Microsoft

Microsoft Dynamics® AX 2012

Customer payment journal service in Microsoft Dynamics AX 2012 R2 (Public Sector only)

This white paper provides an overview of operations exposed in the Customer payment journal service in Microsoft Dynamics AX 2012 R2. It also outlines some common development scenarios and provides step-by-step instructions to integrate with the customer payment journal web service.

http://microsoft.com/dynamics/ax

Date: May 2013

Send suggestions and comments about this document to adocs@microsoft.com. Please include the title with your feedback.


Microsoft Dynamics

Table of Contents

Introduction

In the **Accounts receivable** module of Microsoft Dynamics AX 2012 R2, payments can be received from customers either electronically or through an external cashiering system. The Customer payment journal service (LedgerCustPaymJournalService) provides operations to accept external payments and create the required journals and journal lines in Microsoft Dynamics AX 2012. The payments are then matched against open transactions and settled automatically based on a system-defined settlement priority. The Customer journal payment service can be used in conjunction with Customer transaction service to find a customer and the customer's open transactions.

Purpose

This document discusses the Customer payment journal service along with its operations and considerations for its implementation. The topics that are covered include:

- Prerequisites for integration with the Customer payment journal service
- Service overview
- Basic service integration scenarios
- Settlement priorities

Terminology

Term	Definition
AIF	Application Integration Framework (AIF) provides an extensible framework that supports multiple asynchronous transports, as well as synchronous transport using Web services, to reliably exchange documents in XML format with trading partners or other systems. [MSDN]

Audience

This document is intended for developers who are implementing a solution that integrates with the Customer payment journal service. General readers can get an overview of the service from the first few sections.

Assumptions

This document assumes that the developer has an understanding of web services in general and web services in Microsoft Dynamics AX in particular. The following table provides links to information about services and the AIF Document Services in Microsoft Dynamics AX 2012.

Document	Description
Services [White paper]	Provides an overview of different types of services available in Microsoft Dynamics AX 2012 and guidance to developers and architects in deciding when to use a specific service type
AIF Document Services [MSDN]	Provides an overview of Application Integration Framework and how to create a document-based web service.

The document also assumes that the *Contoso* dataset is imported into Microsoft Dynamics AX 2012. The example uses the CEU company.

Prerequisites

To benefit from this white paper, you should have experience in the following areas:

- Writing and debugging code in .NET Framework languages such as C# and VB.NET. (C# is used for the examples in this white paper.)
- Familiarity with Microsoft Visual Studio®
- Working experience with X++ and the MorphX development environment

You must have Microsoft Visual Studio 2010 (or later) installed to develop or debug the C# consumer application described in this white paper.

Customer payment journal service

The Customer payment journal service is a document-based service that is used to create and manage customer payment journals. The service is located under AOT\Services\LedgerCustPaymJournalService.

Operations

The Customer payment journal service supports the following operations:

- create
- find
- findKeys
- getChangedKeys
- getKeys
- read
- settlePayment
- update

Enhancements for Microsoft Dynamics AX 2012 R2

The settlePayment" operation was added to the Customer payment journal service. This operation allows the user to mark open customer transactions for settlement according to system-defined settlement priorities.

Settlement priority is an Accounts receivable feature that allows system administrators to define settlement priorities based on certain attributes such as transaction type, due date, and transaction amount. For information about configuring settlement priorities in the Accounts receivable module, see Appendix A: Configure the settlement priority later in this paper.

Setting up the Customer payment journal service

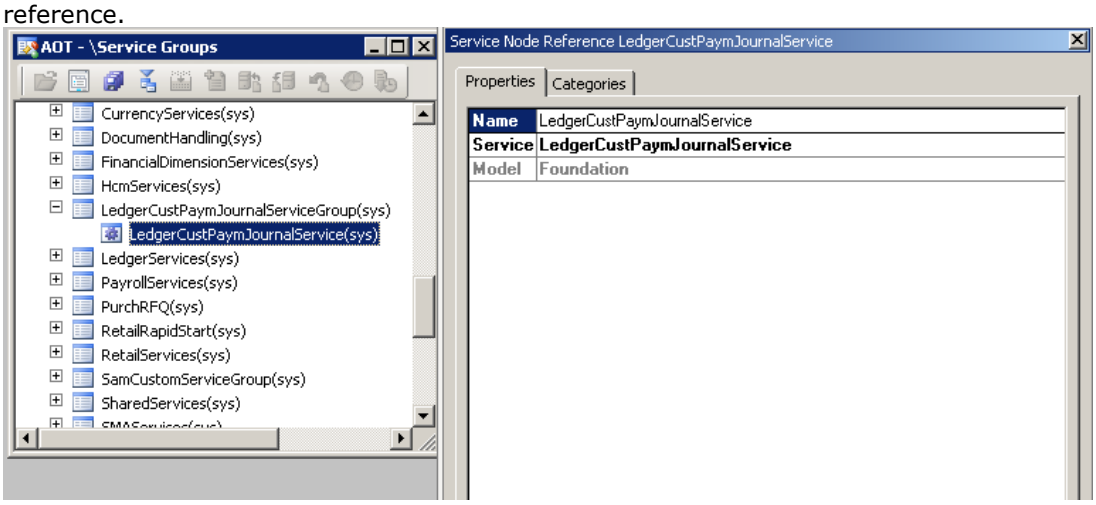
This section describes how to set up the services.

Deploying the service

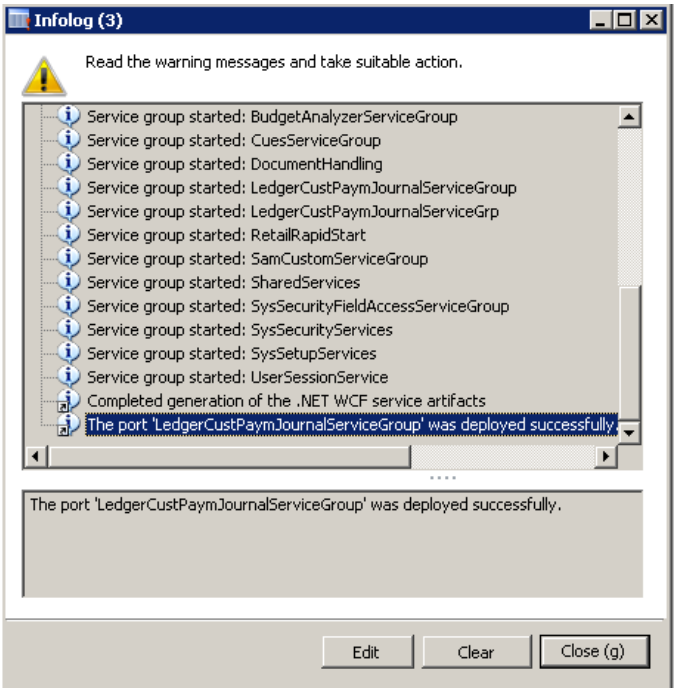
Follow these steps to deploy and configure the service on a basic integration port. For more information, see Using Basic Integration Ports.

1. Open Microsoft Dynamics AX 2012 and switch to the development workspace.
2. Right click **AOT** > **Service Groups,** and then select **New Service Group.**
3. Name the group ***LedgerCustPaymJournalServiceGroup.***
4. Right click the newly-created group, and then select **New Service Node Reference.**
5. Rename the service node reference to ***LedgerCustPaymJournalService.***

6. Right-click the newly-created service node reference, and then select **Properties**.
7. Select **LedgerCustPaymJournalService** under the service property of the service node reference.

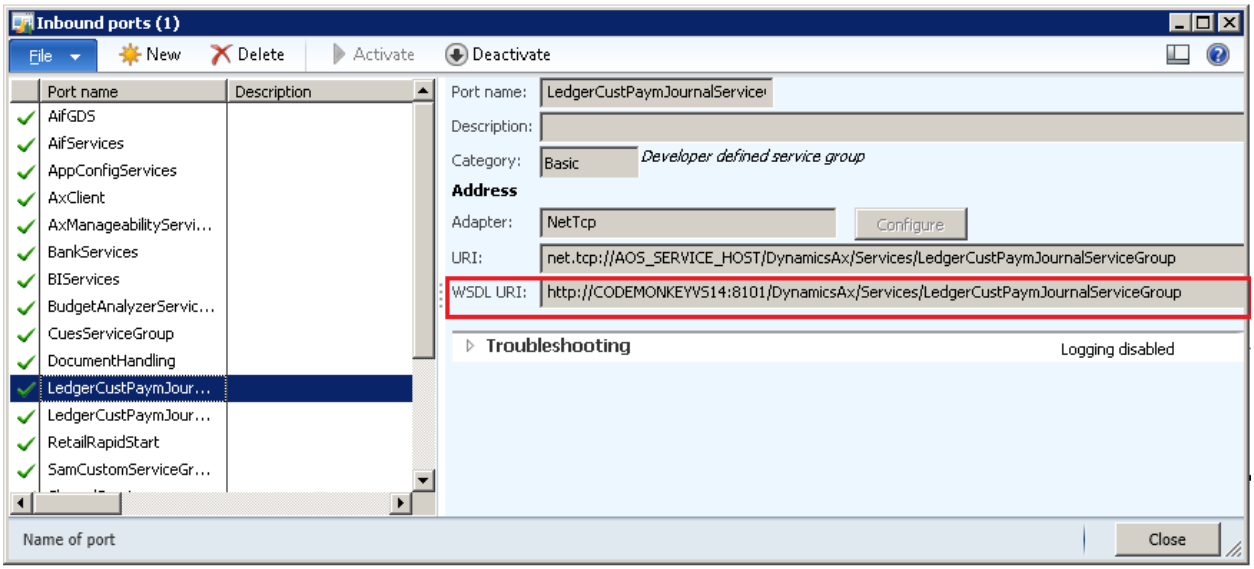


8. Right-click **LedgerCustPaymJournalServiceGroup**, and then click **Deploy Service Group**.
9. Make sure the service is deployed successfully.



10. Go to **System Administration** (module) > **Setup** > **Services and Application Integration Framework** > **Inbound ports**.
11. Select the **LedgerCustPaymentJournalServiceGroup** and make sure it's activated. If it is not activated, click the **Activate** button.

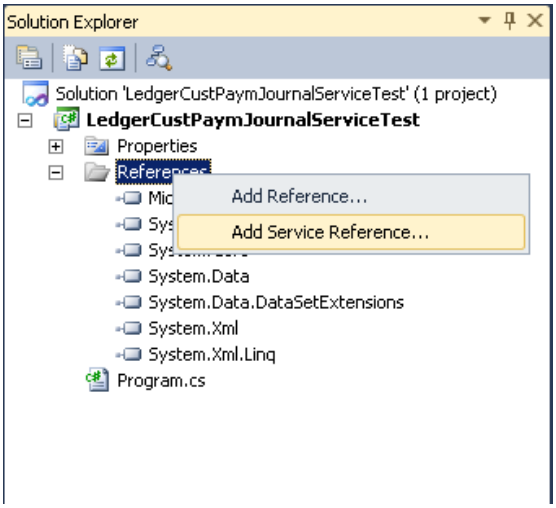
12. Accept the default **WSDL URI** as shown in the following illustration.



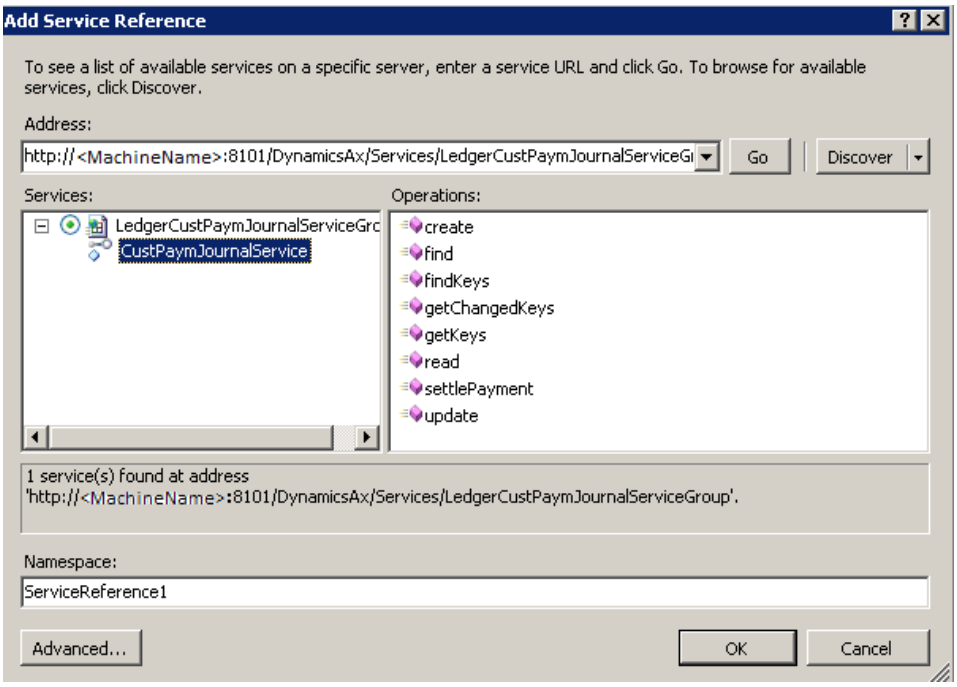
Creating a .NET project to consume the service

Follow these steps to create a project with the **LedgerCustPaymJournalServiceTest** web service reference:

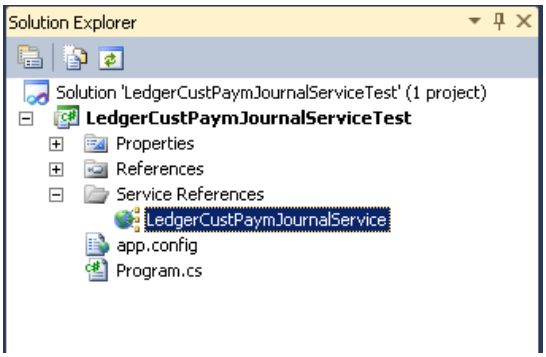
1. Open Microsoft Visual Studio 2010.
2. Click **File** > **New** > **Project.**
3. Select **Console Application** under **Visual C#** > **Windows** templates.
4. Name the Project **LedgerCustPaymJournalServiceTest**, and then click **OK**.
5. In Solution Explorer, right-click the **References** node, and then select **Add Service Reference**.



6. In the **Add Service Reference** dialog box, paste the WSDL URI (from step 12 of the previous procedure, *Deploy the service*) into the **Address** bar, and then click **Go**. You should be able to see the service and its operations as shown in the following illustration.



7. Enter a meaningful name for the namespace, such as **LedgerCustPaymJournalService**, and then click **OK.**



The service reference has successfully been added to the console test application.

Using the Customer journal payment service

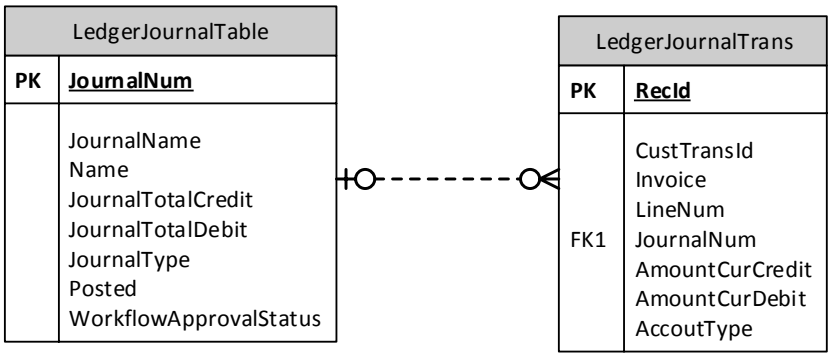
This section outlines how to use the service.

Skeleton of the C# class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

// import the service reference as a namespace
using LedgerCustPaymJournalServiceTest.LedgerCustPaymJournalService;

namespace LedgerCustPaymJournalServiceTest
{
    class Program
    {
        static void Main(string[] args)
        {

        }
    }
}
```

Creating a payment journal and line

The following data model represents a payment journal and line in the Microsoft Dynamics AX database.



The following sample C# code demonstrates the creation of a payment journal and line. Add this code in the project created earlier and run it with Visual Studio 2010.

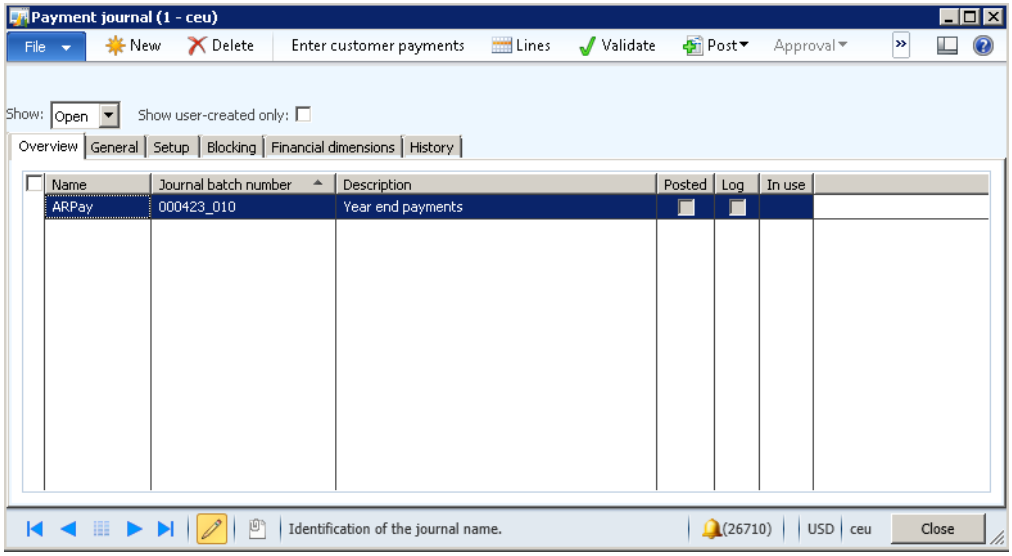
```
// For this demo code,
// The calling context should be switched to company 'CEU'
CallContext callContext = new CallContext { Company = "CEU" };

CustPaymJournalServiceClient serviceClient = new CustPaymJournalServiceClient();
```

```
AxdCustPaymJournal custPaymJournal = null;
EntityKey[] entityKeyList;

// Line amount
int lineAmount = 120;

// Query to find out whether the payment journal with the same name already exists?
QueryCriteria queryCriteria = new QueryCriteria();
queryCriteria.CriteriaElement = new CriteriaElement[1];
queryCriteria.CriteriaElement[0] = new CriteriaElement();
queryCriteria.CriteriaElement[0].DataSourceName = "LedgerJournalTable";
queryCriteria.CriteriaElement[0].FieldName = "Name";
queryCriteria.CriteriaElement[0].Operator = Operator.Equal;
queryCriteria.CriteriaElement[0].Value1 = "Year end payments";

entityKeyList = serviceClient.findKeys(callContext, queryCriteria);


// Create a new journal record if one wasn't found...
if (entityKeyList == null || entityKeyList.Count() == 0)
{
custPaymJournal = new AxdCustPaymJournal();
custPaymJournal.LedgerJournalTable = new AxdEntity_LedgerJournalTable[1];
       custPaymJournal.LedgerJournalTable[0] = new AxdEntity_LedgerJournalTable();
       custPaymJournal.LedgerJournalTable[0].JournalName = "ARPay";
       custPaymJournal.LedgerJournalTable[0].Name = "Year end payments";
       custPaymJournal.LedgerJournalTable[0].JournalTotalDebit = 0.0M;
       custPaymJournal.LedgerJournalTable[0].JournalTotalCredit = 0.0M;
       custPaymJournal.LedgerJournalTable[0].CurrencyCode = "USD";

       entityKeyList = serviceClient.create(callContext, custPaymJournal);
}

// Read the journal record.
// Note: The record must be read in order to do updates against it.
custPaymJournal = serviceClient.read(callContext, entityKeyList);

custPaymJournal.LedgerJournalTable[0].action = AxdEnum_AxdEntityAction.update;
custPaymJournal.LedgerJournalTable[0].actionSpecified = true;

// Build the payment lines (Journal transactions)
List<AxdEntity_LedgerJournalTrans> ledgerJournalTransList = new
List<AxdEntity_LedgerJournalTrans>();

AxdEntity_LedgerJournalTrans ledgerJournalTrans = new AxdEntity_LedgerJournalTrans();
ledgerJournalTrans.action = AxdEnum_AxdEntityAction.create;
ledgerJournalTrans.actionSpecified = true;

ledgerJournalTrans.Company = callContext.Company;
ledgerJournalTrans.Txt = "Payment Transaction";
ledgerJournalTrans.PaymMode = "Cash";
```

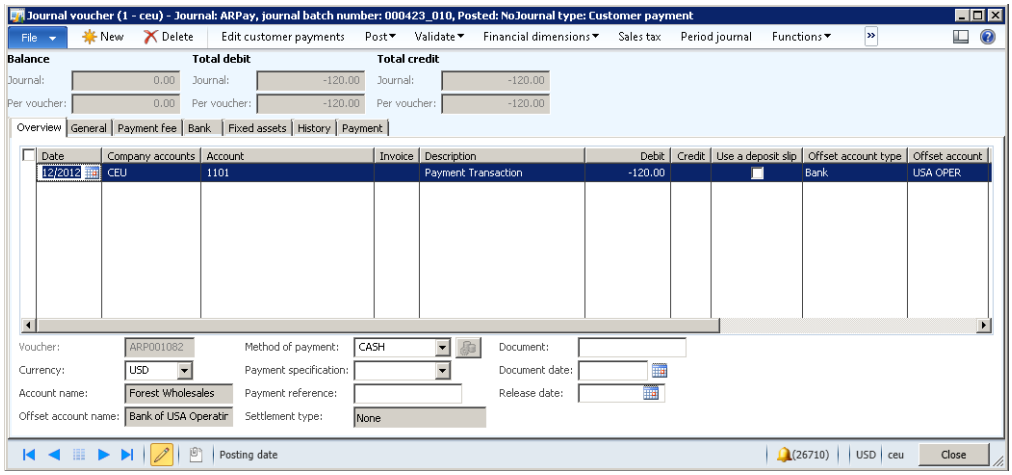
```
ledgerJournalTrans.AccountType = AxdEnum_LedgerJournalACType.Cust;
ledgerJournalTrans.AccountTypeSpecified = true;
ledgerJournalTrans.LedgerDimension = new AxdType_MultiTypeAccount();
ledgerJournalTrans.LedgerDimension.Account = "1101";
ledgerJournalTrans.LedgerDimension.DisplayValue = "1101";
ledgerJournalTrans.LedgerDimension.Values = null;
ledgerJournalTrans.CustTransOpen = null;
ledgerJournalTrans.OffsetCompany = callContext.Company;

if (lineAmount > 0)
{
// Revenue is the primary account type, which is credit balance.
// Therefore if an amount is passed as a negative, it is meant to be a debit...
ledgerJournalTrans.AmountCurDebit = lineAmount * -1;
       ledgerJournalTrans.AmountCurDebitSpecified = true;

       // The JournalTotalDebit and JournalTotalCredit fields need to match the total
// journal amount, not just the amount that we are submitting.
       // add to the existing values only...
       custPaymJournal.LedgerJournalTable[0].JournalTotalDebit += lineAmount * -1;
       custPaymJournal.LedgerJournalTable[0].JournalTotalDebitSpecified = true;
}
else
{
ledgerJournalTrans.AmountCurCredit = lineAmount;
       ledgerJournalTrans.AmountCurCreditSpecified = true;
custPaymJournal.LedgerJournalTable[0].JournalTotalCredit += lineAmount;
custPaymJournal.LedgerJournalTable[0].JournalTotalCreditSpecified = true;
}

ledgerJournalTransList.Add(ledgerJournalTrans);

// Overwrite the ledgerJournalTrans array with our new values.
// This is done because AX doesn't need or want any lines that we are not updating...
custPaymJournal.LedgerJournalTable[0].LedgerJournalTrans = ledgerJournalTransList.ToArray();

// Update the payment journal after creating the payment line
serviceClient.update(callContext, entityKeyList, custPaymJournal);
}
```

After executing this above code, verify the creation of the payment journal and line.

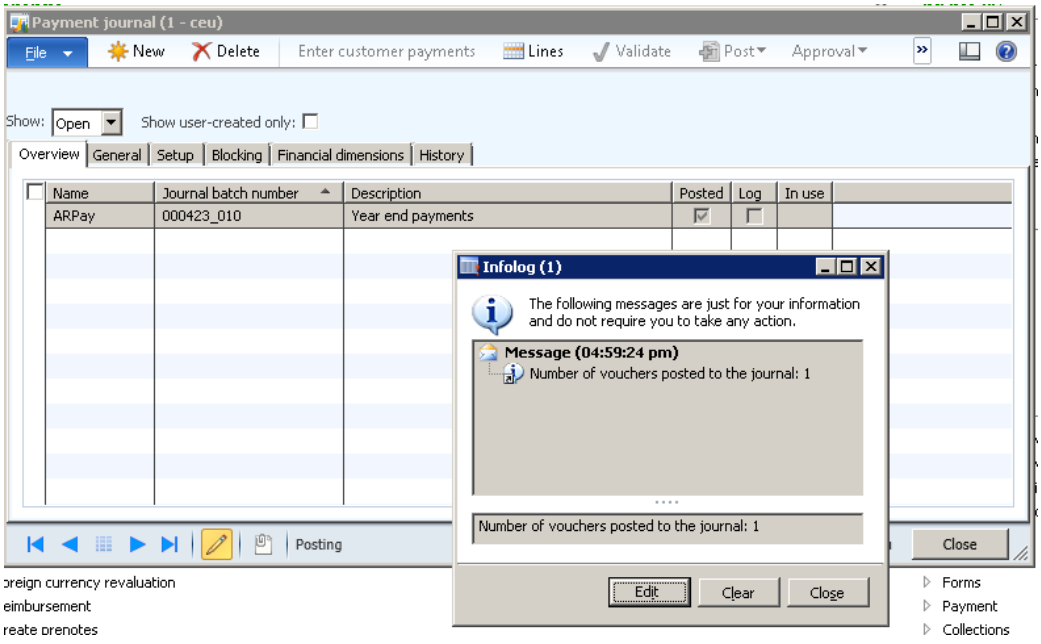
1. Open Microsoft Dynamics AX 2012 and switch the company to **CEU**.
2. Navigate to **Accounts receivable** > **Journals** > **Payments > Payment journal**. The newly-created payment journal is shown in the following illustration.



3. Select the payment journal, click the **Lines** button on the Action pane, and verify that the line has been created with the correct line amount on the **Journal voucher** form.



4. Close the **Journal voucher** form and post the journal by clicking **Post** on the **Payment journal** form.



Marking payments using settlement priorities

The settlePayment operation was added to the Customer payment journal web service to allow the application to mark open transactions for settlement based on system-defined settlement priorities. Consider the following scenario and sample code.

Note: The scenario was developed by using the CEU company in the Contoso dataset.

1. The customer, **2011 – Kiwi Conference Center**, has the following open invoices:

Transaction ID	Due date	Amount (USD)
101039	6/22/2008	350.73
101078	7/27/2008	584.55
101107	8/15/2012	4000000.00

2. The system has the following defined settlement priorities for each transaction type with the lowest number being the highest priority. (For information about enabling and defining settlement priorities, see Appendix A: Configure the settlement priority.)

Priority	Attribute	Active
1	Due Date	Yes
2	Transaction type	Yes
3	Cash discount date	No
4	Transaction date	No

5	Transaction amount	No
6	Voucher	No

3. The customer pays USD 500 and according to the defined settlement priorities, the transactions in the following illustration are marked for settlement.

Transaction ID	Due date	Amount (USD)	Marked
101039	6/22/2008	350.73	Yes
101078	7/27/2008	584.55	Yes (partially)
101107	8/15/2012	4000000.00	No

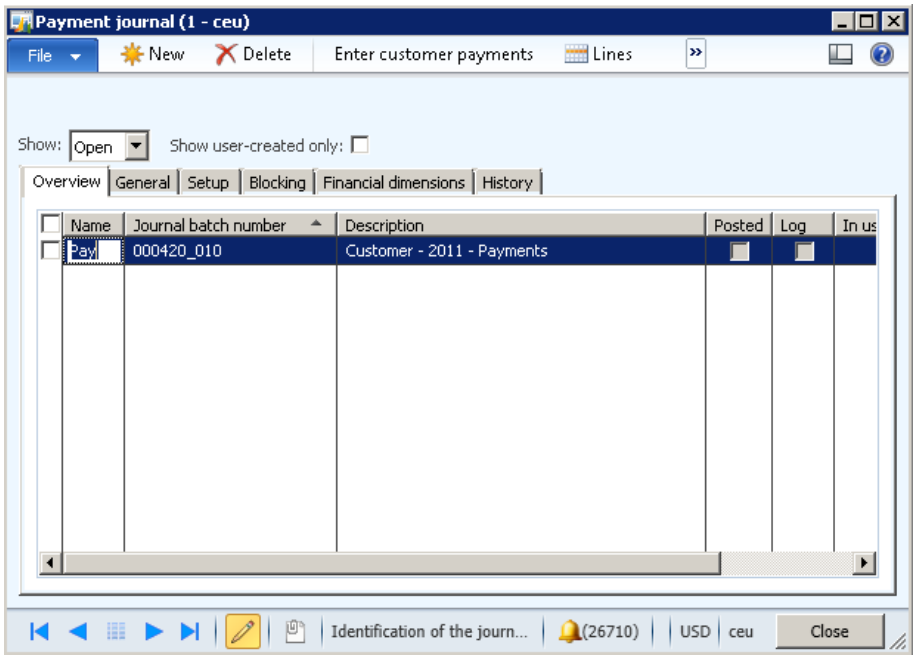
The following sample C# code creates a payment journal and line, and marks open invoices based on the entered line amount of USD 500 for the customer **2011 – Kiwi Conference Center**.

```
// For this demo code,
// The calling context should be switched to company 'CEU'
CallContext callContext = new CallContext { Company = "CEU" };

CustPaymJournalServiceClient serviceClient = new CustPaymJournalServiceClient();

AxdCustPaymJournal custPaymJournal = null;
EntityKey[] entityKeyList;

Int64 paymentToSettleRecId;

// Line amount
int lineAmount = 500;

// Query to find out whether the payment journal with the same name already exists?
QueryCriteria queryCriteria = new QueryCriteria();
queryCriteria.CriteriaElement = new CriteriaElement[1];
queryCriteria.CriteriaElement[0] = new CriteriaElement();
queryCriteria.CriteriaElement[0].DataSourceName = "LedgerJournalTable";
queryCriteria.CriteriaElement[0].FieldName = "Name";
queryCriteria.CriteriaElement[0].Operator = Operator.Equal;
queryCriteria.CriteriaElement[0].Value1 = "Customer - 2011 - Payments";

entityKeyList = serviceClient.findKeys(callContext, queryCriteria);

// Create a new journal record if one wasn't found...
if (entityKeyList == null || entityKeyList.Count() == 0)
{
custPaymJournal = new AxdCustPaymJournal();
       custPaymJournal.LedgerJournalTable = new AxdEntity_LedgerJournalTable[1];
       custPaymJournal.LedgerJournalTable[0] = new AxdEntity_LedgerJournalTable();
       custPaymJournal.LedgerJournalTable[0].JournalName = "ARPay";
       custPaymJournal.LedgerJournalTable[0].Name = "Customer - 2011 - Payments";
       custPaymJournal.LedgerJournalTable[0].JournalTotalDebit = 0.0M;
       custPaymJournal.LedgerJournalTable[0].JournalTotalCredit = 0.0M;
       custPaymJournal.LedgerJournalTable[0].CurrencyCode = "USD";

       entityKeyList = serviceClient.create(callContext, custPaymJournal);
}
```

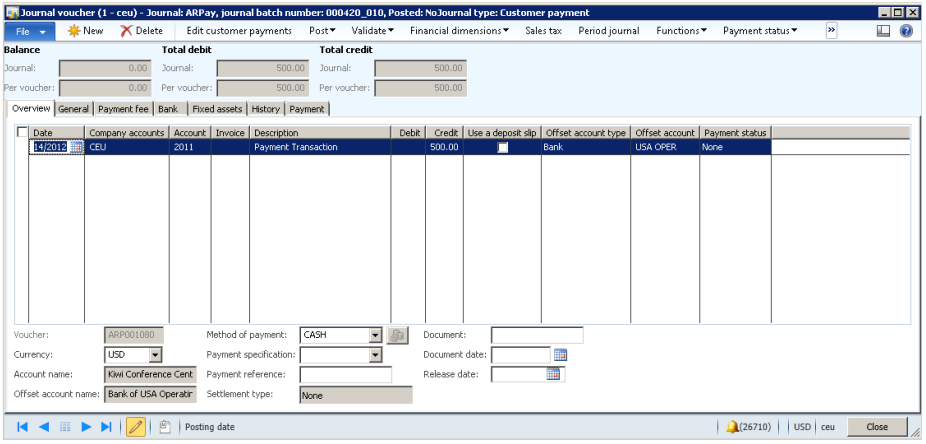
```
// Read the journal record.
// Note: The record must be read in order to do updates against it.
custPaymJournal = serviceClient.read(callContext, entityKeyList);

custPaymJournal.LedgerJournalTable[0].action = AxdEnum_AxdEntityAction.update;
custPaymJournal.LedgerJournalTable[0].actionSpecified = true;

// Build the payment lines (Journal transactions)
List<AxdEntity_LedgerJournalTrans> ledgerJournalTransList = new
List<AxdEntity_LedgerJournalTrans>();

AxdEntity_LedgerJournalTrans ledgerJournalTrans = new AxdEntity_LedgerJournalTrans();
ledgerJournalTrans.action = AxdEnum_AxdEntityAction.create;
ledgerJournalTrans.actionSpecified = true;

ledgerJournalTrans.Company = callContext.Company;
ledgerJournalTrans.Txt = "Payment Transaction";
ledgerJournalTrans.PaymMode = "Cash";
ledgerJournalTrans.AccountType = AxdEnum_LedgerJournalACType.Cust;
ledgerJournalTrans.AccountTypeSpecified = true;
ledgerJournalTrans.LedgerDimension = new AxdType_MultiTypeAccount();
ledgerJournalTrans.LedgerDimension.Account = "2011";
ledgerJournalTrans.LedgerDimension.DisplayValue = "2011";
ledgerJournalTrans.LedgerDimension.Values = null;
ledgerJournalTrans.CustTransOpen = null;
ledgerJournalTrans.OffsetCompany = callContext.Company;
ledgerJournalTrans.AmountCurCredit = lineAmount;
ledgerJournalTrans.AmountCurCreditSpecified = true;
custPaymJournal.LedgerJournalTable[0].JournalTotalCredit += lineAmount;
custPaymJournal.LedgerJournalTable[0].JournalTotalCreditSpecified = true;

ledgerJournalTransList.Add(ledgerJournalTrans);

// Overwrite the ledgerJournalTrans array with our new values.
// This is done because AX doesn't need or want any lines that we are not updating...
custPaymJournal.LedgerJournalTable[0].LedgerJournalTrans = ledgerJournalTransList.ToArray();

// Update the payment journal after creating the payment line
serviceClient.update(callContext, entityKeyList, custPaymJournal);

// re-read the journal to get the newly created objects
custPaymJournal = serviceClient.read(callContext, entityKeyList);
paymentToSettleRecId =
Convert.ToInt64(custPaymJournal.LedgerJournalTable[0].LedgerJournalTrans[0].RecId);

if (custPaymJournal != null)
{
     // Mark the open transaction by passing payment line record ID.
     serviceClient.settlePayment(callContext, paymentToSettleRecId);
}
```

After executing this sample code, verify that the payment journal was created and that the open transactions were marked.

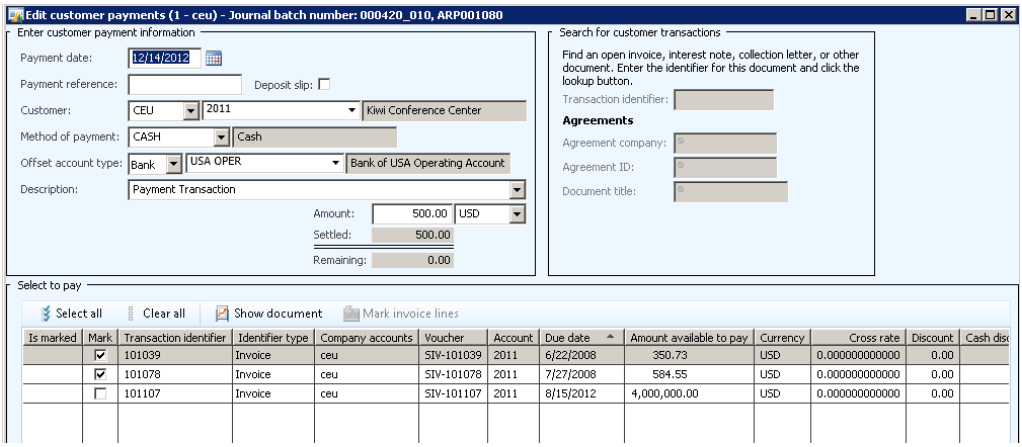
1. Open Microsoft Dynamics AX 2012 and switch the company to **CEU**.
2. Navigate to **Accounts receivable** > **Journals** > **Payments>Payment journal**. The payment journal that you created is shown in the following illustration.



3. Select the payment journal, and then click **Lines** on the Action pane. Verify that the line was created with 500 USD as the line amount.



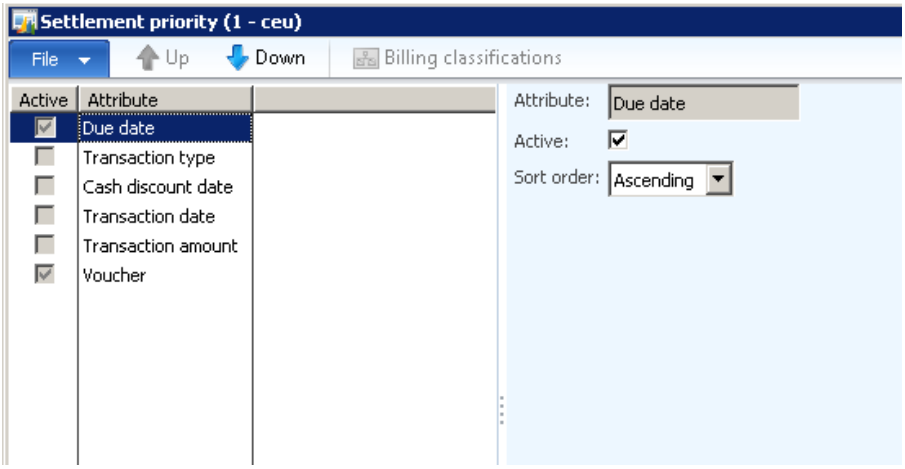
4. Click **Edit customer payments**. Verify that the open transactions were marked for settlement by due date.



Appendix A: Configure the settlement priority

1. Open Microsoft Dynamics AX 2012
2. Navigate to **Accounts receivable** > **Setup** > **Accounts receivable parameters.**
3. Click the **Settlement** tab**.**
4. Select the **Prioritize settlement** check box under the **Priority** group.

5. Click the **Manage priority** link.
6. Select the **Due date** attribute in the left column.
7. Select the **Active** check box in the main content area.



8. Click **Up** to move **Due date** up to the top of the priority order.
9. Close the **Settlement priority** form and the **Accounts receivable parameters** form.

Note: The **Settlement** check box on the **Customer posting profiles** form (**Accounts receivable > Setup > Customer posting profiles**) has no effect on these settings.

Microsoft Dynamics is a line of integrated, adaptable business management solutions that enables you and your people to make business decisions with greater confidence. Microsoft Dynamics works like and with familiar Microsoft software, automating and streamlining financial, customer relationship and supply chain processes in a way that helps you drive business success.

U.S. and Canada Toll Free 1-888-477-7989
Worldwide +1-701-281-6500