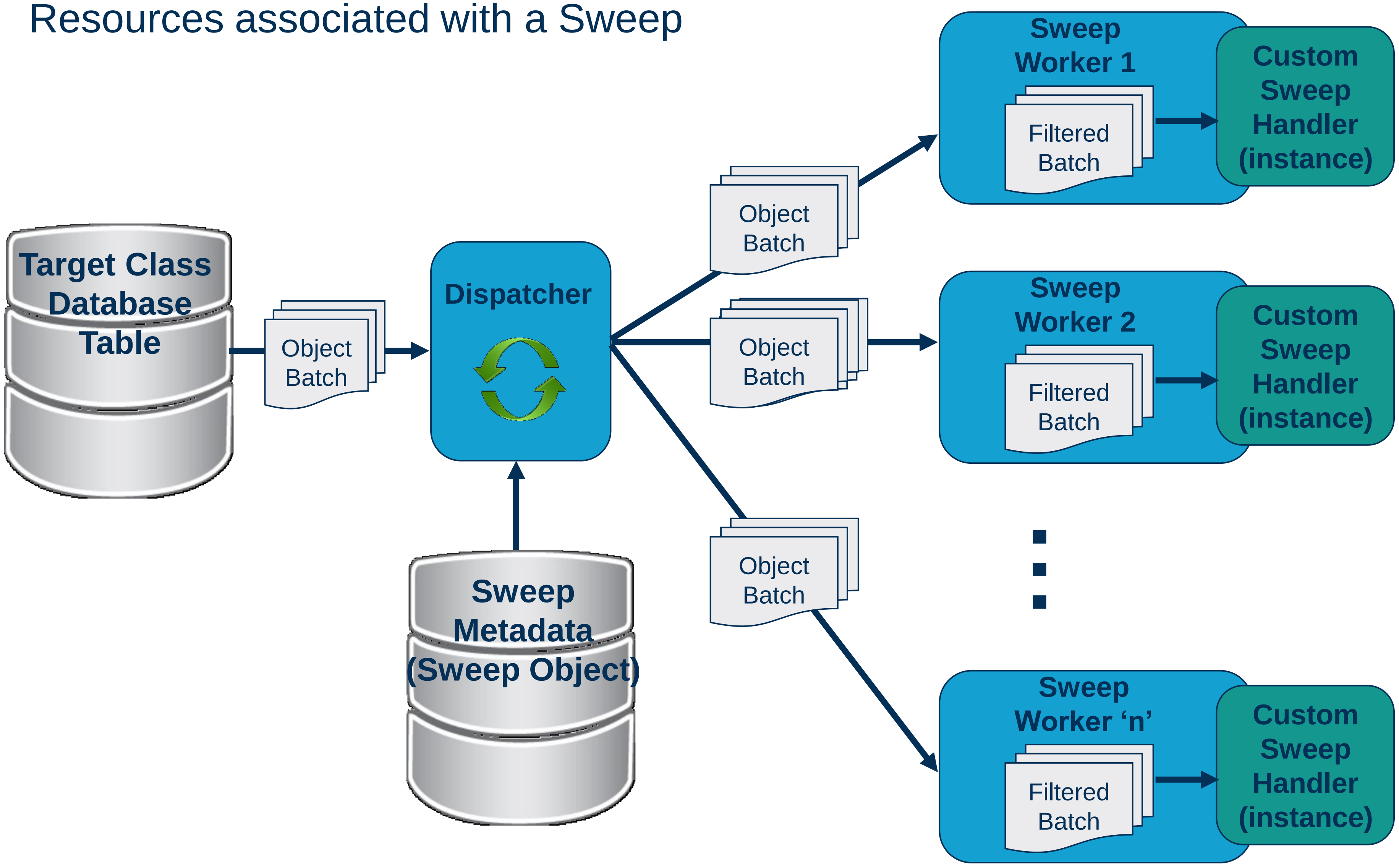
Sweep Action Component

- The Sweep Action component is the Content Engine API object that represents a Sweep Handler

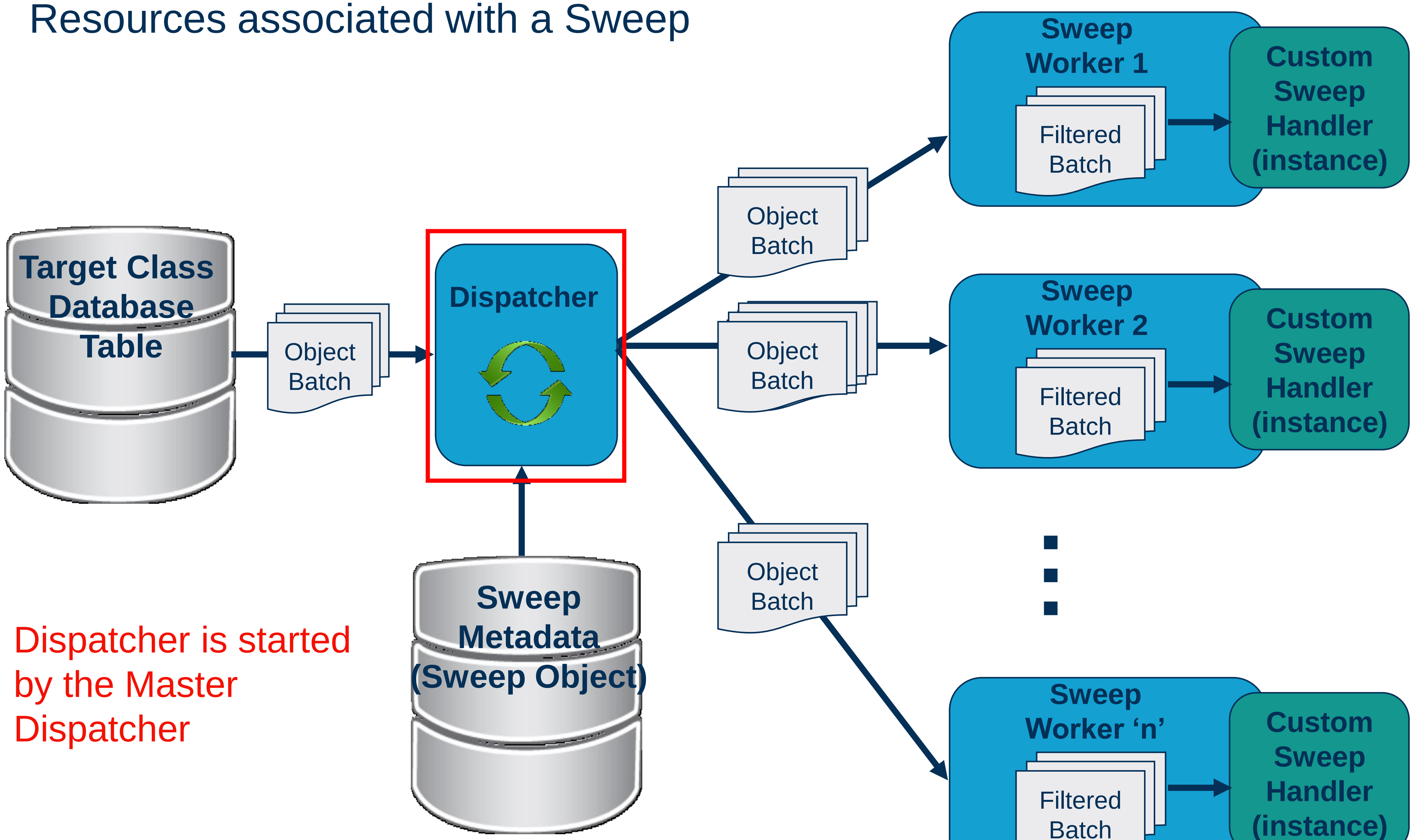
Sweep Object

- Sweep Objects are API objects that are instances of some subclass of the Sweep class. For Custom Sweeps, this will be an instance of either CmCustomSweepJob, CmCustomSweepPolicy, or CmCustomQueueSweep
- Conceptually, the Sweep Object is a proxy that represents the collection of resources that makes up a sweep. It also provides the interface that allows clients to use and manage them
- Sweep Resources include:
  - A dispatcher
  - A pool of workers
  - A Sweep Action and handler
  - Scheduling information on when it will run
  - Status information on the a sweep while it is in progress
  - The results of the sweep

Resources associated with a Sweep

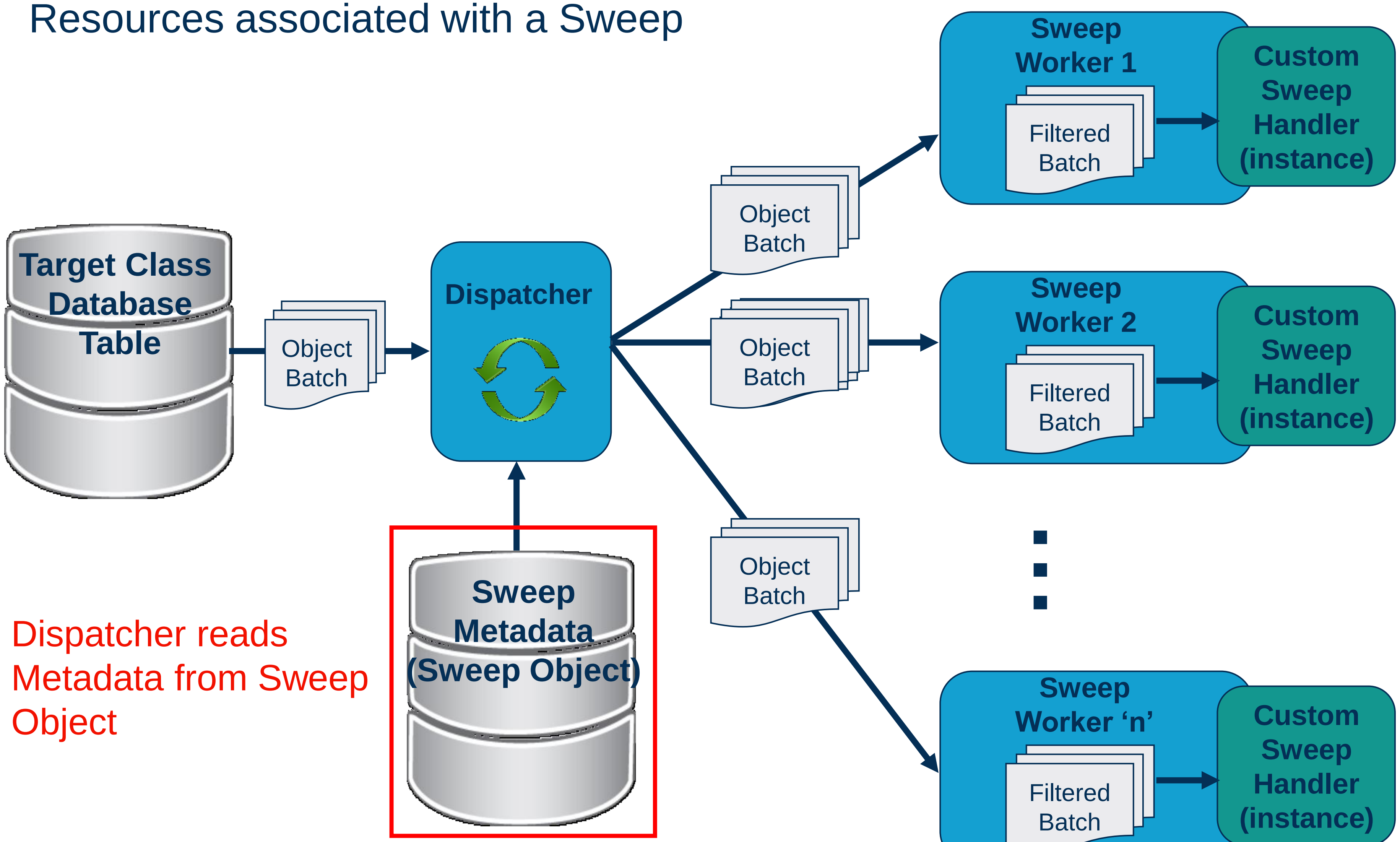

Target Class Database Table
Object Batch
Dispatcher
Sweep Metadata (Sweep Object)
Object Batch
Object Batch
Object Batch
Sweep Worker 1
Filtered Batch
Custom Sweep Handler (instance)
Sweep Worker 2
Filtered Batch
Custom Sweep Handler (instance)
Sweep Worker 'n'
Filtered Batch
Custom Sweep Handler (instance)

Resources associated with a Sweep


Target Class Database Table
Object Batch
Dispatcher
Sweep Metadata (Sweep Object)
Object Batch
Object Batch
Object Batch
Sweep Worker 1
Filtered Batch
Custom Sweep Handler (instance)
Sweep Worker 2
Filtered Batch
Custom Sweep Handler (instance)
Sweep Worker 'n'
Filtered Batch
Custom Sweep Handler (instance)

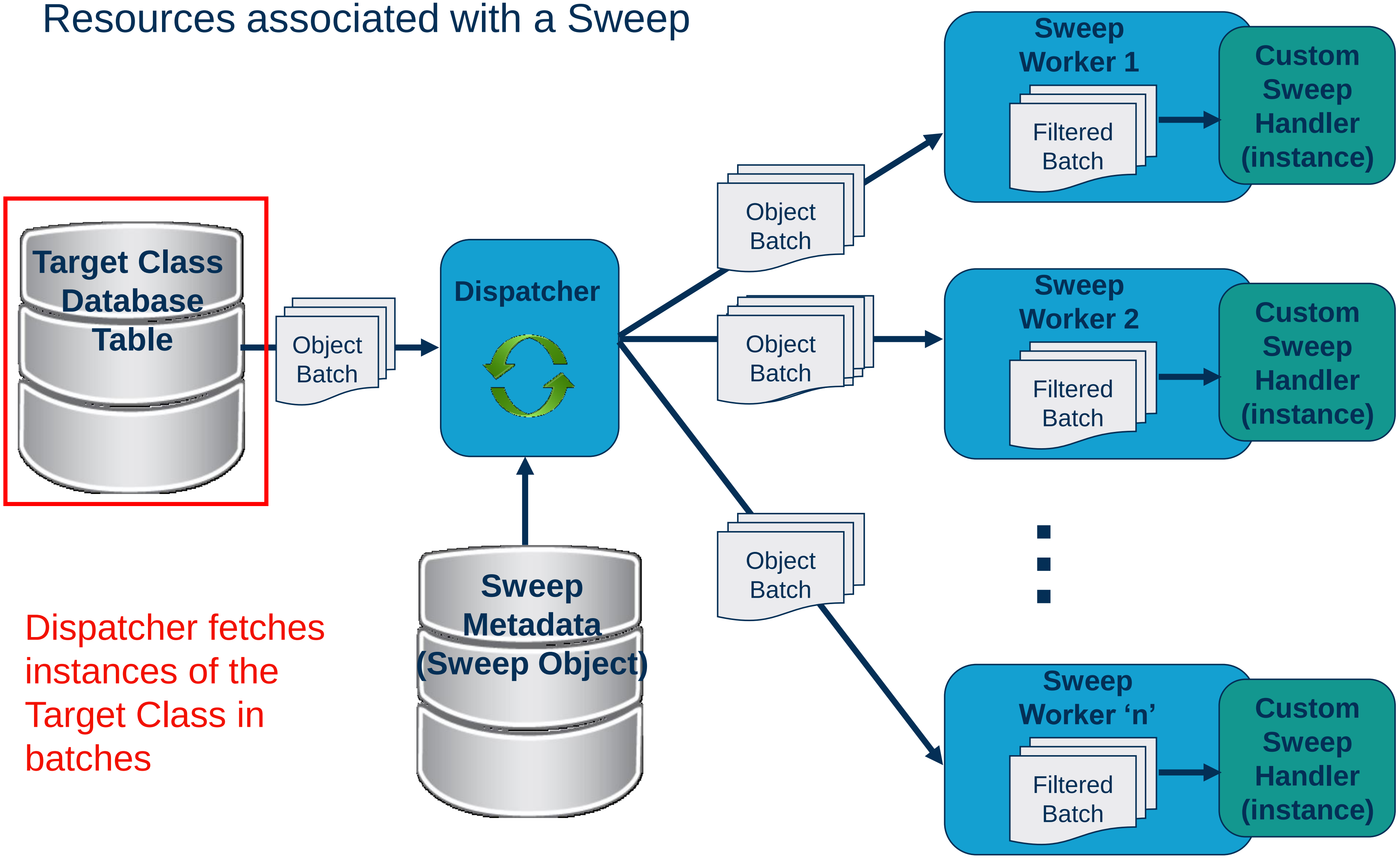
Dispatcher is started by the Master Dispatcher

Resources associated with a Sweep


Target Class Database Table
Object Batch
Dispatcher
Sweep Metadata (Sweep Object)
Object Batch
Object Batch
Object Batch
Sweep Worker 1
Filtered Batch
Custom Sweep Handler (instance)
Sweep Worker 2
Filtered Batch
Custom Sweep Handler (instance)
Sweep Worker ‘n’
Filtered Batch
Custom Sweep Handler (instance)

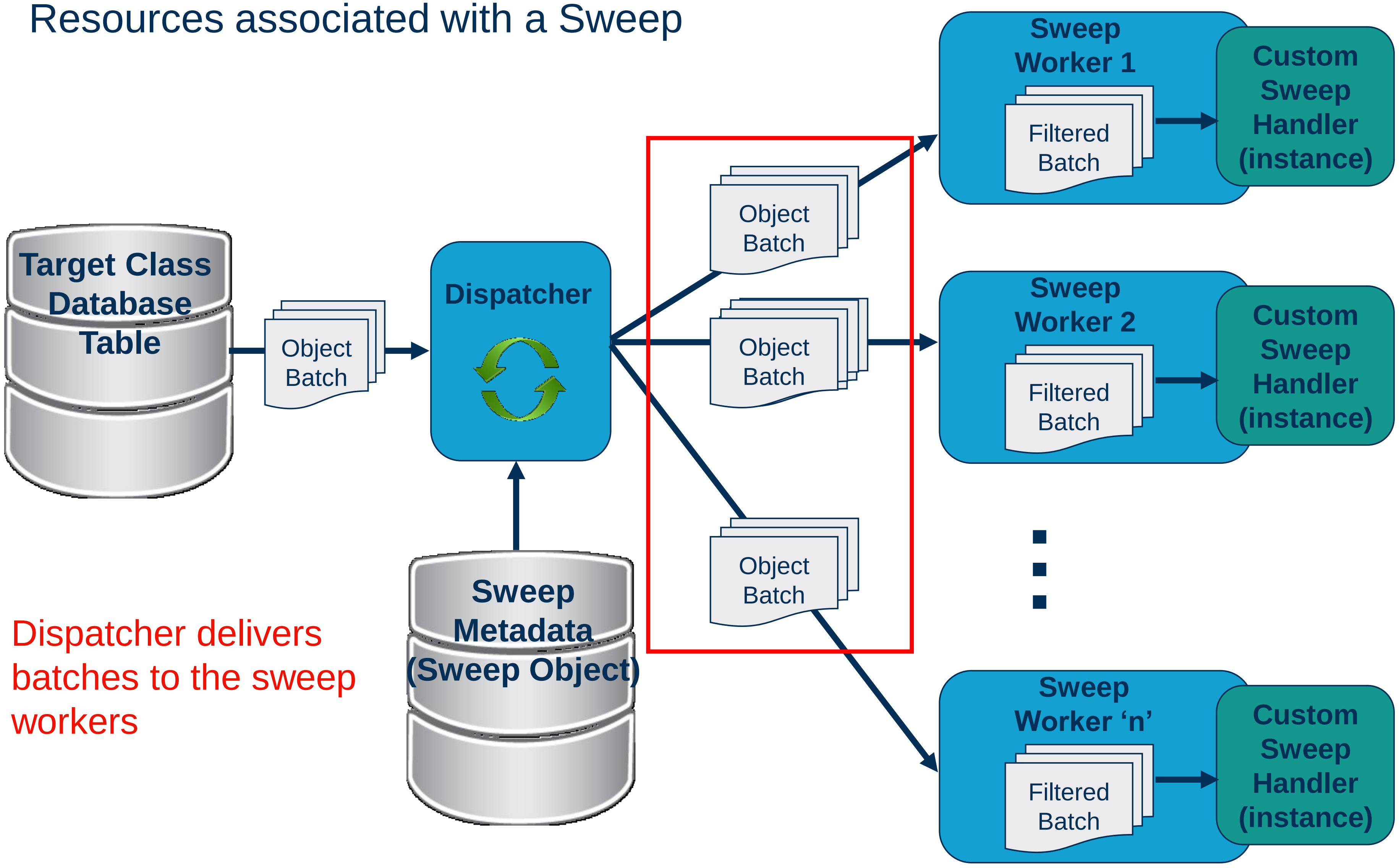
Dispatcher reads Metadata from Sweep Object

Resources associated with a Sweep


Target Class Database Table
Object Batch
Dispatcher
Sweep Metadata (Sweep Object)
Object Batch
Object Batch
Object Batch
Sweep Worker 1
Filtered Batch
Custom Sweep Handler (instance)
Sweep Worker 2
Filtered Batch
Custom Sweep Handler (instance)
Sweep Worker ‘n’
Filtered Batch
Custom Sweep Handler (instance)

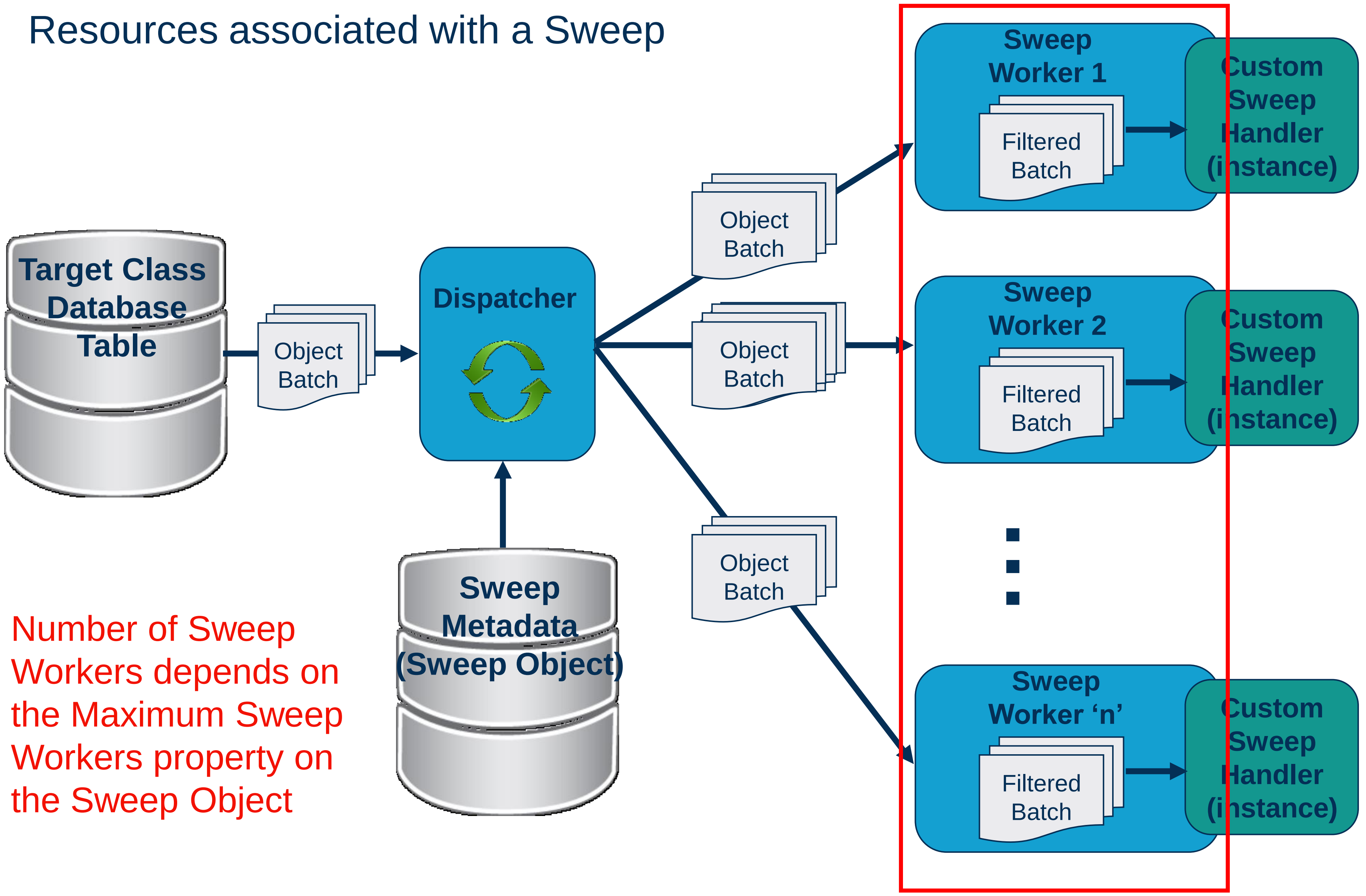
Dispatcher fetches instances of the Target Class in batches

Resources associated with a Sweep


Target Class Database Table
Object Batch
Dispatcher
Sweep Metadata (Sweep Object)
Object Batch
Object Batch
Object Batch
Sweep Worker 1
Filtered Batch
Custom Sweep Handler (instance)
Sweep Worker 2
Filtered Batch
Custom Sweep Handler (instance)
Sweep Worker 'n'
Filtered Batch
Custom Sweep Handler (instance)

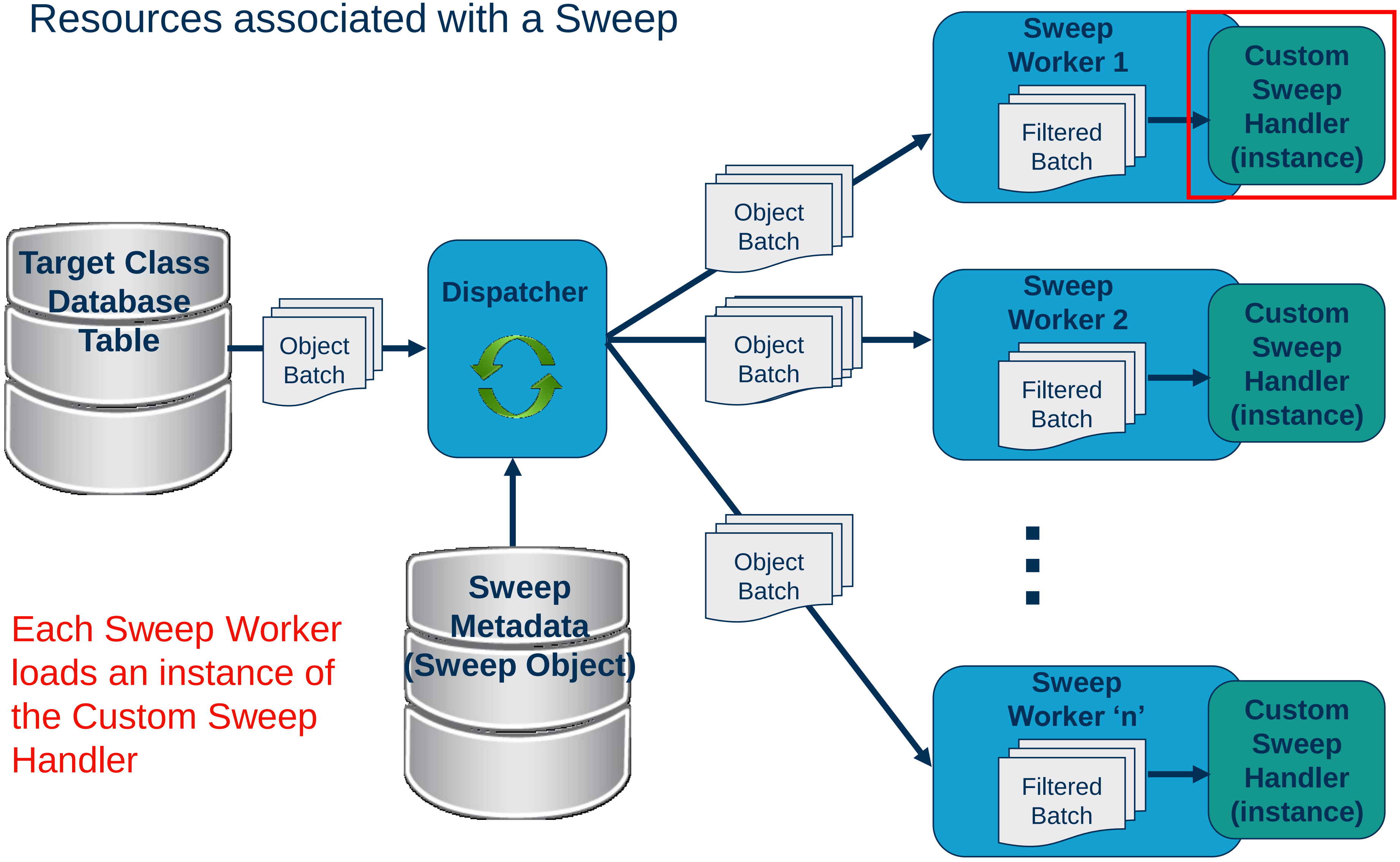
Dispatcher delivers batches to the sweep workers

Resources associated with a Sweep


Target Class Database Table
Object Batch
Dispatcher
Sweep Metadata (Sweep Object)
Object Batch
Object Batch
Object Batch
Sweep Worker 1
Filtered Batch
Custom Sweep Handler (instance)
Sweep Worker 2
Filtered Batch
Custom Sweep Handler (instance)
Sweep Worker 'n'
Filtered Batch
Custom Sweep Handler (instance)

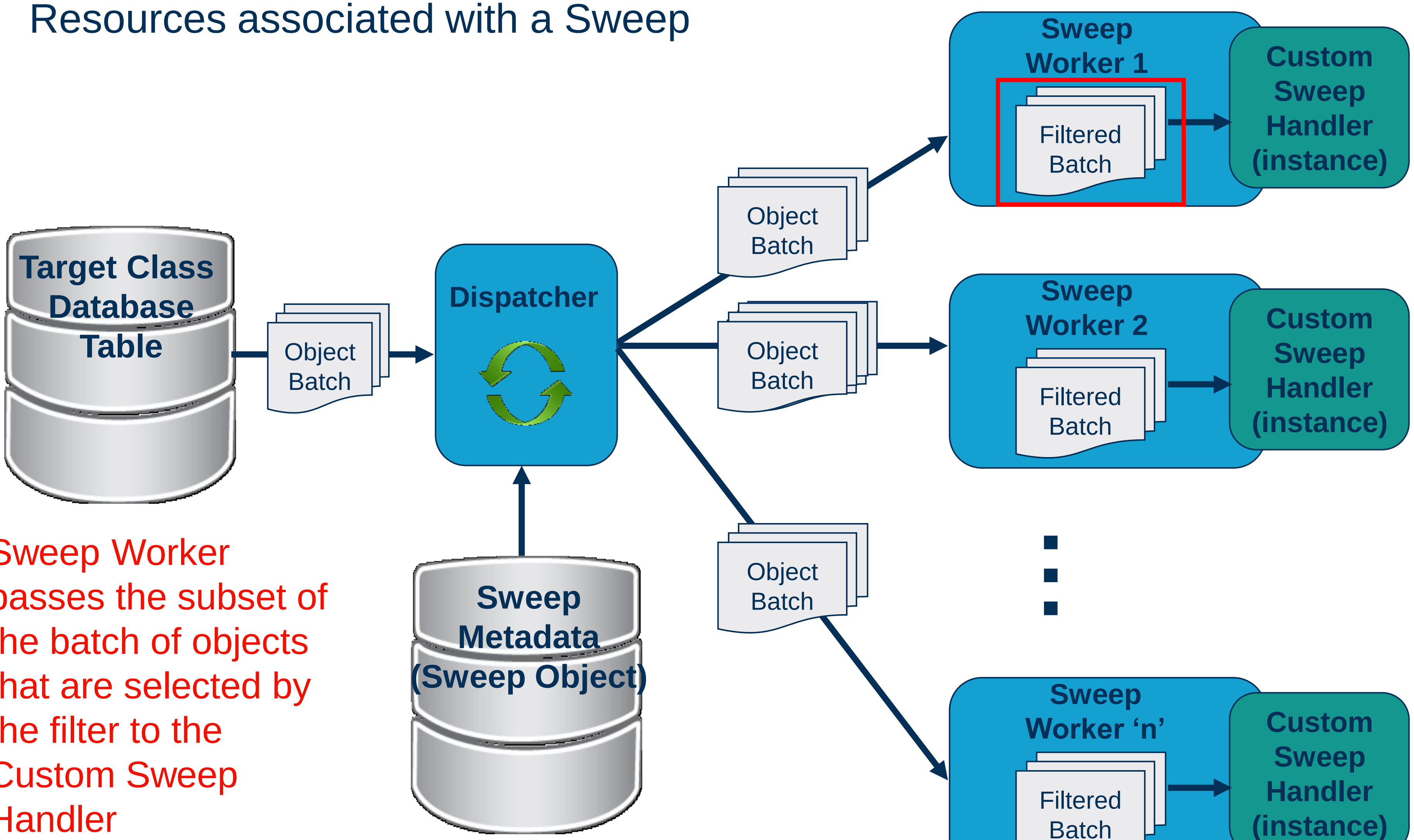
Number of Sweep Workers depends on the Maximum Sweep Workers property on the Sweep Object

Resources associated with a Sweep


Target Class Database Table
Object Batch
Dispatcher
Sweep Metadata (Sweep Object)
Object Batch
Object Batch
Object Batch
Sweep Worker 1
Filtered Batch
Custom Sweep Handler (instance)
Sweep Worker 2
Filtered Batch
Custom Sweep Handler (instance)
Sweep Worker 'n'
Filtered Batch
Custom Sweep Handler (instance)

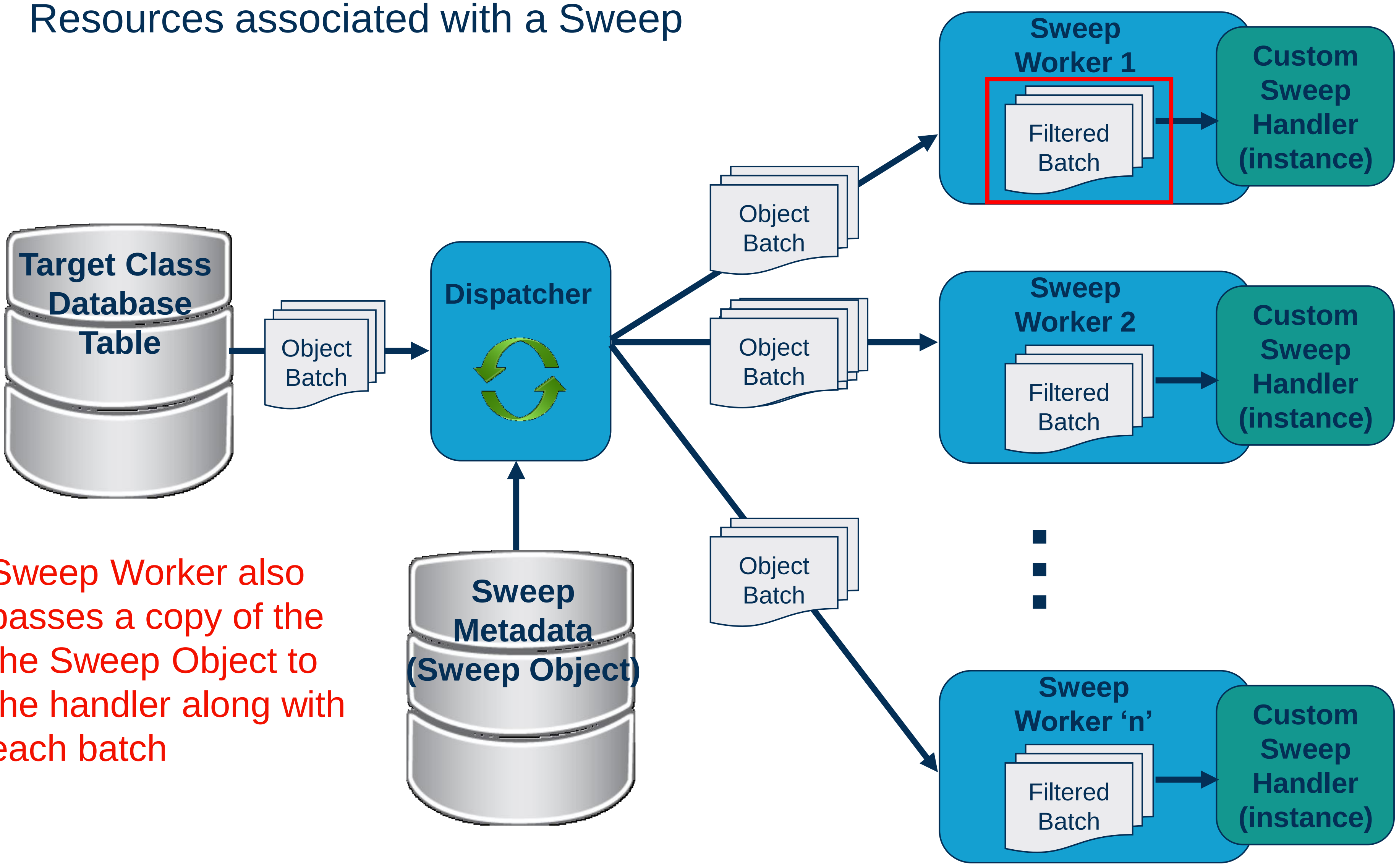
Each Sweep Worker loads an instance of the Custom Sweep Handler

Resources associated with a Sweep


Target Class Database Table
Object Batch
Dispatcher
Sweep Metadata (Sweep Object)
Object Batch
Object Batch
Object Batch
Sweep Worker 1
Filtered Batch
Custom Sweep Handler (instance)
Sweep Worker 2
Filtered Batch
Custom Sweep Handler (instance)
Sweep Worker 'n'
Filtered Batch
Custom Sweep Handler (instance)

Sweep Worker passes the subset of the batch of objects that are selected by the filter to the Custom Sweep Handler

Resources associated with a Sweep


Sweep Worker 1
Filtered Batch
Custom Sweep Handler (instance)
Target Class Database Table
Object Batch
Dispatcher
Object Batch
Sweep Worker 2
Filtered Batch
Custom Sweep Handler (instance)
Object Batch
Object Batch
Sweep Metadata (Sweep Object)
Sweep Worker 'n'
Filtered Batch
Custom Sweep Handler (instance)

Sweep Worker also passes a copy of the the Sweep Object to the handler along with each batch

Course Roadmap

▪ Feature Description
▪ Sweep Framework Overview
▪ Basic Components of a Sweep
➔ Creating a Custom Sweep Handler
▪ Source Code Example
▪ Coding Tips
▪ Demonstration
▪ Security Considerations
▪ Course Summary

Creating a Sweep Handler

- A sweep handler is created by implementing the following interface:
  - com.filenet.api.engine.SweepActionHandler
- Handlers can be implemented in either Java or Java Script

Creating a Sweep Handler

- SweepActionHandler interface
  - The SweepAction Handler interface consists of the following methods:

void o**nSweep**(CmSweep sweepObject, SweepActionHandler.SweepItem [] sweepItems);

void **onPolicySweep**(CmSweep sweepObject, CmSweepPolicy sweepPolicy,

SweepActionHandler.SweepItem [] sweepItems);

String [] **getRequiredProperties**();

Creating a Sweep Handler

- onSweep() method
  - Signature:
    - void onSweep(CmSweep sweepObject, SweepActionHandler.SweepItem [] sweepItems);
  - OnSweep method is where you implement the behavior that you want your Sweep Action to perform. It is where you will spend the bulk of your coding effort when creating a custom sweep handler
  - The sweepObject parameter is a reference to the Custom Sweep API object with which the handler is associated. The type will be either: Custom Sweep Job, Custom Queue Sweep or Policy Controlled Sweep.
  - The sweepItems parameter is array of items that are instances of the Sweep Target class specified on the Sweep Object that satisfy the filter condition
  - The number of items in the array will always be some value that is less than or equal to the batch size specified by the Sweep Object
  - The onSweep() method will typically be called many times, once for each batch of items that the dispatcher fetches from the table associated with the Target Class and that also satisfy the filter condition
  - The onSweep() method must be thread safe

Creating a Sweep Handler

- Basic pattern typical of onSweep() implementations

```
FOR EACH SweepItem in the SweepItem Array DO:
    IF server is shutting down THEN
        THROW E_BACKGROUND_TASK_TERMINATED
    END IF
    CALL Action(SweepItem)
    IF Action is successful THEN
        CALL SweepItem.setOutcome(PROCCESSED)
    ELSE
        CALL SweepItem.setOutcome(FAILED)
    END IF
END FOR EACH
```

Creating a Sweep Handler

- OnPolicySweep() method
  - The onPolicySweep() method is just like the onSweep() method except it supports an additional parameter that is used to pass in the Custom Sweep Policy object to your handler
  - The Sweep Object in this case is a Policy Controlled Sweep, which normally will not hold much interest for custom sweep implementers

Creating a Sweep Handler

- getRequiredProperties() method
  - Returns an array of strings consisting of names of properties that are defined for the Sweep Target class
  - Used to provide a hint to the Sweep Dispatcher for how to construct the query more efficiently when obtaining batches of candidate objects to pass to the handler
  - Declares to the sweep framework what properties the handler is interested in seeing
  - Returning an empty array causes the framework to include all properties associated with each object instance
  - Returning a non-empty array that fails to include one of the properties the handler needs could result in raising a PROPERY NOT IN CACHE exception if the handler attempts to retrieve its value

Creating a Sweep Handler

- SweepItem interface
  - The SweepItem interface is how the Sweep Handler interacts with the items that are delivered to it by the Sweep Framework for the handler to process
  - An array of SweepItems are passed to the onSweep() method with each invocation
  - The array contains instances of the Target Class specified on the Sweep Object that also satisfy the filter condition
  - The SweepItem interface consists of the following methods:

    IndependentlyPersistableObject **getTarget**();

    void **setOutcome**(SweepItemOutcome outcome, String description);

    void **setDeferral** (int deferralSeconds, byte[] deferralData);

Creating a Sweep Handler

- getTarget() method
  - The getTarget() method returns an instance of IndependentlyPersistableObject that is the object upon which you will perform the operations associated with your custom handler
  - IndependentlyPersistableObject is the abstract base class for all Content Engine objects that can be created and saved and are independently addressable
  - The object returned by getTarget() can be cast to the class specified by the SweepTarget property on the Sweep object

Creating a Sweep Handler

- setOutcome() method
  - Call this method once for each Sweep Item in the batch to notify the framework of the outcome of the processing
  - Possible outcomes:
    - SweepItemOutcome.PROCESSED
      - Indicates that this sweep item was processed successfully and will cause the ProcessedObjectCount property on the Sweep object to be incremented.
    - SweepItemOutcome.FAILED
      - Indicates that processing failed for this sweep item and will cause the FailedObjectCount property on the Sweep object to be incremented
      - The description argument can also be used to provide a string that indicates the reason for the failure. This value will show up in the Sweep Result if the sweep was configured to record failures
    - SweepItemOutcome.IGNORED
      - Indicates no action was taken
    - SweepItemOutcome.FAILED_NO_RETRY
      - Only used by for Queue Sweeps
      - Failed items are automatically retried unless this value is used

Creating a Sweep Handler

- setDeferral() method
  - Only used in CustomQueueSweeps, otherwise it is ignored
  - Purpose is to notify the sweep framework that it should defer processing for this Sweep Item
  - By ‘defer processing’ we mean that the Sweep Item will not be passed back to the handler until a specified number of seconds have expired off the clock
  - deferralSeconds parameter tells the sweep framework how long to defer processing
  - deferralData is an arbitrary data blob that a custom handler can use to tag a deferred Sweep Item with context information so that it when it appears again, the handler will know its been seen before and in what context
  - Example:

    the deferral data could be set the job number for a print job

Course Roadmap

- Feature Description
- Sweep Framework Overview
- Basic Components of a Sweep
- Creating a Custom Sweep Handler

➔ Source Code Example

- Coding Tips
- Demonstration
- Security Considerations
- Course Summary

Course Roadmap

- Feature Description
- Sweep Framework Overview
- Basic Components of a Sweep
- Creating a Custom Sweep Handler
- Source Code Example
- ➔ Coding Tips
- Demonstration
- Security Considerations
- Course Summary

Coding Tips

- SweepHandler.OnSweep() is not started within the context of an enclosing J2EE transaction
- Use the handler call context for trace logging
- Make sure server is not shutting down at that top of your Sweep Item processing loop
- Make sure the outcome is set for each item in the batch – use try catch blocks
- In general, you should handle all exceptions and not allow them to propagate to the caller
- Never update the sweep item in a custom queue sweep handler
- Use getRequiredProperties()
- Make sure a property is in the cache before referring to it

Course Roadmap

- Feature Description
- Sweep Framework Overview
- Basic Components of a Sweep
- Creating a Custom Sweep Handler
- Source Code Example
- Coding Tips
- ➔ Demonstration
- Security Considerations
- Course Summary

Course Roadmap

- Feature Description
- Sweep Framework Overview
- Basic Components of a Sweep
- Creating a Custom Sweep Handler
- Source Code Example
- Coding Tips
- Demonstration

→ Security Considerations

- Course Summary

Security Considerations

- Custom sweep handler execute with access checking disabled
- This means:
  - You do not need to worry about your handler being unable to perform an operation due to having insufficient access rights
  - The handler executes as privileged code inside the server and so only trusted users should be allowed to create or modify them)

Course Roadmap

- Feature Description
- Sweep Framework Overview
- Basic Components of a Sweep
- Creating a Custom Sweep Handler
- Source Code Example
- Coding Tips
- Demonstration
- Security Considerations

➔ Course Summary

Course Summary

In this course we covered the following:

- We described a custom sweep as a user initiated background task that has custom behavior intended to process large numbers of objects for some specified class
- We identified 3 basic types of Custom Sweeps and the types of use cases they support:
  - CustomSweepJobs – Single iteration sweeps for one-time-only updates
  - CustomSweepPolicies – Open ended sweeps for processing objects as they age
  - CustomQueueSweeps – Generalized queuing service for slow, resource intensive processes
- We identified the components of a Sweep:
  - Handler, Code Module, Sweep Action, Sweep object
- We described the SweepActionHandler interface, which is the interface that must be implemented by the handler to create a custom sweep
- We described the general pattern for implementing the onSweep() method
- We described how to use the SweepItem interface in a custom handler to retrieve the target objects and to return outcomes

Contacts

- Product Marketing Manager:
  - Robert Finn
- Product Manager:
  - Stephen Hussey
- Subject Matter Experts (SME)/Area of Expertise:
  - Grace Smith (Development Manager)
  - Eric Edeen (Software Developer)
  - Bob Kreuch (Software Developer)
- Support:
  - Erik Fonkalsrud (L3 Manager)

Product Help / Documentation / Resources

- P8 5.2.1 Information Center (available October 31st)
  http://www.ibm.com/support/knowledgecenter/SSNW2F_5.2.1/
- Custom Sweep documentation in TOC:
  - ➔ Developing FileNet P8 applications
    - ➔ Content Engine Development
      - ➔ Content Engine Java and .NET Developer’s Guide
        - ➔ Server Extensions
          - ➔ Custom Sweeps
        - ➔ Reference
          - ➔ SQL Syntax Reference
            - ➔ Relational Queries