

# Workflow Services



**DEVELOPMENTOR**

DEVELOPING PEOPLE WHO DEVELOP SOFTWARE

# Objectives

- **Why workflow services**
- **Declarative services using XAML**
- **Service activities**
- **Correlation**



# Workflow and Services

- **Workflow powerful tool for building services**
  - declarative model provides clarity in processing
  - auto persistence infrastructure for long running stateful processes
- **Service composition simplified**
  - composing services can make it hard to understand the data flow between the different services
  - workflow's data flow and graphical designer make it simpler to understand how services are composed



# Creating Workflow Based Services

- **WF 4.0 now supports a fully declarative mode for authoring services**
  - services and endpoints can be configured in **XAML**
  - services defined in .xamlx file
  - contracts and endpoints still be declared in config



# Service Activities

- **New activities for modelling service interaction**
- **Four new activities**
  - **Receive** – inbound one-way message
  - **ReceiveAndSendReply** – inbound request/response processing
  - **SendAndReceiveReply** – request/response outbound
  - **Send** – one-way outbound



# Receive

- **Simple inbound request**
  - captures inbound data

**Receive**

OperationName

Content *View parameters...*

Content Definition

☐ Message

☒ Parameters

Name	Type	Assign To
order	Order	<i>Enter a VB expression</i>

*Add new parameter*

OK Cancel

**Properties**

System.ServiceModel.Activities.Receive

Search:  Clear

**Correlations**

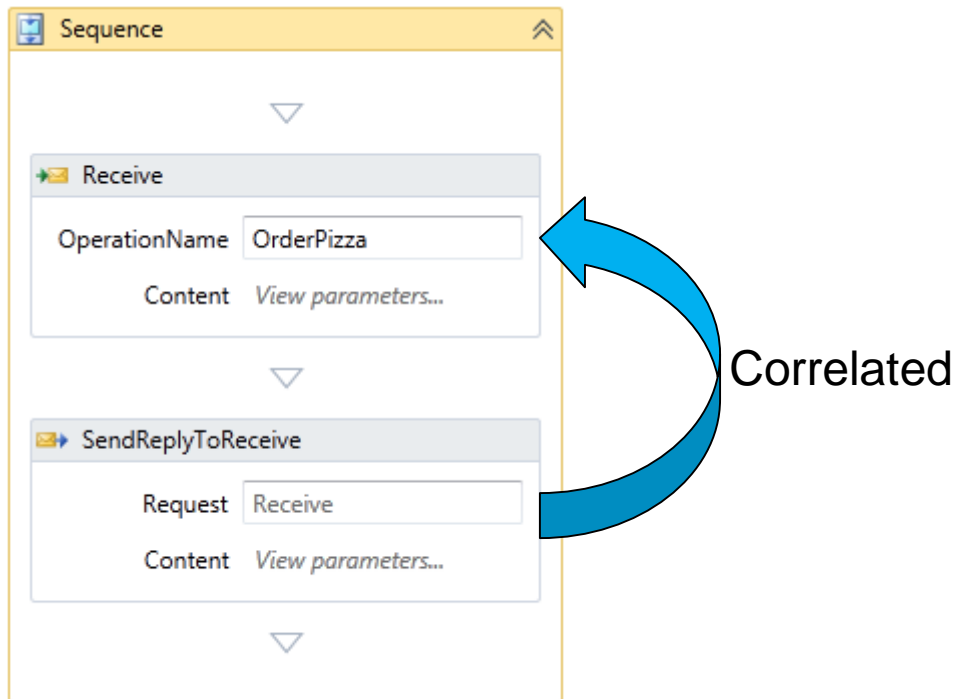
CorrelatesOn	(Collection)	...
CorrelatesWith	<i>Correlation handle</i>	...
CorrelationInitializers	(Collection)	...

**Misc**

Content	(Content)	...
DisplayName	Receive	
OperationName	ReceiveOrder	
ServiceContractName	{http://develop.com/services}	
More Properties		
Action		
CanCreateInstance	<input type="checkbox"/>	
KnownTypes	(Collection)	...
ProtectionLevel	(null)	
SerializerOption	DataContractSerializer	

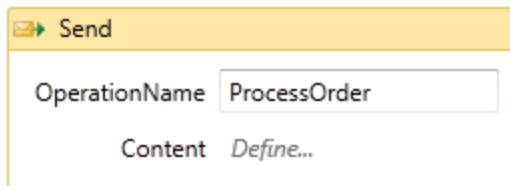
# ReceiveAndSendReply

- **Inbound request / response messaging**
  - Uses two correlated activities

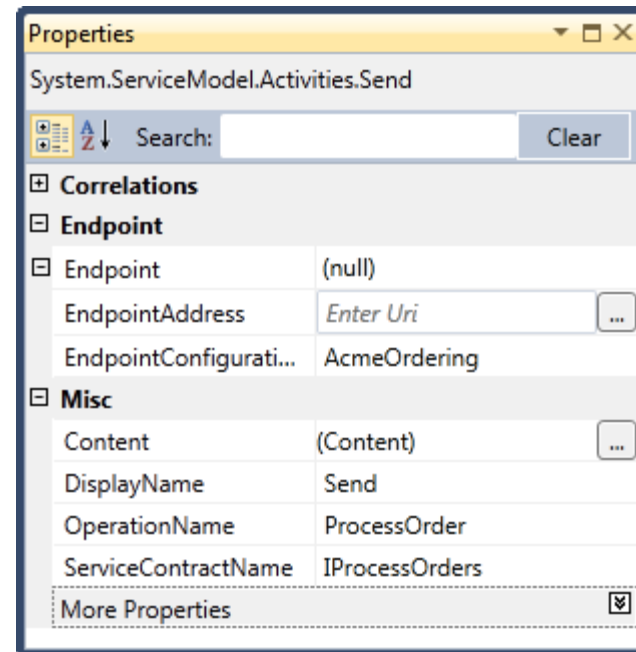


# Send

- **One-way outbound message processing**
  - Specify endpoint details explicitly or reference named endpoint in config
  - Specify operation name to be invoked



The 'Send' activity configuration window shows the 'OperationName' set to 'ProcessOrder' and the 'Content' set to 'Define...'.



The 'Properties' window for the 'Send' activity (System.ServiceModel.Activities.Send) shows the following configuration:

Properties	
System.ServiceModel.Activities.Send	
Search: <input type="text"/> Clear	
Correlations	
Endpoint	
Endpoint	(null)
EndpointAddress	Enter Uri
EndpointConfigurati...	AcmeOrdering
Misc	
Content	(Content)
DisplayName	Send
OperationName	ProcessOrder
ServiceContractName	IProcessOrders
More Properties	





# SendAndReceiveReply

- **Request / Response outbound message processing**
  - Correlated pair of activities
  - Specify endpoint details explicitly or reference named endpoint in config
  - Specify operation name to be invoked

The diagram illustrates the configuration of a **Send** activity and its corresponding **ReceiveReplyForSend** activity within a **Sequence** container. A blue curved arrow labeled **Correlated** points from the **Send** activity to the **ReceiveReplyForSend** activity, indicating their relationship.

**Sequence Window:**

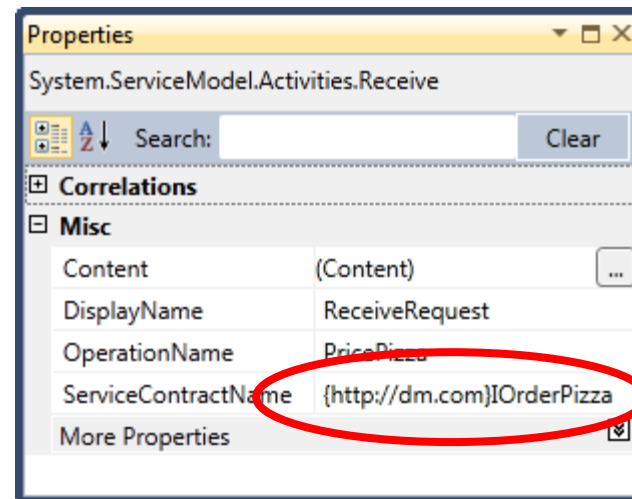
- Send Activity:**
  - OperationName:
  - Content:
- ReceiveReplyForSend Activity:**
  - Request:
  - Content:

**Properties Window (System.ServiceModel.Activities.Send):**

- Correlations:**
- Endpoint:**
  - Endpoint: (null)
  - EndpointAddress:
  - EndpointConfigurati...:
- Misc:**
  - Content: (Content)
  - DisplayName:
  - OperationName:
  - ServiceContractName:
  - More Properties: ☒

# Contract Inference

- **Contract is inferred from receives**
  - Service contract is string entered in property window
  - Operations inferred from receive operation names and parameters
  - Care must be taken to ensure service contract name exactly matches for all operations intended to be on same contract



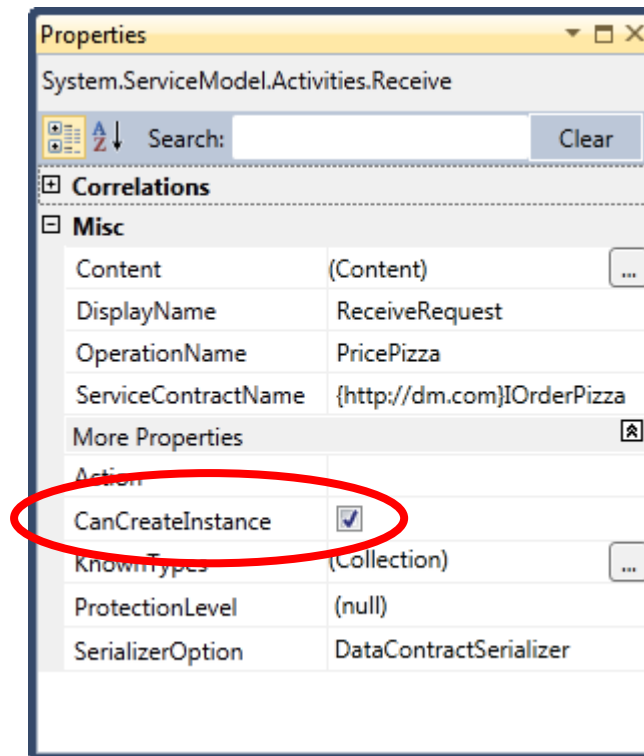
# Importing Service Contracts

- **In 4.5 can import Service Contract**
  - Generates activities
  - Right click on project and import
- **Can Update Service Contract**
  - Right click on contract in Service Contracts folder in project



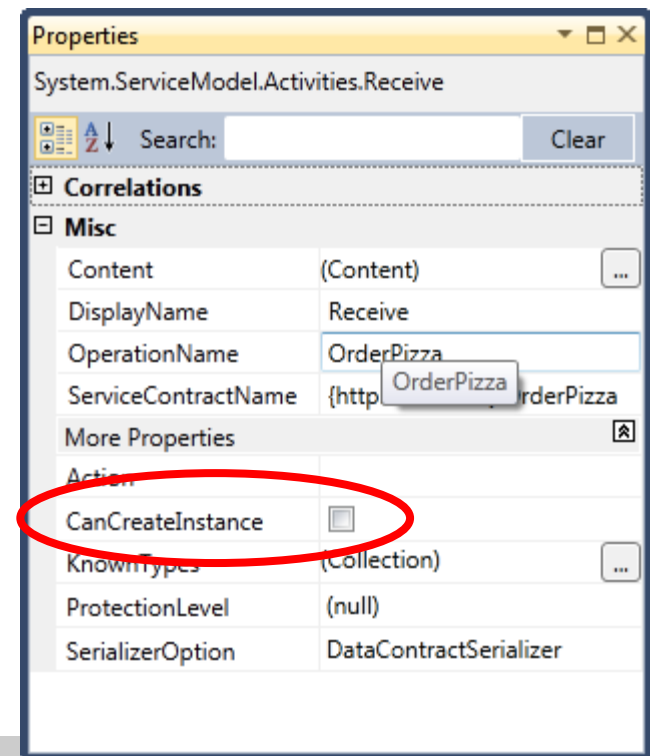
# CanCreateInstance

- A receive often states that a message received creates a new instance of the workflow



# Long Running Execution

- **Services that run for extended periods have particular requirements**
  - state must be maintained: persistence
  - further requests must be directed to already running instance of workflow: correlation
- **Need non-creating receives**



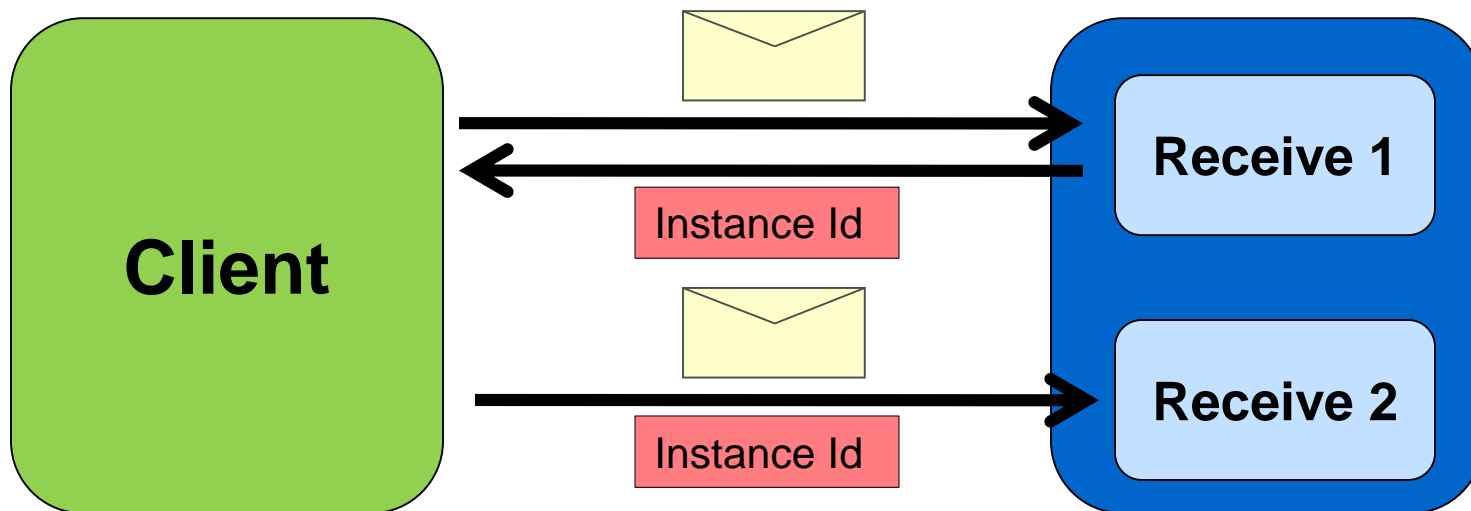
# Correlation

- **3.5 supported “context correlation”**
  - workflow instance id passed back to client
  - client sends workflow instance id in each request
  - handled automatically by ContextChannel
  - client needed to store and restore context between client restarts
- **WCF 4.0 supports context correlation and data correlation**
  - data correlation uses data in the message to map to the appropriate workflow instance –e.g. Invoice Number
  - same model as BizTalk



# Context Correlation

- **ContextChannel** manages the workflow instance id
  - stored inside a message header or cookie
- **Must use special bindings containing the context channel**
  - WSHttpContextBinding
  - NetTcpContextBinding
  - BasicHttpContextBinding



# Managing the client context

- Long running services require restartable clients
- Client can obtain and save context from channel property

```
IChannel chan = (IChannel)client.InnerChannel;  
IContextManager ctxMgr = chan.GetProperty<IContextManager>();  
IDictionary<string, string> ctx = ctxMgr.GetContext();  
WriteContextToStore(ctx);
```

- Client can read and restore context using channel property

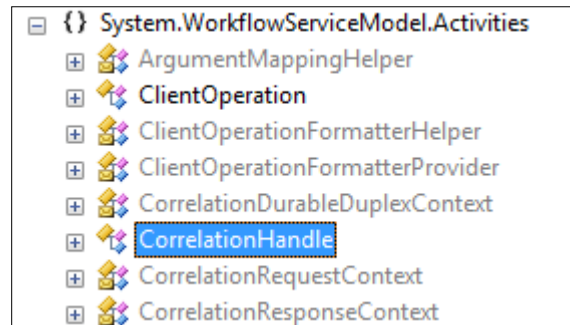
```
IChannel chan = (IChannel)client.InnerChannel;  
IContextManager ctxMgr = chan.GetProperty<IContextManager>();  
IDictionary<string, string> ctx = GetContextFromStore();  
ctxMgr.SetContext(ctx);
```





# Data Correlation

- **Need a way to specify what data in the message indicates the destination workflow instance**
  - specified data must be unique for the lifetime of the workflow instance, e.g. Invoice Number
- **CorrelationHandle**
  - used to identify data to be used for a correlation
  - single workflow instance may have a number of different correlations



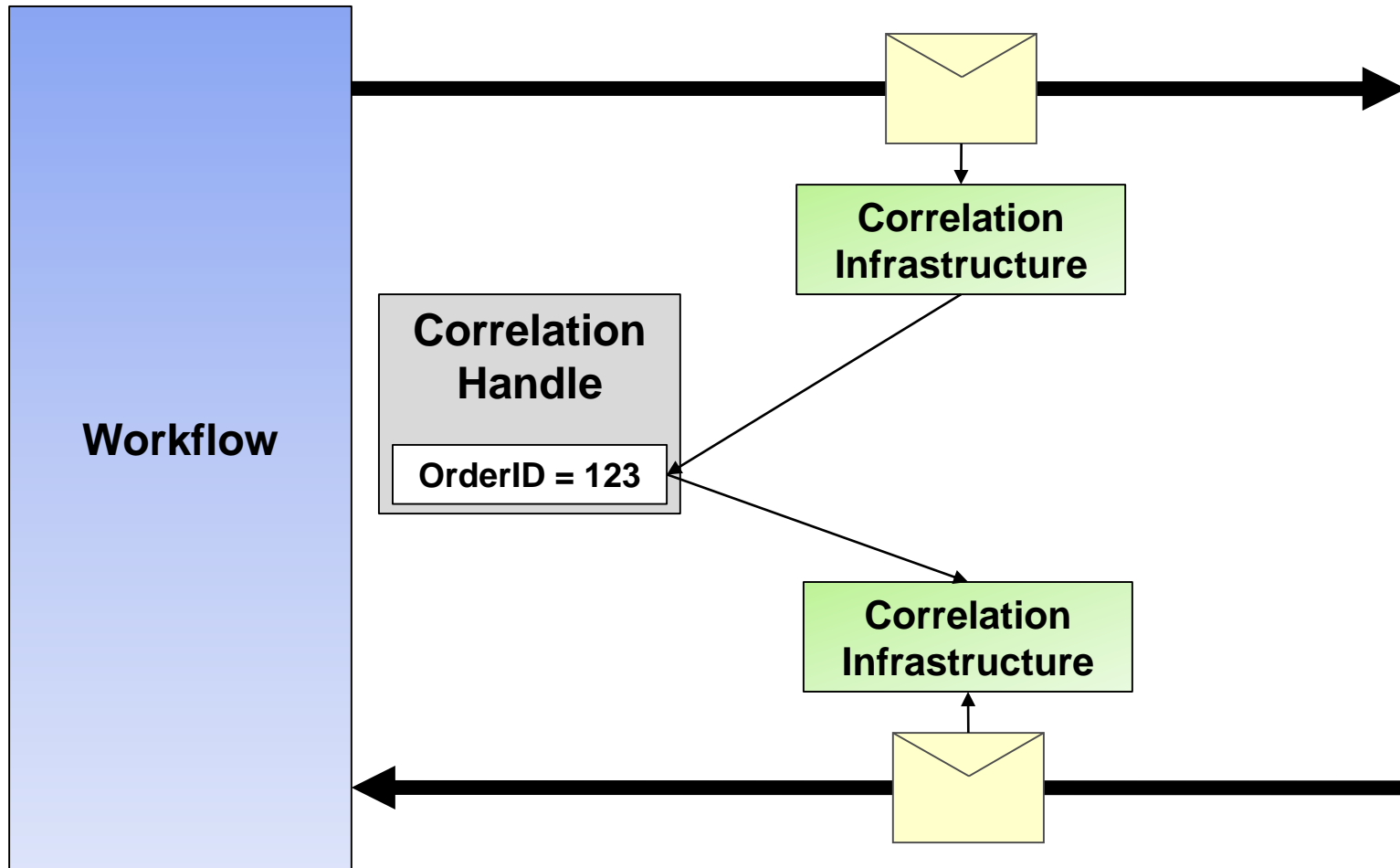
# Setting Up Data Correlation

1. Create variable of type **CorrelationHandle**
2. Initialize correlation via **CorrelationInitializers** in receive or send
  - specify XPath statements to the correlation data in the request or response message
3. Specify same **CorrelationHandle** in one or more receive activities with **CorrelatesWith**
  - specify XPath statements to the correlation data in this message using **CorrelatesOn**

Correlations		
CorrelatesOn	(Collection)	...
CorrelatesWith	pizzaCorrelation	...
CorrelationInitializ...	(Collection)	...



# Data Correlation Example



# Summary

- **Services can be built from workflows**
- **Services can now be built fully declaratively**
- **Services now support two models for correlation**
  - context
  - data orientated

