

# Web Transaction Tutorial



HELP.BCFESITSTRANTUT

**Release 4.6C**



## Copyright

© Copyright 2001 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft<sup>®</sup>, WINDOWS<sup>®</sup>, NT<sup>®</sup>, EXCEL<sup>®</sup>, Word<sup>®</sup>, PowerPoint<sup>®</sup> and SQL Server<sup>®</sup> are registered trademarks of Microsoft Corporation.

IBM<sup>®</sup>, DB2<sup>®</sup>, OS/2<sup>®</sup>, DB2/6000<sup>®</sup>, Parallel Sysplex<sup>®</sup>, MVS/ESA<sup>®</sup>, RS/6000<sup>®</sup>, AIX<sup>®</sup>, S/390<sup>®</sup>, AS/400<sup>®</sup>, OS/390<sup>®</sup>, and OS/400<sup>®</sup> are registered trademarks of IBM Corporation.

ORACLE<sup>®</sup> is a registered trademark of ORACLE Corporation.

INFORMIX<sup>®</sup>-OnLine for SAP and Informix<sup>®</sup> Dynamic Server<sup>™</sup> are registered trademarks of Informix Software Incorporated.

UNIX<sup>®</sup>, X/Open<sup>®</sup>, OSF/1<sup>®</sup>, and Motif<sup>®</sup> are registered trademarks of the Open Group.






HTML, DHTML, XML, XHTML are trademarks or registered trademarks of W3C<sup>®</sup>, World Wide Web Consortium, Massachusetts Institute of Technology.

JAVA<sup>®</sup> is a registered trademark of Sun Microsystems, Inc.

JAVASCRIPT<sup>®</sup> is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

SAP, SAP Logo, R/2, RIVA, R/3, ABAP, SAP ArchiveLink, SAP Business Workflow, WebFlow, SAP EarlyWatch, BAPI, SAPHIRE, Management Cockpit, mySAP.com Logo and mySAP.com are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other products mentioned are trademarks or registered trademarks of their respective companies.

## Icons

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax

## Inhalt

<b>Web Transaction Tutorial</b> .....	<b>5</b>
Web Transaction Tutorial: Prerequisites .....	7
Web Transaction Tutorial: Basic R/3 Transaction .....	8
ITS Basic Concepts .....	9
Copying the Basic R/3 Transaction .....	10
SAP@Web Studio Environment Setup.....	11
Starting the SAP@Web Studio.....	13
Creating an ITS Project.....	14
Defining an ITS Site.....	15
<b>Lesson 1: Web Transaction Generation</b> .....	<b>16</b>
Running the R/3 Transaction .....	17
Creating the ITS Service .....	19
Creating HTML Templates .....	21
Creating Language Resources .....	23
Publishing the ITS Service .....	25
Testing the Web Transaction .....	26
<b>Lesson 2: Data Transfer with Scrolling</b> .....	<b>27</b>
Adding and Testing the Scrolling Logic in R/3 .....	28
Implementing Scrolling in the Web Transaction.....	30
Tracking the Transaction in the ITS Debugger .....	31
<b>Lesson 3: Mass Data Transfer to the Web Browser</b> .....	<b>32</b>
Adding Mass Data Transfer Logic in R/3 .....	33
Implementing Input Help in the Web Transaction.....	35
<b>Lesson 4: Mass Data Transfer From the Web Browser</b> .....	<b>36</b>
Adding Email Capability to R/3 .....	37
Adding Email Capability to the Web Transaction .....	39
<b>Lesson 5: Multi-Frame Transactions</b> .....	<b>40</b>
Adding the Subscreen Logic in R/3.....	41
Implementing the Multi-Frame Web Transaction .....	43
<b>Lesson 6: Transaction Synchronization</b> .....	<b>44</b>
<b>Synchronizing Single-Frame Transactions</b> .....	<b>46</b>
Handling Single-Frame Synchronization.....	48
<b>Synchronizing Multi-Frame Transactions</b> .....	<b>50</b>
Handling Multi-Frame Synchronization .....	53
<b>Synchronizing Multi-Frame Transactions with Update</b> .....	<b>55</b>
Handling Multi-Frame Synchronization with Update .....	58
<b>Checking ITS Services Into ITS Source Control</b> .....	<b>60</b>

## Web Transaction Tutorial

This tutorial describes how you can create Web transactions using the Web transaction programming model. Web transactions are Internet-enabled R/3 transactions.

Thanks to the Internet Transaction Server (ITS), which links the R/3 System to the Internet, you can turn R/3 transactions into Web transactions, and run them as Internet Application Components (IACs) from any suitable Web browser.

You develop Web transactions both inside and outside the R/3 System:

- Inside R/3, you use the ABAP Workbench to create a transaction by developing an ABAP dialog program.
- Outside R/3, you use the SAP@Web Studio to enable the transaction for Internet use by creating an ITS service, which contains all the components required to implement and run an IAC.

### What the Tutorial Covers

This tutorial is designed to be read as a continuous document. It covers all the steps you have to take to convert R/3 transactions to Web transactions, and demonstrates specific features of interest in the context of Web transaction development.

- Introductory sections cover essential prerequisites and describe how to set up the SAP@Web Studio environment.
- Six lessons covering eight transactions demonstrate how to generate Web transactions from R/3 transactions. In these lessons, you learn how to:
  - Convert an existing R/3 transaction to a fully fledged Web transaction that can be run from any Web browser over the Internet
  - Modify the transaction progressively in order to add more features.

Due to the inevitable differences between running a transaction from the SAPgui and running a transaction from a Web browser, this involves not only amending the ABAP code of the R/3 transaction, but also the HTML code generated for each R/3 screen in the SAP@Web Studio.

- A concluding section shows you how to transfer ITS services created in the SAP@Web Studio to ITS source control in the R/3 System.

### Further Information

For detailed information on all aspects of the ITS, see the following online documentation:

[ITS Administration Guide \[Extern\]](#)

[ITS Implementation Models \[Extern\]](#)

[SAP@Web Studio \[Extern\]](#)

[HTMLBusiness Language Reference \[Extern\]](#)

You can also find this documentation either in the R/3 System or in the SAP@Web Studio:

- In the R/3 System, choose *Help* → *SAP Library* → *Basis Components* → *Frontend Services (BC-FES)* → *ITS / SAP@Web Studio (BC-FES-ITS)*.

---

**Web Transaction Tutorial**

- In the SAP@Web Studio, choose *Help*.

**What the Tutorial Does Not Cover**

This tutorial does **not** teach you how to create transactions in R/3. It assumes that you are familiar with ABAP programming techniques, and that you know how to use the various programming tools available in the ABAP Workbench to develop dialog transactions.

**Further Information**

For detailed information about developing ABAP programs in the ABAP Workbench, see:

[BC - ABAP Programming \[Extern\]](#)

[BC - ABAP Workbench Tools \[Extern\]](#)

[BC ABAP Workbench Tutorial \[Extern\]](#)

## Web Transaction Tutorial: Prerequisites

Before you start working through the tutorial, you should:

- Install the Internet Transaction Server (ITS) that corresponds to the current R/3 release
- Install the SAP@Web Studio that corresponds to the current R/3 release
- Be able to log on to an R/3 System

Further prerequisites include:

- Transporting the sample programs and other ITS files to the R/3 System
- Setting up debugging facilities

If your company has an ITS administrator, most of these tasks have probably been completed. If you need to transport the sample programs and set up debugging facilities yourself, refer to the sections below.

### Transporting the Sample Programs

SAP provides a set sample programs and ITS files, which are complete solutions to the lessons in this tutorial. To be able to examine these programs and files, you must transport them to your R/3 System.

### Setting Up Debugging Facilities

The ITS debugger allows you to start a Web transaction from your Web browser, and follow the processing on the R/3 side in a SAPgui window.

To use the ITS debugger, you must:

- Enable debugging for the active virtual ITS instance
  - You (or your system administrator) can do this in ITS Administration.
  - For further information, see [Enabling and Disabling Debugging \[Extern\]](#).
- Create an entry in the SAP Logon box to be able to log on to the ITS application server
  - When doing this, enter:
    - A brief description
    - The name of the application server
    - The system number
      - The default is 00

## Web Transaction Tutorial: Basic R/3 Transaction

### General Features

The Web Transaction Tutorial uses a basic R/3 transaction as the foundation for creating a Web transaction. In a series of lessons that demonstrate specific features in the development of Web transactions, the tutorial modifies this transaction progressively, and converts each one to a Web transaction.

The basic R/3 transaction (WWY1) is an employee directory, which allows users to look up employees and display information on them. It is based on the dialog program WWIACWY1, which has three screens (100, 200, and 300) covering the following scenario:

#### Employee Directory (Transaction WWY1)

Screen	What the System Does	What the User Does
100	Prompts the user to enter selection criteria to look up employees.	Enters selection criteria to find employees.
200	Displays employees from the database that match the criteria	Enters an employee number to request detailed information.
300	Displays detailed information about the employee with the specified employee number.	Returns to the beginning.

### Special Features

In the PAI module for screen 100, the transaction uses a special function module called BAPI\_EMPLOYEE\_GETDATA to retrieve employee information.

All function modules that have names beginning with the prefix BAPI\_ are known as BAPIs (or Business APIs). BAPIs provide a standard interface to SAP business objects, which are central business entities like employee, customer, material, purchase order, quotation, or request for quotation. A business object collects together all processes and data relevant to the entity within a single object.

The R/3 System contains a large number of business objects, all of which are maintained in the Business Object Repository (BOR). Each business object has a set of BAPIs which are methods used to manipulate the relevant business entity. BAPIs provide an object-oriented view of R/3 that allows external applications to access SAP business functionality.

The function module BAPI\_EMPLOYEE\_GETDATA retrieves information about company employees, and returns it packed into three internal tables called PERSONAL\_DATA, ORG\_ASSIGNMENT, and INTERNAL\_DATA. To access employee information, our transaction merely needs to supply these tables.

For further information about BAPIs, see:

[BAPI Programming Guide \[Extern\]](#)

[BAPI User Guide \[Extern\]](#)

## ITS Basic Concepts

Implementing Web transactions requires some understanding of components and concepts.

### Components

- Internet Transaction Server (ITS)

The ITS is the interface between the Internet and the R/3 System. When an Internet user starts an R/3 transaction from a Web browser, the ITS starts the transaction and generates an HTML page for the first R/3 screen returned. When the user enters data on the HTML page, the ITS passes this data back to R/3. In this way, one HTML page is generated for each R/3 screen.
- Internet Application Components (IACs)

IACs are Internet-enabled dialog transactions, function modules, or reports.

This tutorial describes how to implement Internet-enabled dialog transactions.
- SAP@Web Studio

The SAP@Web Studio is a PC tool for creating, modifying, and managing the ITS components on which IACs are based. ITS components include a service description for each transaction, and HTML templates for each transaction screen. There may also be language resources, and Multipurpose Internet Mail Extension (MIME) objects (such as graphics and images), but these components are optional. Before you can create these components in the SAP@Web Studio, an R/3 transaction must already exist.

### Concept

At runtime, the ITS interfaces between the Web server and the R/3 System to control dialog processing. On each screen change (PBO), the R/3 System sends all data for display on the new screen to the ITS. The ITS merges this data with the relevant HTML template and passes the finished HTML page to the Web server.

When the Internet user enters data on this page, the Web server returns the data to the ITS, which in turn passes it back to R/3. The R/3 System receives this data, merges it with the relevant ABAP fields, and triggers PAI processing.

---

**Copying the Basic R/3 Transaction**

## Copying the Basic R/3 Transaction

To save time and effort in this tutorial, you do not need to create a new transaction for each lesson. Instead, you can copy a program from a previous lesson, rename it, and then modify it as required.

To get started, you can copy and rename the basic R/3 transaction WWIACWY1. From this, you can generate the first Web transaction. Subsequently, you can use your copy of the basic R/3 transaction to develop further Web transactions in the course of the tutorial.

Although this section outlines the main steps for you, the Web Transaction Tutorial assumes that you are familiar with the ABAP Workbench tools and therefore know how to copy and modify ABAP programs.

### Procedure

To copy a program:

1. Log on to the R/3 System.
2. Choose *Tools* → *ABAP Workbench* → *Overview* → *Object Navigator*.
3. Copy the program you want to use as a model, and define it as a local object.

When doing this, copy all screens and other program components.

Since you are building a copy of another program, it's a good idea to give your program a similar name. You can use any name you like, but remember that customer-defined programs must always begin with **y** or **z**.

4. Create a new transaction and specify your new program as the underlying program.

For further information, see:

[BC - ABAP Workbench Tools \[Extern\]](#)

[BC ABAP Workbench Tutorial \[Extern\]](#)

## SAP@Web Studio Environment Setup

### Purpose

Before you can begin to convert your R/3 transactions to Web transactions, you have to set up the environment in the SAP@Web Studio.

The SAP@Web Studio is a PC tool for creating, modifying, and managing all the external components required by the Internet Transaction Server (ITS) to implement R/3 transactions as Web transactions, so that users can run them from a Web browser as fully fledged Internet Application Components (IACs). When you have created all the necessary components, you can transfer them to ITS source control in the R/3 System, and assign them to a change request, just like any other development object.

### Prerequisites

The SAP@Web Studio must be installed, and the transaction(s) must already exist in R/3.

### Process Flow

To set up the environment in the SAP@Web Studio, you need to:

- Create an ITS project
- Define an ITS site

When you have set up the environment, you can begin to implement Web transactions.

For each R/3 transaction you want to convert to a Web transaction, you create an ITS service that comprises some or all of the following components:

- Service description

Each Web transaction has one service description, which contains all the parameters that define how the ITS service is run.
- HTML templates

For each screen in an R/3 transaction, there is a corresponding HTML template in the Web transaction.

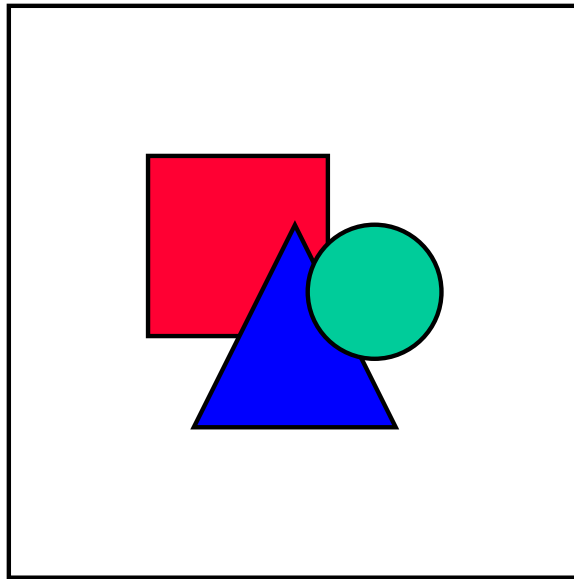
HTML templates contain standard HTML and HTML<sup>Business</sup> statements. HTML<sup>Business</sup> is an SAP-specific macro language, which allows you to merge R/3 data into HTML templates at runtime.
- Language resources

For each language in which an ITS service runs, you can create a language resource file, which contains all the language-specific texts used in the HTML templates.

Language resource files are optional, but it makes sense to use them, because they allow you to create a single set of HTML templates for all languages.
- Multipurpose Internet Mail Extension (MIME) files

MIME files contain any graphics, images, sound or video components that you may want to include in your ITS service. These files are also optional.

---

**SAP@Web Studio Environment Setup**

The Web transactions covered in this tutorial contain service descriptions, HTML templates, and language resources. There are no examples of MIME files.

When you have defined a service by generating the service description, HTML templates, and language resources, you can publish all the files to the ITS site you have defined.

Finally, you can check all the files into ITS source control, where you assign them to a change request in R/3, using the Workbench Organizer.

## **Result**

The result in each case is a Web transaction that can be started from a Web browser.

## Starting the SAP@Web Studio

In this exercise, you learn how to start the SAP@Web Studio.

### Prerequisites

The SAP@Web Studio must be installed.

### Procedure

On the Windows NT desktop, choose *Start* → *Programs* → *SAP@Web Studio* → *Studio* <release number>.

### Result

You see the SAP@Web Studio environment.

Now, you can [create an ITS project \[Seite 14\]](#) and [define an ITS site \[Seite 15\]](#).

---

## Creating an ITS Project

### Creating an ITS Project

In this exercise, you learn how to create an Internet Transaction Server (ITS) project for all the Web transactions you create in this tutorial, and how to open existing projects.

An ITS project is a directory that you create in the SAP@Web Studio to store the components of your Web transaction(s) while you work on them. When you open a project, it becomes the default location for any components you create.

### Procedure

#### Creating a New ITS Project

To create a new ITS project:

1. Start the SAP@Web Studio as described in [Starting the SAP@Web Studio \[Seite 13\]](#).
2. Choose *File* → *New* and select the *Projects* tab.
3. Enter a name for the project in the *Project Name* field and choose *OK*.

The SAP@Web Studio creates your project, and automatically opens it.

#### Opening an Existing ITS Project

To open an existing ITS project:

1. Start the SAP@Web Studio as described in [Starting the SAP@Web Studio \[Seite 13\]](#).
2. Choose *File* → *Open Project* and browse for the project you want, or choose *File* → *Recent Projects* and select a project from the list.

The SAP@Web Studio displays your project in the *File View* and *IAC View* tabs.

## Defining an ITS Site

In this exercise, you learn how to define an Internet Transaction Server (ITS) site.

An ITS site is the location to which you publish (transfer) all the components you create in the SAP@Web Studio to implement a Web transaction.

- The location is an R/3 System
- The components are files
  - Dynamic files such as service descriptions, HTML templates, and language resources are stored in the appropriate ITS server directories.
  - Static files such as Multipurpose Internet Mail Extension (MIME) files are stored in the Web server directory.

During the development stage, the SAP@Web Studio uses the ITS site to access all the information it needs to generate a service.

## Procedure

To define an ITS site:

1. Start the SAP@Web Studio as described in [Starting the SAP@Web Studio \[Seite 13\]](#).
2. Choose *Project* → *Site Definition*.
3. Choose *New*.
4. Enter a site name, and choose *Next*.
5. Enter the Web server host name, and choose *Next*.
6. Enter the ITS server host name, and choose *Next*.
7. Enter an ITS virtual instance name, and choose *Next*.
8. Choose *Finish* to confirm
9. Choose *OK* to exit.

## Result

The ITS site is displayed in the list box on the left of the application toolbar.

---

**Lesson 1: Web Transaction Generation**

## Lesson 1: Web Transaction Generation

In this lesson, you learn how to convert an existing R/3 transaction to a Web transaction by creating a service that can be started from a Web browser and run as an Internet Application Component (IAC).

The steps covered in this lesson include:

- [Running the R/3 Transaction \[Seite 17\]](#)
- [Creating the ITS Service \[Seite 19\]](#)
- [Creating HTML Templates \[Seite 21\]](#)
- [Creating Language Resources \[Seite 23\]](#)
- [Publishing the ITS Service \[Seite 25\]](#)
- [Testing the Web Transaction \[Seite 26\]](#)

For an illustration of the complete implementation, see sample program WWIACWY1 (transaction WWY1).

## Running the R/3 Transaction

Before you can convert your R/3 transaction to a Web transaction, the transaction must already exist in the R/3 System.

As a first step, it is a good idea to try out the transaction you intend to implement as a Web transaction by running it in R/3.

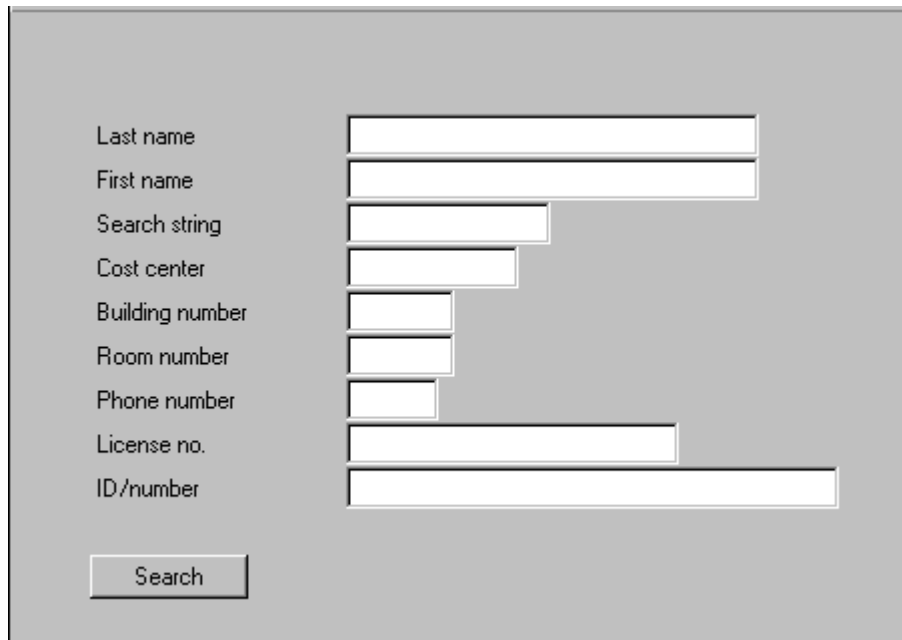
The example transaction WWY1 (described in [Web Transaction Tutorial: Basic R/3 Transaction \[Seite 8\]](#)) allows you to look up employee information in an employee directory.

### Procedure

To run transaction WWY1 in R/3:

1. Log on to the R/3 System that contains the sample programs.
2. In the command field, enter **wwy1** to start the transaction.

The system displays screen 100, which allows you to enter selection criteria.



Last name	<input type="text"/>
First name	<input type="text"/>
Search string	<input type="text"/>
Cost center	<input type="text"/>
Building number	<input type="text"/>
Room number	<input type="text"/>
Phone number	<input type="text"/>
License no.	<input type="text"/>
ID/number	<input type="text"/>

3. In the *Last name* field, enter a generic value (for example, **a\***) to generate a long list of names, and choose *Search*.

The system displays screen 200, which contains a list of all employees that match your selection criteria.

Running the R/3 Transaction

Personnel no.

Pers.no.	Last name	First name	Phone
1006	Auermann	Janine	4626
1037	Anselm	Karin	8834
1329	Apfel	Kerstin	7476
1700	Anton	Karl	8401
8021	Austin	Martin	1799
10044	Anderson	George	9727
10100	Abbott	Frank	5178

- In the *Personnel no.* field, enter one of the listed personnel numbers, and choose *Detail*.

The system displays screen 300, which contains detailed information about the employee whose personnel number you entered.

Personnel no.	1006
Last name	Auermann
First name	Janine
Text	Accounts Payable (D)
Job title	Functional specialist
Activity	FI Agent Release level 3
License no.	TXL-UK 8721
Phone number	4626
Phone number	7694
Changed on	29.01.1997
Last changed by	MIERZWA
Personnel area	1300
EE_group	1

## Creating the ITS Service

In this exercise, you learn how to create the Internet Transaction Server (ITS) service for the R/3 transaction you are implementing as a Web transaction, and how to specify an ITS theme for the service. You do this in the SAP@Web Studio.

- An ITS service is the set of external components that make up a Web transaction. These components always include a service description and HTML templates to match the R/3 transaction screens, and may also contain language resources and Multipurpose Internet Mail Extension (MIME) files.

Each ITS service is defined by the ITS service description, which specifies essential parameters such as the R/3 transaction to be called, the R/3 System to be used, logon information such as client, user name, password, and logon language, as well as any other parameters needed to run the transaction.

- An ITS theme is an instance of an ITS service with its own set of HTML templates and language resources. An ITS theme lends an ITS service a particular look and feel, but the functionality remains the same.

### Prerequisites

You must have created an ITS project.

### Procedure

#### Creating the ITS Service

To create the ITS service, use the SAP@Web Studio Service Wizard:

1. If it is not already running, start the SAP@Web Studio as described in [Starting the SAP@Web Studio \[Seite 13\]](#).
2. Choose *File* → *New* and select the *File Wizards* tab.
3. Select *Service Wizard* and choose *OK*.

You see the *SAP@Web Service Wizard* initial screen.

4. Choose *Next*.
5. In the *Service Name* field, enter the service name and choose *Next*.
6. Select an R/3 System and choose *Next*.
7. Select *Specify a service specific login*.
8. Enter values in the *Client*, *User*, *Password*, and *Language* fields, and choose *Next*.

Remember to use the correct length language key.

- R/3 3.x releases require a 1-character language ID (for example, **E** for English),
- R/3 4.x releases require a 2-character language ID (for example, **EN** for English).

9. Enter 10 in the *Timeout (min)* field and choose *Next*.
10. Select *Use WebTransaction* and enter the name of the R/3 transaction.

### Creating the ITS Service

11. Choose *Next*.
12. Choose *Finish* to confirm.

This creates a node entry for the new service under the ITS project in the SAP@Web Studio's *File View*.

### Creating an ITS Theme

To create an ITS theme for your ITS service:

1. If it is not already running, start the SAP@Web Studio as described in [Starting the SAP@Web Studio \[Seite 13\]](#).
2. Choose the *File View* or *IAC View* tab and double-click on the relevant service name.  
You see the service description in the SAP@Web Studio's work area. This file contains all the parameters so far defined to run the service from a Web browser.
3. Place the cursor on any empty row, and double-click.  
The SAP@Web Studio displays the *String Properties* dialog box.
4. In the *Key* field, change `~param` to `~theme`.
5. In the *Value* field, enter `99`, and then press `Enter`.
6. Save your work.

## Creating HTML Templates

In this exercise, you learn how to create HTML templates for your Internet Transaction Server (ITS) service, and check the results. You do this in the SAP@Web Studio.

HTML templates correspond to the screens of your R/3 transaction. Each HTML template contains standard HTML code and HTML<sup>Business</sup> statements. At runtime, the ITS merges R/3 screen data, R/3 texts, R/3 error messages, non-R/3 texts and graphics, and dynamically generated URLs into each template by interpreting the HTML<sup>Business</sup> statements. This produces a version of the R/3 screen to be displayed in the user's Web browser.

Depending on the Web transaction you are implementing, you may need to modify the automatically generated HTML code in the SAP@Web Studio editor.

### Procedure

#### Creating HTML Templates

To create HTML templates for your ITS service, use the SAP@Web Studio Template Wizard:

1. If it is not already running, start the SAP@Web Studio as described in [Starting the SAP@Web Studio \[Seite 13\]](#).
2. Choose *File* → *New* and select the *File Wizards* tab.
3. Select *Template Wizard* and choose *OK*.  
You see the *SAP@Web Template Wizard* initial screen.
4. Choose *Next*.
5. Select the R/3 System from which the screens for the transaction should be taken, and choose *Next*.
6. Enter values in the *Client*, *User*, *Password*, and *Language* fields, and choose *Next*.  
The Studio prompts you for information about the web transaction.
7. Enter information about the Web transaction you are creating:
  - In the *Program* field, enter the program name of the R/3 transaction
  - In the *Screen Number* field, enter the screen numbers used by the above program  
Separate each screen number with a semicolon. For example, 100 ; 200 ; 300.
  - In the *Service* field, enter the ITS service name.  
This is the service name you specified in [Creating the ITS Service \[Seite 19\]](#).
  - In the *Theme* field, enter the ITS theme  
The default theme is 99. Unless you are using multiple themes, or you want to use a different name, you do not have to change the default value.
8. Choose *Next*.
9. Choose *Finish* to confirm.

The SAP@Web Studio logs on to the R/3 System, retrieves the transaction information needed to create the HTML templates, and downloads them to your PC.

## Creating HTML Templates

### Checking the Results

To view the HTML templates generated for your ITS service in the SAP@Web Studio:

1. Select the *File* tab and expand the hierarchy for the relevant ITS service and theme.

HTML template files have the format

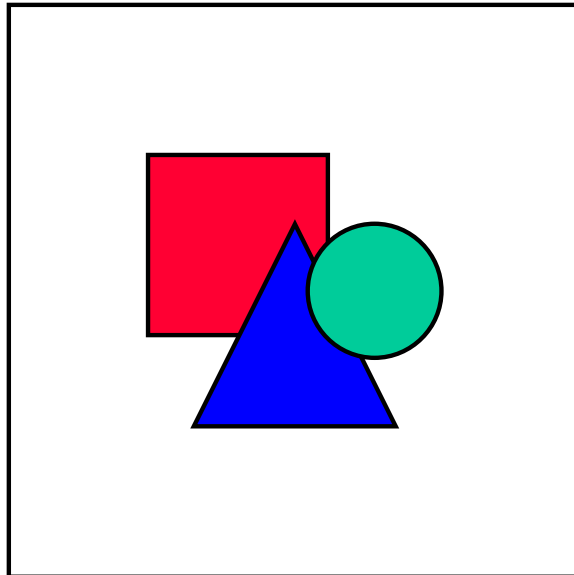
```
<program name>_<screen number>.html
```

2

The SAP@Web Studio displays the HTML code for the template in the work area.

3. Scroll through the file.

Notice the HTML code in blue. This is the special HTML<sup>Business</sup> code added to help the Internet Transaction Server (ITS) transfer data between the R/3 System and the Web server.



HTML<sup>Business</sup> is SAP's extension to standard HTML. For further information, see [HTMLBusiness Language Reference \[Extern\]](#).

## Creating Language Resources

In this exercise, you learn how to create language resources for your Internet Transaction Server (ITS) service. You do this in the SAP@Web Studio.

Language resources allow you to keep your ITS services language independent. A language resource file contains all the texts required to run an ITS service in a particular language. For each text, you specify a placeholder and the appropriate text in the language resource file. You then use the placeholders in HTML templates instead of the texts themselves. This means that you can create a single set of HTML templates for all languages, rather than a separate set for each language. At runtime, the ITS detects the placeholders in each template and replaces them with the correct language text from the relevant language resource file.



Not all ITS services use language resources. Some HTML templates contain hard-coded language texts and are therefore language-dependent. To keep your ITS services as flexible as possible, you should always use language resources.

Unless you decide to work with language-specific HTML templates, you must create at least one language resource file for the language in which the Web transaction is being developed.

## Procedure

### Creating a Language Resource File

To create a language resource file for your ITS service, use the SAP@Web Studio Resource Wizard:

3. If it is not already running, start the SAP@Web Studio as described in [Starting the SAP@Web Studio \[Seite 13\]](#).
2. Choose *File* → *New* and select the *File Wizard* tab.
3. Select *Resource Wizard* and choose *OK*.
4. Enter values for *Service*, *Theme*, and *Language*.

Remember to use the correct length language key.

- R/3 3.x releases require a 1-character language ID (for example, **E** for English),
- R/3 4.x releases require a 2-character language ID (for example, **En** for English).

Choose *Next*.

You see a list of all existing language resource files for this ITS service. There may be none.

5. Select all existing language resource files for import and choose *Next*.
6. Choose *Finish*.

This creates an entry for the new language resource file under the relevant service in the SAP@Web Studio's *File View*.

## Creating Language Resources

### Adding Placeholders and Language Texts

To add the language-specific texts to the language resource files generated for your ITS service in the SAP@Web Studio:

1. Select the *File* tab and expand the hierarchy for the relevant ITS service and theme.

Language resource files have the format

```
<service name>_<language>.htrc
```

4. Double-click on any language resource file.

The SAP@Web Studio displays the language resource file in the work area. If you have just created the file, it should be empty.

3. Place the cursor on any empty row, and double-click.

The SAP@Web Studio opens an edit box in the *Name* column. This box contains the value `param<number>`, where `number` corresponds to the row number.

4. Enter the key value for the language text.

This is the placeholder you specify in HTML templates instead of the language text.

5. Tab to the *Value* field, and enter the text value in the relevant language
6. Save your work.

## Publishing the ITS Service

In this exercise, you learn how to tie all the components of your Internet Transaction Server (ITS) service together to create your Web transaction by publishing (transferring) the ITS service to an ITS site. You do this in the SAP@Web Studio.

### Procedure

Publish your new ITS service to an ITS site, and test it immediately:

1. Choose *Project* → *Set Active Service* and select the relevant service.

This makes the service you select the currently active service and activates the *Publish*, *Build*, and *Go* buttons in the toolbar immediately above the work area.

2. Choose *Options* → *Studio Properties* and select the *Web Browser* tab.
3. Select *Open Browser window in Studio* (if not already selected) and choose *OK*.

This allows you to run the new Web transaction directly from the SAP@Web Studio.

4. Choose *Go*.

When you do this, the SAP@Web Studio:

- Builds your ITS service
- Publishes your ITS service files to the ITS site you defined earlier
- Starts your Web browser
- Calls the new Web transaction

The SAP@Web Studio displays a brief log of these actions in the message area immediately below the work area.

You then see the first screen of the Web transaction in the Web browser. This screen is the HTML template that corresponds to the first screen of the R/3 transaction.

Depending on what you want to achieve, you can also use the *Build* and *Publish* buttons separately.

---

**Testing the Web Transaction**

## Testing the Web Transaction

When you have made your ITS service the active service and started it in the SAP@Web Studio, you can test it.

### Procedure

To run the Web transaction WWY1 in the SAP@Web Studio:

1. Choose *Go*.

You see screen 100 in your Web browser.

2. In the *Last name* field, enter a generic value (for example, *a\**) to generate a long list of names, and choose *Search*.

The Web browser displays the HTML page that corresponds to R/3 screen 200 in transaction WWY1.

3. In the *Personnel no.* field, enter one of the listed personnel numbers, and choose *Detail*.

The Web browser displays the HTML page that corresponds to R/3 screen 300 in transaction WWY1.

4. To return, click the right mouse button and choose *Back*.

To test the language resources, you need to change the service language, and run the transaction again.

## Lesson 2: Data Transfer with Scrolling

In this lesson, you learn how to extend the Web transaction you created in *Lesson 1* in order to enable scrolling in the Web browser.

The original version of the program cannot display more than a single page of data, so you have two options for enhancements:

- Add scrolling logic to the ABAP program

In a non-scrolling screen, the Internet Transaction Server (ITS) retrieves only as much data as can be handled in a single step loop.

If you add scrolling logic to the ABAP program, the ITS retrieves all the data in the step loop tables. As a result, the Web browser displays scroll bars in the resulting HTML page, thus allowing the user to scroll through all the employees selected.

This lesson shows you how to implement scrolling logic.

- Use the ITS mass data channel

When handling large amounts of data, you can bypass normal screen processing by transferring data via the ITS mass data channel.

This option is covered in [Lesson 3: Mass Data Transfer to the Web Browser \[Seite 32\]](#).

For an illustration of the complete implementation, see sample program WWIACWY2 (transaction WWY2).

## Adding and Testing the Scrolling Logic in R/3

# Adding and Testing the Scrolling Logic in R/3

To implement scrolling logic in the ABAP program, you can use a step loop. The code changes here add two function keys - F21 for scrolling to the top of the file, and F23 for scrolling down one page.

## Procedure

### Adding the Scrolling Logic

1. Copy program WWIACWY1 (or your copy of this program) to a new program, as described in [Copying the Basic R/3 Transaction \[Seite 10\]](#).

Make sure that your program name conforms to the usual naming conventions.

2. Declare a global step loop variable called STEPLOOP\_SIZE with TYPE I.

```
DATA:   ORG_ASSIGNMENT   LIKE BAPIP0001T OCCURS 0 WITH HEADER
LINE.
DATA:   PERSONAL_DATA    LIKE BAPIP0002  OCCURS 0 WITH HEADER
LINE.
DATA:   INTERNAL_CONTROL LIKE BAPIP0032  OCCURS 0 WITH HEADER
LINE.

DATA:   STEPLOOP_SIZE    TYPE I.
DATA:   FCODE            LIKE SY-UCOMM.
DATA:   EMPLOYEE_IDX     TYPE I.
```

3. In the PBO processing for screen 200, the program calls the module READ\_EMPLOYEE to select the next screen page of employees to display. Modify this code to set the value of STEPLOOP\_SIZE.

```
MODULE READ_EMPLOYEE OUTPUT.
  PERFORM READ_EMPLOYEE_INTO_WA USING EMPLOYEE_IDX.
  STEPLOOP_SIZE = SY-LOOPC.
  IF SY-SUBRC <> 0.
    EXIT FROM STEP-LOOP.
  ENDIF.
ENDMODULE.                                " PROCESS_EMPLOYEE OUTPUT
```

4. The PAI module USER\_COMMAND\_200 responds to function codes. Add the code to handle the function codes FT21 (*First page* button) and FT23 (*Next page* button).

```
CASE FCODE.
  WHEN 'DETA'.
    PERFORM READ_EMPLOYEE_WITH_KEY_INTO_WA USING ZZIACWXX-
SEL_PERNR.
    CLEAR ZZIACWXX-SEL_PERNR.
    IF SY-SUBRC = 0.
      SET SCREEN 300.
    ELSE.
      MESSAGE W002.
    ENDIF.
  WHEN 'FT23'.
    EMPLOYEE_IDX = EMPLOYEE_IDX + STEPLOOP_SIZE.
  WHEN 'FT21'.
```

## Adding and Testing the Scrolling Logic in R/3

```
EMPLOYEE_IDX = 1.  
WHEN 'BACK'.  
SET SCREEN 100.  
ENDCASE.
```

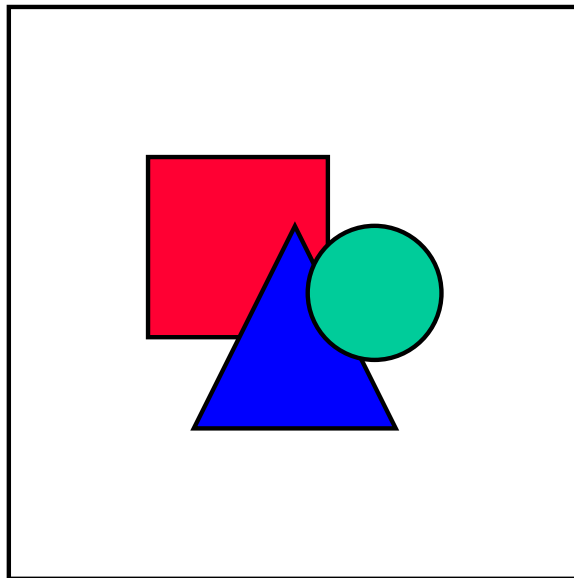
5. In the Menu Painter, modify the GUI status BACK to enable the function keys FT21 (*First page*) and FT23 (*Next page*).

### Testing the Scrolling Logic

1. Generate the program, screens and user interface.
2. Start the transaction.
3. In the *Last Name* field, enter \* as selection criterion to generate more than one screen page of employees.

The system displays screen 200. Notice that the F21 and F23 buttons are active.

4. Test these buttons to see how they work.



This example applies only to display scenarios.

---

## Implementing Scrolling in the Web Transaction

# Implementing Scrolling in the Web Transaction

When you have added and tested the scrolling logic in R/3, you can generate and test the Web transaction in the SAP@Web Studio.

## Procedure

### Generating the Web Transaction

1. Create the ITS service for the transaction, as described in [Creating the ITS Service \[Seite 19\]](#).
2. Create the HTML templates for the transaction, as described in [Creating HTML Templates \[Seite 21\]](#).
3. Create any language resources required for the transaction, as described in [Creating Language Resources \[Seite 23\]](#).

Although scope for language resources is limited in this case, the sample language resource files provided with the sample transaction demonstrate their purpose.

4. Make this ITS service the currently active ITS service, and publish it to your ITS site, as described in [Publishing the ITS Service \[Seite 25\]](#).

### Testing the Web Transaction

1. In the *Last name* field, enter \* as selection criterion to generate more than one screen page of employees.

You get a scrollable list of employees.

2. Check the HTML template.

Notice that the HTML code has not changed from the non-scrolling version, but the Internet Transaction Server (ITS) detects that the function keys F21 and F23 have been enabled, and automatically triggers scrolling in the HTML template.

## Tracking the Transaction in the ITS Debugger

You can start your Web transaction in the Web browser, and follow the processing on the R/3 side in the Internet Transaction Server (ITS) debugger at the same time.

### Prerequisites

Your current virtual ITS instance must be enabled for debugging, and your R/3 System must be set up to run the ITS debugger, as described in [Web Transaction Tutorial: Prerequisites \[Seite 7\]](#).

### Procedure

1. Start the transaction in your Web browser.  
You see the HTML template for the first screen.
  2. In the SAP Logon box, log on to the ITS debugger (that is, the R/3 application server for which ITS debugging has been enabled)  
You see the first screen of the transaction in the R/3 SAPgui window.
  3. In the Web browser, enter \* as selection criterion in the *Last name* field to generate more than one screen page of employees, and choose *Search*.
  4. Switch to the SAPgui window.  
You see screen 200, where the resulting list of employees is automatically scrolling.
  5. In the Web browser, enter a personnel number, and choose *Detail*.
  6. Switch to the SAPgui window.  
You see screen 300, which displays detailed data for the specified employee.
- On the R/3 side, you can switch on debugging any time by entering `/h` in the command field.

---

**Lesson 3: Mass Data Transfer to the Web Browser**

## Lesson 3: Mass Data Transfer to the Web Browser

In this lesson, you learn how to implement mass data transfer from the R/3 System to the Web browser using the Internet Transaction Server (ITS) mass data channel.

The original version of the program cannot display more than a single page of data, so you have two options for enhancements:

- Add scrolling logic to the ABAP program

In a non-scrolling screen, the Internet Transaction Server (ITS) retrieves only as much data as can be handled in a single step loop.

If you add scrolling logic to the ABAP program, the ITS retrieves all the data in the step loop tables. As a result, the Web browser displays scroll bars in the resulting HTML page, thus allowing the user to scroll through all the employees selected.

This option was covered in [Lesson 2: Data Transfer with Scrolling \[Seite 27\]](#).

- Use the ITS mass data channel

If you want to transfer large amounts of data, it is better to use the ITS mass data channel, which uses Remote Function Call (RFC) as its communication method instead of the DIAG channel. This way, you bypass normal screen processing completely by avoiding the usual dialog steps.

Screen processing **does** occur, but using the SAPgui for debugging purposes is difficult, because the mass data is not actually visible on the screen.

Lesson 3 covers this option.

For an illustration of the complete implementation, see sample program WWIACWY3 (transaction WWY3).

## Adding Mass Data Transfer Logic in R/3

As an example of mass data transfer from R/3 to the Web browser, we can implement input help for the *Cost Center* field in screen 100.

### Procedure

To implement mass data transfer, you need to:

1. Copy the program you created in Lesson 2 (or the sample program WWIACWY2) to a new program, as described in [Copying the Basic R/3 Transaction \[Seite 10\]](#).

Make sure that your program name conforms to the usual naming conventions.

2. Modify screen 100.
3. Extend the ABAP code.

### Modifying Screen 100

In the PBO flow logic, add two new module calls:

- GET\_INPUT\_HELP and FIELD-TRANSPORT

```
PROCESS BEFORE OUTPUT.
  MODULE STATUS_0100.
  MODULE GET_INPUT_HELP.
  MODULE FIELD-TRANSPORT.
```

### Extending the ABAP Code

1. To collect the cost center data, add a new internal table called COSTCENTER\_TAB:

```
* Internal table for drop down box cost centers
DATA: COSTCENTER_TAB_INITIALIZED VALUE 0.
DATA: BEGIN OF COSTCENTER_TAB OCCURS 100,
      COSTCENTER LIKE BAPIEMPL-KOSTL,
      DESCRIPTION LIKE M_KOSTS-MCTXT,
      END OF COSTCENTER_TAB.

* View for cost centers
TABLES: M_KOSTS.
```

2. In the module GET\_INPUT\_HELP, select the cost centers from the database. To transfer the cost center data, use the FIELD-SET macro, which is defined in the include AVWRTCXM.

```
FORM GET_INPUT_HELP.
  DATA: COUNT TYPE I.
  DATA: ROW TYPE I.

  IF COSTCENTER_TAB_INITIALIZED = 0.
    CLEAR COSTCENTER_TAB.
    SELECT KOSTL MCTXT FROM M_KOSTS
    INTO (COSTCENTER_TAB-COSTCENTER, COSTCENTER_TAB-DESCRIPTION)
    WHERE SPRAS EQ SY-LANGU
      AND KOKRS EQ '1000'.
    COLLECT COSTCENTER_TAB.
  ENDSELECT.
```

### Adding Mass Data Transfer Logic in R/3

```
    SORT COSTCENTER_TAB BY COSTCENTER.  
    COSTCENTER_TAB_INITIALIZED = 1.  
ENDIF.  
  
DESCRIBE TABLE COSTCENTER_TAB LINES COUNT.  
IF COUNT GT 0.  
    ROW = 0.  
    LOOP AT COSTCENTER_TAB TO COUNT.  
        ROW = ROW + 1.  
        FIELD-SET 'CostCenter' ROW COSTCENTER_TAB-COSTCENTER.  
        FIELD-SET 'Description' ROW COSTCENTER_TAB-DESCRIPTION.  
    ENDLOOP.  
ENDIF.  
  
ENDFORM.                " GET_INPUT_HELP
```

FIELD\_SET places the data in temporary storage as rows of an internal table. The first argument in each macro statement specifies the field name used by HTML to reference the same data.

3. Since the data stored by FIELD-SET is not transferred immediately, your program must also call the macro FIELD-TRANSPORT:

```
MODULE FIELD-TRANSPORT OUTPUT.  
    FIELD-TRANSPORT.  
ENDMODULE.                " FIELD-TRANSPORT OUTPUT
```

At runtime, FIELD-TRANSPORT flushes all the data previously stored by FIELD-SET, and sends it over the mass data channel.

## Results

If you run the transaction in R/3, you won't see any difference. Since we have not added a dropdown box to the R/3 screen, the *Cost Center* field looks just the same as before.

## Implementing Input Help in the Web Transaction

When you have added the mass data transfer logic to the R/3 transaction, you can generate and test the Web transaction in the SAP@Web Studio. To implement this in the Web transaction, you also need to modify the HTML code in the template for screen 100.

### Procedure

#### Generating the Web Transaction

5. Create the ITS service for the transaction, as described in [Creating the ITS Service \[Seite 19\]](#).
6. Create the HTML templates for the transaction, as described in [Creating HTML Templates \[Seite 21\]](#).
7. In the HTML template for screen 100, delete the automatically generated `<input>` statement for the *Cost Center* field, and replace it with `<select>` code that reads values from the mass data channel table:

```
`BAPIEMPL-KOSTL.label`  
<select name="BAPIEMPL-KOSTL">  
  `repeat with i from 1 to Costcenter.dim`  
    <option value="`costcenter[i]`"> `description[i]`  
  `end`  
</select>
```

8. Create any language resources required for the transaction, as described in [Creating Language Resources \[Seite 23\]](#).  
Although scope for language resources is limited in this case, the sample language resource files provided with the sample transaction demonstrate their purpose.
9. Make this ITS service the currently active ITS service, and publish it to your ITS site, as described in [Publishing the ITS Service \[Seite 25\]](#).

### Result

When the Web browser displays the first screen of your transaction, the *Cost Center* field has a dropdown box allowing you to select values.

---

**Lesson 4: Mass Data Transfer From the Web Browser**

## Lesson 4: Mass Data Transfer From the Web Browser

In this lesson, you learn how to implement mass data transfer from the Web browser to the R/3 System by allowing users to send email from the detail screen (screen 300), using the Internet Transaction Server (ITS) mass data channel.

The ITS mass data channel can transfer large amounts of data quickly, because it bypasses normal screen processing completely.

To implement email capability, you need to:

- Add an input text field and pushbutton to screen 300, so that the user can enter a message and send it to the specified recipient
- Add an internal table to the ABAP code to hold the message text
- Add code to read the message text from the ITS and send it to the intended recipient
- Add a multi-line text field to the HTML template

For an illustration of the complete implementation, see sample program WWIACWY4 (transaction WWY4).

## Adding Email Capability to R/3

As an example of mass data transfer from the Web browser to R/3, we can implement email capability on screen 300 to allow the user to enter a short message and send it to the employee whose data is displayed.

### Procedure

1. Copy the program you created in Lesson 3 (or the sample program WWIACWY3) to a new program, as described in [Copying the Basic R/3 Transaction \[Seite 10\]](#).

Make sure that your program name conforms to the usual naming conventions.

2. Modify screen 300 in the Screen Painter:
  - a. Define an input text field called SOOD1-OBJDES
  - b. This field is meant to contain the title of the mail message.
  - c. Define a pushbutton to allow the user to send the message.
  - d. Define a group box called MAIL to hold these screen elements
3. Declare two internal tables MESSAGE\_TEXT and MESSAGE\_LINE.

```
tables:
  message_text like soli occurs 0 with header line,
  message_line like soli occurs 0 with header line.
```

4. Declare the table SOOD1.

```
data:
  sood1.
```

5. Add code to the USER\_COMMAND\_300 module to respond to send requests by reading in the message text and sending it.

To read the message from the Internet Transaction Server (ITS), use the FIELD-GET macro to get each line of text.

To send the message to its recipient, call the function module Z\_SEND\_MESSAGE:

```
CASE FCODE.
  WHEN 'BACK'.
    SET SCREEN 200.
  WHEN 'SEND'.
    DO.
      FIELD-GET 'MESSAGE_TEXT' SY-INDEX MESSAGE_LINE
LINE_LENGTH.
      IF SY-SUBRC <> 0.
        EXIT.
      ENDIF.
      APPEND LINES OF MESSAGE_LINE TO MESSAGE_TEXT.
    ENDDO.
  CALL FUNCTION 'Z_SEND_MESSAGE'
    EXPORTING
      MESSAGE_RECEIVER = BAPIP0002-PERNR
      MESSAGE_SUBJECT  = SOOD1-OBJDES
    TABLES
```

---

**Adding Email Capability to R/3**

```
                MESSAGE_TEXT      = MESSAGE_TEXT
      EXCEPTIONS
                MESSAGE_NOT_SEND = 1
                OTHERS           = 2.
    IF SY-SUBRC = 0.
      MESSAGE W004.
    ENDIF.
    CLEAR SOOD1-OBJDES.
  ENDCASE.
```

## Results

The modified program sends the message text via the ITS mass data channel, but all other information is transferred via screen fields. For this reason, you cannot run the transaction in R/3 alone - you can only test it from the Web browser.

## Adding Email Capability to the Web Transaction

When you have modified the R/3 transaction by adding email capability, you can generate and test the Web transaction in the SAP@Web Studio. You also need to modify the HTML code in the template for screen 300.

### Procedure

10. Create the ITS service for the transaction, as described in [Creating the ITS Service \[Seite 19\]](#).
11. Create the HTML templates for the transaction, as described in [Creating HTML Templates \[Seite 21\]](#).
3. In the HTML template for screen 300, add a text box with multiple lines to allow the user to enter an email message

```
<textarea name="message_text[]" cols="46" rows="10"></textarea>
```

4. Create any language resources required for the transaction, as described in [Creating Language Resources \[Seite 23\]](#).

Although scope for language resources is limited in this case, the sample language resource files provided with the sample transaction demonstrate their purpose.

5. Make this ITS service the currently active ITS service, and publish it to your ITS site, as described in [Publishing the ITS Service \[Seite 25\]](#).
5. Enter **a\*** in the *Last Name* field.
6. Scroll down and choose *Search*.  
The Web browser displays a list of employees matching the selection criteria.
7. Enter a personnel number in the *Personnel no.* field and choose *Detail*.  
The Web browser displays detailed information on that employee, and a box to enter an email message.
9. Enter a mail message and send it.

### Result

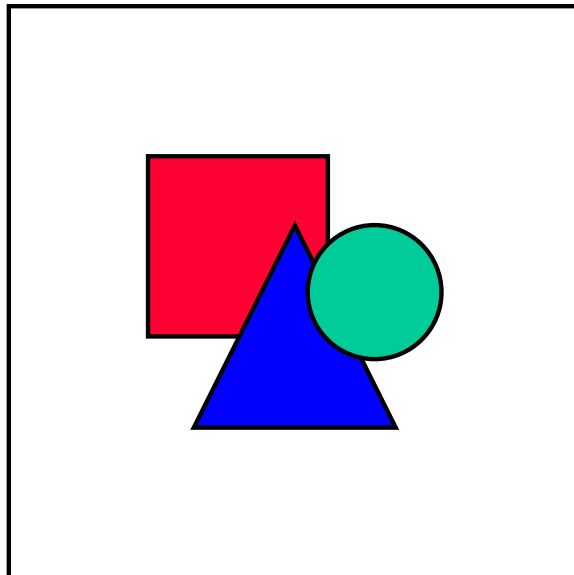
On next logon to R/3, the user should find mail waiting in their inbox.

**Lesson 5: Multi-Frame Transactions****Lesson 5: Multi-Frame Transactions**

In this lesson, you learn how to create a multi-frame Web transaction, using an ABAP transaction that contains subscreens.

Using frames in an HTML page has advantages and disadvantages:

- On the positive side, frames improve flexibility and clarity. When the user selects a link or a pushbutton, the Web browser only needs to regenerate the target frame, not the whole page. In ABAP, subscreens provide a similar capability.
- On the negative side, HTML pages that contain frames are often large and cannot be displayed satisfactorily on smaller monitors. Therefore, remember that using frames brings no advantage if the Web browser has to display more frames than can be displayed comfortably on the screen.



The R/3 transaction you create here is intended only for use as a Web transaction and should be used with the SAP GUI only for test purposes. A comparable transaction for the SAP GUI would have to be designed from a completely different point of view.

For an illustration of the complete implementation, see sample program WWIACWY5 (transaction WWY5).

## Adding the Subscreen Logic in R/3

To add subscreen logic, you need to create a base screen and change the original screens to subscreens

### Procedure

1. Copy the program you created in Lesson 2 (or the sample program WWIACWY2) to a new program, as described in [Copying the Basic R/3 Transaction \[Seite 10\]](#).

Make sure that your program name conforms to the usual naming conventions.

2. Create a base screen (see below)
3. Change the original screens to subscreens (see below)

### Creating a Base Screen

1. Create a new screen 1000, type *Normal*, and enter the following in the flow logic editor:

```
PROCESS BEFORE OUTPUT.  
  MODULE STATUS_1000.  
    CALL SUBSCREEN SS_SELECTION INCLUDING '<program name>' '0100'.  
    CALL SUBSCREEN SS_EMPLS INCLUDING   '<program name>' '0200'.  
    CALL SUBSCREEN SS_DETAIL INCLUDING   '<program name>' '0300'.  
  
PROCESS AFTER INPUT.  
  CALL SUBSCREEN SS_SELECTION.  
  CALL SUBSCREEN SS_EMPLS.  
  CALL SUBSCREEN SS_DETAIL.  
  MODULE USER_COMMAND_1000.
```

In the screen layout, add the following elements (in any configuration you like):

- a. Three subscreens, named SS\_SELECTION, SS\_EMPLS, and SS\_DETAIL.
  - b. Three boxes around the subscreens (optional)
2. Modify the ABAP code:
    - a. For module STATUS\_1000, code: set PF-STATUS to 'BACK'.
    - b. Leave USER\_COMMAND\_1000 blank.

You define this module using code from screen 100 (see below).

### Changing the Original Screens to Subscreens

1. In the Screen Painter, change the original screens (100, 200, and 300) to subscreens:
  - a. In the field list, delete the name of the OK code field.
  - b. In the screen attributes, change the screen type from *Normal* to *Subscreen*.
  - c. In the screen flow logic, delete the module calls to STATUS\_0100, STATUS\_0200, or STATUS\_0300.
2. Modify screen 100

With subscreens, all function codes must be handled in the containing screen 1000. This means making the following changes:

### Adding the Subscreen Logic in R/3

In the ABAP code, cut the `USER_COMMAND_0100` subroutine and paste the code into the `USER_COMMAND_1000` subroutine. Modify this code in two ways:

- a. Delete the `set screen 200` statement and the containing `if` condition
- b. In the CASE statement, add a `when 'BACK'` condition.

```
FORM USER_COMMAND_1000.
  DATA: ILINES LIKE SY-TABIX.
  CASE FCODE.
    WHEN 'BACK'.
      SET SCREEN 0.
    WHEN 'SELC'.
      ...
```

#### 3. Modify screen 200

In the screen, modify the subscreen display:

- a. To make the step loop smaller, reduce its size to between six and eight lines
- b. To make the step loop table narrower, delete the *Phone* field
- c. Place the *Detail* button above the step loop
- d. In the ABAP code, delete the `set screen 300` statement. The code for `USER_COMMAND_0200` then reads:

```
FORM USER_COMMAND_0200.
  CASE FCODE.
    WHEN 'DETA'.
      PERFORM READ_EMPLOYEE_WITH_KEY_INTO_WA USING ZZIACWXX-
SEL_PERNR.
      CLEAR ZZIACWXX-SEL_PERNR.
      IF SY-SUBRC = 0.
        ELSE.
          MESSAGE W002.
        ENDIF.
      WHEN 'FT23'.
        EMPLOYEE_IDX = EMPLOYEE_IDX + STEPLOOP_SIZE.
      WHEN 'FT22'.
        EMPLOYEE_IDX = EMPLOYEE_IDX - STEPLOOP_SIZE.
      WHEN 'FT21'.
        EMPLOYEE_IDX = 1.
      ENDCASE.
  ENDFORM. " USER_COMMAND_0200
```

#### 4. Modify screen 300

- a. In the screen flow logic, delete the call to `USER_COMMAND_0300`, and the related ABAP code.
- b. In the screen layout, add (or delete) any fields you like.

## Implementing the Multi-Frame Web Transaction

When you have modified the R/3 transaction by adding subscreen logic, you can generate and test the Web transaction in the SAP@Web Studio.

The HTML generator produces `form` statements for each set of function codes that updates a separate frame. However, the HTML generator cannot know which frame is updated by which form, so you must add this information yourself.

For this reason, you need to modify the HTML code in the template for screen 200. This is because the HTML generator does not know about the transaction flow, so you must modify the templates to perform the navigation you want by specifying the target screen.

For each function code or group of function codes sent, you need to define the `wgateURL` statement accordingly.

### Procedure

12. Create the ITS service for the transaction, as described in [Creating the ITS Service \[Seite 19\]](#).
13. Create the HTML templates for the transaction, as described in [Creating HTML Templates \[Seite 21\]](#).
14. In the HTML template for screen 100, modify the `wgateURL` statement.

If the user chooses *Search* after entering selection criteria, you need to direct the Web transaction to the correct target frame by setting the value of the `~target` parameter:

```
<FORM ACTION="`wgateURL(~target="SS_EMPLS")`" METHOD="post">
```

4. In the HTML template for screen 200, modify the `wgateURL` statement.

If the user enters an employee number and chooses *Detail*, you need to direct the Web transaction to the correct target frame by setting the value of the `~target` parameter:

```
<FORM ACTION="`wgateURL(~target="SS_DETAIL")`" METHOD="post">
  `ZZIACWXX-SEL_PERNR.label`
<INPUT TYPE="text" name="ZZIACWXX-SEL_PERNR" VALUE="`ZZIACWXX-
SEL_PERNR`" maxlength=" 8" size=" 8" >
<INPUT TYPE="submit" name="~OkCode(DETA)" value="`DETAIL.label`">
</FORM>
```

5. Create any language resources required for the transaction, as described in [Creating Language Resources \[Seite 23\]](#).

Although scope for language resources is limited in this case, the sample language resource files provided with the sample transaction demonstrate their purpose.

6. Make this ITS service the currently active ITS service, and publish it to your ITS site, as described in [Publishing the ITS Service \[Seite 25\]](#).

## Lesson 6: Transaction Synchronization

# Lesson 6: Transaction Synchronization

In this lesson, you learn how to synchronize single-frame and multi-frame transactions.

The previous lessons in this tutorial assumed that users running Web transactions from a Web browser navigate using only hyperlinks and pushbuttons defined in the HTML template. This guarantees consistency between HTML templates in the Web browser and screens in the R/3 transaction, but does not take into account what might happen if the user navigates using the buttons available in the Web browser.

If a user starts a Web transaction from a Web browser and navigates with one of the Web browser buttons, R/3 screens can get out of step with their corresponding HTML templates, since no screen change is triggered in R/3. This means that the Internet Transaction Server (ITS) cannot pass data to the R/3 System, because the data does not match the current status of the system. As a result, data input can get lost.



Suppose the user starts a Web transaction, passes through several screens, but then returns to a previously viewed page using the Web browser's *Back* button.

In this case, the Web browser retrieves the required HTML page from a memory cache rather than accessing R/3 using an HTTP request. This is because the Internet was originally intended to distribute static information, and memory cache functions are ideal for speeding up access time and reducing the pressure on networks.

Synchronization problems can occur in any Web transaction, because a stateless medium (the Web browser) is trying to interact with a status-dependent medium (the R/3 System).

By synchronizing the transaction, you ensure that the HTML template displayed in the Web browser always corresponds to the correct R/3 screen.

## How ITS Synchronization Works

When you start a Web transaction, the ITS generates HTML templates that correspond to R/3 screens. These HTML templates contain forms and hypertext links that send data back to the ITS, which in turn passes it on to the R/3 System. In R/3, the corresponding R/3 screen processes the data.

Since the ITS keeps track of the R/3 screen needed to process the data, it can detect when the R/3 screen and the HTML template do not match. When this happens, synchronization is required.

You can resolve the synchronization problem in various ways. The method you choose depends on the transaction and the ease of use you wish to offer the user.

Synchronization occurs in two steps:

1. The ITS notifies R/3 that the R/3 screen does not match the HTML template.

To do this, the ITS sends a special function code, and the transaction executes a subroutine called SYNCHRONIZE. This subroutine calls the function module ITS\_GET\_SYNC\_INFO to get information (screen number and module pool) about the screen to which synchronization is needed. It is the responsibility of the R/3 transaction to change to the main screen (and subscreens) requested by the ITS.

---

**Lesson 6: Transaction Synchronization**

2. When the transaction has been successfully synchronized, the ITS posts all data received from the Web browser to R/3.

Changing to the screen that can process the Web browser data is not always sufficient, because the information posted from the Web must be sufficient to reconstruct the complete state of the transaction at the time the original page was created. For some transactions this may not be possible due to the large amount of information required.

In these cases, it is advisable to disable synchronization by setting the `~SyncBehaviour` parameter in the ITS service file to `NoResync`. This setting causes a warning message to be displayed in the browser if the user navigates to a previous HTML page with the *Back* button. Generally, it is good application design to offer navigation buttons in your Web application.

### Example Transactions

The next three example transactions in the tutorial progressively demonstrate the synchronization problem in three scenarios showing single frames, multiple frames, and multiple frames with update.

- For the complete solution to synchronizing single-frame transactions, see sample program WWIACWY6 (transaction WWY6).
- For the complete solution to synchronizing multi-frame transactions, see sample program WWIACWY7 (transaction WWY7).
- For the complete solution to synchronizing multi-frame transactions with update, see sample program WWIACWY8 (transaction WWY8).

## Synchronizing Single-Frame Transactions

## Synchronizing Single-Frame Transactions

With single-frame transactions, the Internet Transaction Server (ITS) uses the function code AWSY to notify the R/3 System that a screen has to be synchronized. When the transaction receives this function code, it must change to the required screen.

### Procedure

To demonstrate how to synchronize a single-frame transaction, we can expand the multi-frame scenario used in Lesson 2:

1. Copy the program you created in Lesson 2 (or the sample program WWIACWY2) to a new program, as described in [Copying the Basic R/3 Transaction \[Seite 10\]](#).

Make sure that your program name conforms to the usual naming conventions.

2. In the PAI modules of screen programs 100, 200 and 300, insert a call to the module SYNCHRONIZE, as shown here for screen 100:

```
PROCESS BEFORE OUTPUT.
  MODULE STATUS_0100.

PROCESS AFTER INPUT.
  MODULE SYNCHRONIZE.
  MODULE USER_COMMAND_0100.
```

3. In the module synchronize, call the ABAP subroutine SYNCHRONIZE, so you only have to implement the synchronization code once.
4. Define the ABAP code for the subroutine synchronize as follows:

```
FORM SYNCHRONIZE.
  DATA: SYNC_INFO LIKE SITSSYNC OCCURS 0 WITH HEADER LINE.
  IF FCODE(4) = 'AWSY'.
    CALL FUNCTION 'ITS_GET_SYNC_INFO'
      TABLES
        SYNC_INFO          = SYNC_INFO
      EXCEPTIONS
        ITS_NOT_AVAILABLE = 1
        OTHERS             = 2.
  * break-point.
  IF SY-SUBRC = 0.
    READ TABLE SYNC_INFO WITH KEY SUBSC = 'MAIN'.
    IF SY-SUBRC = 0.
      SET SCREEN SYNC_INFO-DYNNR.
    ENDIF.
  ENDIF.
ENDIF.
ENDFORM.                " SYNCHRONIZE
```

If the transaction receives the function code AWSY, it calls the function module ITS\_GET\_SYNC\_INFO, which returns the necessary synchronization information (module pool and screen number) in an internal table called SYNC\_INFO.

### Synchronizing Single-Frame Transactions

The internal table is defined as a SITSSYNC structure and contains only one line for single-frame transactions, each with three fields:

Field	Description
SUBSC	Subscreen name. For the main screen, this field contains the character string MAIN. After returning from ITS_GET_SYNC_INFO, the code should search for the line in the table where SUBSC equals MAIN.
REPID	Module pool name. Since this example has only one module pool, this field is ignored.
DYNNR	Screen number. This is the screen to which the transaction should synchronize, so the program should execute a SET SCREEN statement to the screen specified.



Comparing FCODE(4) with AWSY ensures that the program checks only the first 4 characters of the OK code. For historical reasons, the function code sent by the ITS is longer than four characters, so you need to ensure that you compare only the first four letters with your application's function code variable.

## Handling Single-Frame Synchronization

# Handling Single-Frame Synchronization

When you have modified the R/3 transaction by inserting the synchronization code, you can generate and test the transaction in the SAP@Web Studio, and track the synchronization process in the ITS debugger.

## Procedure

### Generating the Web Transaction

15. Create the ITS service for the transaction, as described in [Creating the ITS Service \[Seite 19\]](#).
16. Create the HTML templates for the transaction, as described in [Creating HTML Templates \[Seite 21\]](#).
17. Create any language resources required for the transaction, as described in [Creating Language Resources \[Seite 23\]](#).

Although scope for language resources is limited in this case, the sample language resource files provided with the sample transaction demonstrate their purpose.

18. Make this ITS service the currently active ITS service, and publish it to your ITS site, as described in [Publishing the ITS Service \[Seite 25\]](#).

### Testing the Web Transaction

To understand how you can synchronize R/3 screens that get out of step with HTML templates, you need to run the Web transaction in your Web browser, and then follow the synchronization process in the ITS debugger:

1. Start the Web transaction in your Web browser, and start the ITS debugger.  
You see the selection screen, both in the Web browser and in screen 100 on the R/3 side.
2. In the Web browser, enter **a\*** in the *Last name* field, and choose *Search*.  
You see a list of employees matching your criteria, both in the Web browser and in screen 200 on the R/3 side.
3. In the Web browser, enter an employee number in the *Personnel number* field, and choose *Detail*.  
You see detail data for the employee, both in the Web browser and in screen 300 on the R/3 side.
4. In the Web browser, choose *Back* to return to the list of employees in screen 200.  
The Web browser displays the list of employees (screen 200), but R/3 still displays the detail data (screen 300) as the current screen. This is because the HTML page is restored from the Web browser's HTML page cache, but the R/3 System knows nothing of this.
5. In R/3, activate debugging by entering **/h** in the command field.  
If you single-step through the debugger at this point, you can observe what happens.
6. In the Web browser, enter a different employee number, and choose *Detail*.

---

**Handling Single-Frame Synchronization**

The transaction goes into debugging mode in R/3, and you see an information message to this effect in the Web browser.

At this point, you are set to single-step through the ABAP code for synchronizing screens. This code calls the function module ITS\_GET\_SYNC\_INFO to find out the identity of the next screen, and then calls SET SCREEN to jump there.

To see how this function works, you must set an explicit breakpoint after the call to ITS\_GET\_SYNC\_INFO:

7. In the debugger, choose *Execute* to step past the function module call, and then use the *Table* function to display the contents of SYNC\_INFO.

The REPID and DYNNR fields contain the module pool name and screen number.

8. Single-step past the SET SCREEN SYNC\_INFO-DYNNR statement

The program redisplay the correct screen (screen 200).

As soon as screen 200 is displayed again in R/3, the ITS posts all data (that is, the new request for detail data on the second employee number) to the transaction.

Consequently, the transaction changes to screen 300 and display the details for the second employee number.

## Synchronizing Multi-Frame Transactions

## Synchronizing Multi-Frame Transactions

As shown in [Synchronizing Single-Frame Transactions \[Seite 46\]](#), the Internet Transaction Server (ITS) uses the function code AWSY to notify the R/3 System that a screen has to be synchronized.

With multi-frame transactions, the Internet Transaction Server (ITS) uses two function codes to notify the R/3 System that a main screen, or one or more subscreens, have to be synchronized. When the transaction receives one of these function codes, it must change to the requested screen or subscreen.

- If the main screen has to be synchronized, the ITS sends the function code AWSY (as for single-frame transactions).
- If one or more subscreens have to be synchronized, the ITS sends the function code AWSS.

To demonstrate how to synchronize a multiframe transaction, we can expand the multi-frame scenario used in Lesson 5:

### Procedure

1. Copy the program you created in Lesson 5 (or the sample program WWIACWY5) to a new program, as described in [Copying the Basic R/3 Transaction \[Seite 10\]](#).

Make sure that your program name conforms to the usual naming conventions.

5. Define the ABAP code for the subroutine SYNCHRONIZE as follows:

```
FORM SYNCHRONIZE.
  DATA: SYNC_INFO LIKE SITSSYNC OCCURS 0 WITH HEADER LINE.
  IF FCODE(4) = 'AWSY' OR FCODE(4) = 'AWSS'.
    CALL FUNCTION 'ITS_GET_SYNC_INFO'
      TABLES
        SYNC_INFO          = SYNC_INFO
      EXCEPTIONS
        ITS_NOT_AVAILABLE = 1
        OTHERS             = 2.
    IF SY-SUBRC = 0.
      * break-point.
      LOOP AT SYNC_INFO.
        CASE SYNC_INFO-SUBSC.
          WHEN 'MAIN'.
            SET SCREEN SYNC_INFO-DYNNR.
          WHEN 'SS_DETAIL'.
            DETAIL_SCREEN = SYNC_INFO-DYNNR.
          ENDCASE.
        ENDLIST.
      ENDIF.
    ENDIF.
  ENDFORM.
" SYNCHRONIZE
```

If the transaction receives the function code AWSY or AWSS, it calls the function module ITS\_GET\_SYNC\_INFO, which returns the necessary synchronization information (module pool and screen number) in an internal table called SYNC\_INFO.

Synchronizing Multi-Frame Transactions

The internal table is defined as a SITSSYNC structure and contains only one line for single-frame transactions, each with three fields:

Field	Description
SUBSC	Subscreen name. For the main screen, this field contains the character string MAIN. After returning from ITS_GET_SYNC_INFO, the code should search for the line in the table where SUBSC equals MAIN.
REPID	Module pool name. Since this example has only one module pool, this field is ignored.
DYNNR	Screen number. This is the screen to which the transaction should synchronize, so the program should execute a SET SCREEN statement to the screen specified.



Comparing FCODE(4) with AWSY or AWSS ensures that the program checks only the first 4 characters of the OK code. For historical reasons, the function code sent by the ITS is longer than four characters, so you need to ensure that you compare only the first four letters with your application's function code variable.

- Add another subscreen (screen 400) that displays different data.  
In transaction ZWY5, screen 300 displays only organizational data for an employee.  
In this transaction, screen 400 is also displayed in the subscreen SS\_DETAIL, but contains personal data for an employee.
- In the Screen Painter, add two buttons to screen 200 (with function codes ORGD and PERD) to enable the user to request the two types of data display.
- Modify the flow logic of screen 1000 to change the contents of subscreen SS\_DETAIL accordingly at runtime:

```

PROCESS BEFORE OUTPUT.
  MODULE STATUS_1000.
  CALL SUBSCREEN SS_SELECTION INCLUDING '<program name>' '0100'.
  CALL SUBSCREEN SS_EMPLS INCLUDING   '<program name>' '0200'.
  CALL SUBSCREEN SS_DETAIL INCLUDING  '<program name>'
DETAIL_SCREEN.

PROCESS AFTER INPUT.
  MODULE SYNCHRONIZE.
  CALL SUBSCREEN SS_SELECTION.
  CALL SUBSCREEN SS_EMPLS.
  CALL SUBSCREEN SS_DETAIL.
  MODULE USER_COMMAND_1000.
  
```

- Change the code in USER\_COMMAND\_0200 to handle the function codes PERD and ORGD.

At runtime, the variable DETAIL\_SCREEN contains either screen 300 or screen 400, depending on the function code received.

```

FORM USER_COMMAND_0200.
  
```

## Synchronizing Multi-Frame Transactions

```
      CASE FCODE.
        WHEN 'ORGD'.
          PERFORM READ_EMPLOYEE_WITH_KEY_INTO_WA USING ZZIACWXX-
SEL_PERNR.
          IF SY-SUBRC = 0.
            DETAIL_SCREEN = '0300'.
          ELSE.
            MESSAGE W002.
          ENDIF.
        WHEN 'PERD'.
          PERFORM READ_EMPLOYEE_WITH_KEY_INTO_WA USING ZZIACWXX-
SEL_PERNR.
          IF SY-SUBRC = 0.
            DETAIL_SCREEN = '0400'.
          ELSE.
            MESSAGE W002.
          ENDIF.
        WHEN 'FT23'.
          EMPLOYEE_IDX = EMPLOYEE_IDX + STEPLOOP_SIZE.
      ...
```

## Results

The final layout for screen 200 contains two pushbuttons.

## Handling Multi-Frame Synchronization

When you have modified the R/3 transaction by inserting the synchronization code, creating subscreen 400 to display personal data, adding two pushbuttons to subscreen 200 to request organizational or personal data, modifying the screen flow logic of screen 1000 to display the required data at runtime, and adding code to handle the two pushbuttons, you can generate and test the transaction in the SAP@Web Studio, and track the synchronization process in the ITS debugger.

### Procedure

#### Generating the Web Transaction

19. Create the ITS service for the transaction, as described in [Creating the ITS Service \[Seite 19\]](#).
20. Create the HTML templates for the transaction, as described in [Creating HTML Templates \[Seite 21\]](#).
21. Create any language resources required for the transaction, as described in [Creating Language Resources \[Seite 23\]](#).

Although scope for language resources is limited in this case, the sample language resource files provided with the sample transaction demonstrate their purpose.
22. Make this ITS service the currently active ITS service, and publish it to your ITS site, as described in [Publishing the ITS Service \[Seite 25\]](#).

#### Testing the Web Transaction

To understand how you can synchronize R/3 screens that get out of step with HTML templates, you need to run the Web transaction in your Web browser, and then follow the synchronization process in the ITS debugger:

9. Start the Web transaction in your Web browser, and start the ITS debugger.

You see the selection screen, both in the Web browser and in subscreen 100 on the R/3 side.
10. In the Web browser, enter **a\*** in the *Last name* field, and choose *Search*.

You see a list of employees matching your criteria, both in the Web browser and in subscreen 200 on the R/3 side.
11. In the Web browser, enter an employee number in the *Personnel number* field, and choose *Organizational*.

You see organizational data for the employee, both in the Web browser and in subscreen 300 on the R/3 side.
12. In the Web browser, choose *Personal*.

You see personal data for the same employee, both in the Web browser and in subscreen 400 on the R/3 side.
13. In the Web browser, enter a different employee number in the *Personnel number* field, and choose *Personal* again.

---

**Handling Multi-Frame Synchronization**

You see personal data for the second employee, both in the Web browser and in subscreen 400 on the R/3 side.

14. In the Web browser, choose *Back* to return to the personal data for the first employee.

The Web browser displays the personal data for the first employee again, but R/3 still displays the personal data for the second employee as the current screen. This is because the ITS restores the HTML page from the Web browser's HTML page cache without notifying the R/3 System.

15. In R/3, activate debugging by entering `/h` in the command field.

If you single-step through the debugger at this point, you can observe what happens.

In this example, the function codes PERD and ORGD can still be processed and employee numbers can be entered at any time. As shown in Lesson 7, more serious synchronization problems occur if we allow the user to make updates.

## Synchronizing Multi-Frame Transactions with Update

As shown in the previous exercise, the Internet Transaction Server (ITS) uses two function codes to notify the R/3 System that a main screen, or one or more subscreens, have to be synchronized. However, this still does not fully demonstrate the importance of synchronization in Web transactions. To do this, we need to allow the user to make updates.

If you run the Web transaction created in [Synchronizing Multi-Frame Transactions \[Seite 50\]](#), or the sample Web transaction WWY7, from a Web browser, display personal data for several employees by using the Web browser's *Back* button, and then decide to update the personal data of a user, the correct screen **must** be present on the R/3 side to be able to process the data posted by the ITS. If the transaction does not synchronize, this data will get lost.

To demonstrate how to synchronize a multiframe transaction, which also allows updates, we can expand the multi-frame scenario used in the previous exercise by adding a *Change* button to screen 400 that allows users to update the employee personal data.

### Procedure

2. Copy the program you created in Lesson 7 (or the sample program WWIACWY7) to a new program, as described in [Copying the Basic R/3 Transaction \[Seite 10\]](#).

Make sure that your program name conforms to the usual naming conventions.

3. In the Screen Painter, modify screen 400
  - a. Add a pushbutton (with function code CHNG) to enable the user to update the personal data (in this case, the two telephone numbers) of an employee.
  - b. Enable the two telephone number fields SS300-TEL01 and SS300-TEL02 to accept user input.
4. Add the following ABAP code to USER\_COMMAND\_1000:

```
FORM USER_COMMAND_1000
  DATA: OPERATION LIKE PSPAR-ACTIO,
         RETURN LIKE BAPIRETURN1,
         VALIDITYBEGIN LIKE P0001-BEGDA,
         ENAME (255) TYPE C,
         ILINES LIKE SY-TABIX.

  DATA: VALUES LIKE PPROP OCCURS 10 WITH HEADER LINE,
         IP0032 LIKE P0032 OCCURS 1 WITH HEADER LINE.

  ...

  WHEN 'CHNG'.
    DETAIL_SCREEN = '0400'.

  * Check if there are currently any values
  * to determine whether to insert or modify:

  CALL FUNCTION 'HR_READ_INFOTYPE'
    EXPORTING
      PERNR = SS300_0001T-PERNR
      INFY = '0032'
      BEGDA = SY-DATUM
      ENDDA = SY-DATUM
```

## Synchronizing Multi-Frame Transactions with Update

```

      TABLES
        INFTY_TAB = IP0032.

DESCRIBE TABLE IP0032 LINES ILINES.
IF ILINES EQ 0.
  OPERATION = 'INS'.
  VALIDITYBEGIN = SY-DATUM.
ELSE.
  READ TABLE IP0032 INDEX 1.
  VALIDITYBEGIN = IP0032-BEGDA.
  OPERATION = 'MOD'.
ENDIF.

* Add new values to value table:

VALUES-INFTY = '0032'.
VALUES-FNAME = 'P0032-TEL01'.
VALUES-FVAL = SS300_0032-TEL01.
APPEND VALUES.
VALUES-INFTY = '0032'.
VALUES-FNAME = 'P0032-TEL02'.
VALUES-FVAL = SS300_0032-TEL02.
APPEND VALUES.

* Call update function module:

CALL FUNCTION 'HR_MAINTAIN_MASTERDATA'
  EXPORTING
    PERNR          = SS300_0001T-PERNR
    ACTIO          = OPERATION
    BEGDA          = VALIDITYBEGIN
    ENDDA          = '99991231'
    SUBTY          = SPACE
    NO_ENQUEUE    = SPACE
  IMPORTING
    RETURN1        = RETURN
  TABLES
    PROPOSED_VALUES = VALUES
    MODIFIED_KEYS   =
  EXCEPTIONS
    OTHERS          = 1.

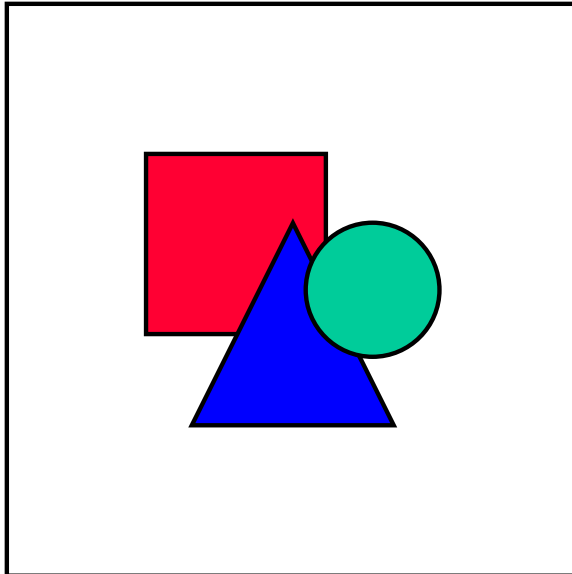
IF RETURN IS INITIAL.
  CONCATENATE SS300_0002-VORNA SS300_0002-NACHN
    INTO ENAME SEPARATED BY SPACE.
  CONDENSE ENAME.
  MESSAGE S006 WITH ENAME SPACE.
ELSE.
  MESSAGE ID      RETURN-ID
    TYPE 'S'
    NUMBER RETURN-NUMBER
  WITH RETURN-MESSAGE_V1 RETURN-MESSAGE_V2
    RETURN-MESSAGE_V3 RETURN-MESSAGE_V4.
ENDIF.

```

...

## Synchronizing Multi-Frame Transactions with Update

After making the necessary data declarations, the logic responds to the function code CHNG by calling the function module HR\_READ\_INFOTYPE, which reads master data for the HR info type 0032, to find out whether values for already exist in the structure P0032. This determines whether an insert or modify operation is needed. The new values are appended to an internal table VALUES, which is passed to the the update function module HR\_MAINTAIN\_MASTERDATA.



HR\_READ\_INFOTYP and HR\_MAINTAIN\_MASTERDATA are standard function modules from the Human Resources (HR) application.

## Results

The final layout for screen 400 allows users to insert or change values in the two telephone number fields, and contains a pushbutton *Change* to trigger the update.

## Handling Multi-Frame Synchronization with Update

# Handling Multi-Frame Synchronization with Update

When you have modified the R/3 transaction by adding a pushbutton to subscreen 400 to enable the user to change telephone numbers, and adding the necessary code to perform the update, you can generate and test the transaction in the SAP@Web Studio, and track the synchronization process in the ITS debugger.

## Procedure

### Generating the Web Transaction

23. Create the ITS service for the transaction, as described in [Creating the ITS Service \[Seite 19\]](#).
24. Create the HTML templates for the transaction, as described in [Creating HTML Templates \[Seite 21\]](#).
25. Create any language resources required for the transaction, as described in [Creating Language Resources \[Seite 23\]](#).

Although scope for language resources is limited in this case, the sample language resource files provided with the sample transaction demonstrate their purpose.

26. Make this ITS service the currently active ITS service, and publish it to your ITS site, as described in [Publishing the ITS Service \[Seite 25\]](#).

### Testing the Web Transaction

To understand how synchronization works in a multi-frame Web transaction that allows updates, you need to run the transaction in your Web browser, and follow the synchronization process in the ITS debugger:

16. Start the Web transaction in your Web browser, and start the ITS debugger.

You see the selection screen, both in the Web browser and in subscreen 100 on the R/3 side.
17. In the Web browser, enter **a\*** in the *Last name* field, and choose *Search*.

You see a list of employees matching your criteria, both in the Web browser and in subscreen 200 on the R/3 side.
18. In the Web browser, enter an employee number in the *Personnel number* field, and choose *Personal*.

You see personal data for the employee, both in the Web browser and in subscreen 400 on the R/3 side.
19. In the Web browser, enter a different employee number in the *Personnel number* field, and choose *Personal* again.

You see personal data for the second employee, both in the Web browser and in subscreen 400 on the R/3 side.
20. In the Web browser, choose *Back* to return to the personal data for the first employee.

The Web browser displays the personal data for the first employee again, but R/3 still displays the personal data for the second employee as the current screen. This is

## Handling Multi-Frame Synchronization with Update

because the HTML page is restored from the Web browser's HTML page cache, but the R/3 System knows nothing of this.

21. In the Web browser, modify the personal data for the first employee by changing one or both of the telephone numbers, and choose *Change*.

Looking at the Web browser, it appears that you you have changed the personal data of the first employee, but the ITS debugger shows otherwise. On the R/3 side, screen 400 still displays the personal data of the second employee, so the transaction thinks you are still processing the second employee, not the first.

22. In R/3, activate debugging by entering `/h` in the command field.

If you single-step through the debugger at this point, you can observe what happens.

In this case, the ITS detects that the transaction needs to display screen 400, and the synchronization code takes care of this. However, the code does not ensure that the correct personal data is displayed in screen 400, because it does not know which employee is to be updated. As a result, the „wrong“ employee gets updated.

### Modifying the HTML Code

To resolve the synchronization problem, the ITS must post enough data to reconstruct the state of the transaction at the time the HTML page was generated. To achieve this, you can add hidden fields to the HTML form.

```
<INPUT TYPE="hidden" name="SS300_0001T-PERNR" VALUE="`SS300_0001T-  
PERNR` ">
```

### Result

The new values are transported back to the transaction on the R/3 side.

## Checking ITS Services Into ITS Source Control

# Checking ITS Services Into ITS Source Control

When you have created and tested your Web transactions in the SAP@Web Studio, you can check the new Internet Transaction Server (ITS) services into ITS source control in the R/3 System. This serves to protect the files against simultaneous modification.

What you can or cannot do with ITS files depends on whether they have been checked into ITS source control at all, and where they currently reside – in the SAP@Web Studio or the R/3 System. All users have read access to files, but only a single user can check them out and modify them.

## Procedure

To check the files of your ITS services into ITS source control, take the following steps:

- In the SAP@Web Studio:
  - a. Connect to R/3
  - b. Check the ITS files into ITS source control
- In the R/3 System:
  - a. Create a change request in the Workbench Organizer
  - b. Assign the ITS files to the change request

At subsequent check-ins, the R/3 System remembers the relevant change request and automatically replaces the files. You don't need to specify a change request each time.

If you release a change request for transport, you must create a new change request and reassign the files to it. If you fail to do this, the files cannot be checked out (or in) by other users.

## Connecting to R/3 and Checking ITS Files Into ITS Source Control

1. Start the SAP@Web Studio as described in [Starting the SAP@Web Studio \[Seite 13\]](#).
2. Select the *Source Control* view.
3. Choose *Tools* → *Source Control* → *Connect to R/3*.

The SAP@Web Studio displays the *Select R/3 System* dialog box, which displays all the R/3 connections you have defined.

4. Select an R/3 System, and choose *OK*.
5. If necessary, enter values for client, user, password, and language.
6. If you want the SAP@Web Studio to remember your password, select *Save Password*.

If you select this checkbox, the SAP@Web Studio stores your password internally in encrypted format. This means that when you log on again, you do not have to enter your password. If you set up the SAP@Web Studio to maintain all other logon information for you, do not see a logon screen at all.

7. Choose *OK*.

## Checking ITS Services Into ITS Source Control

The SAP@Web Studio logs you on to the R/3 System, and displays all source controlled ITS services in the *Source Control* view. To display the objects in each service, double-click them.

8. Choose *Tools* → *Source Control* → *Add file(s)*.
9. If the files you want to add are not already displayed, navigate to the relevant directory.
10. Select the files to be added.
11. Choose *Add*.
12. Ensure that the files you want are checked, and choose *OK*.

## Creating a Change Request and Assigning ITS Files

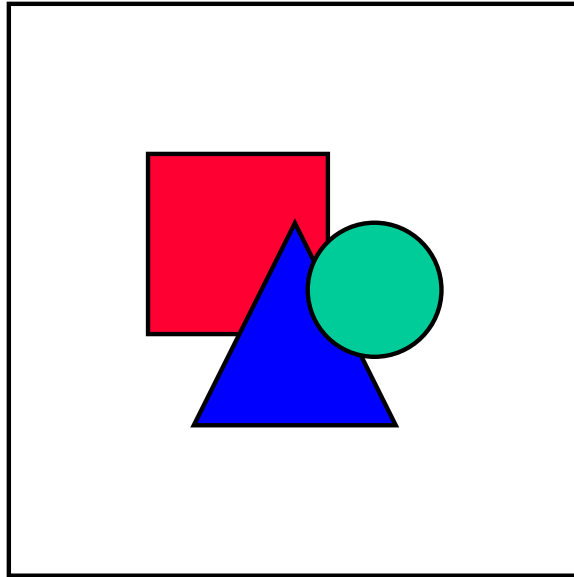
1. In the R/3 System, create a change request in the Workbench Organizer
  - To do this, choose *Tools* → *ABAP Workbench* → *Overview* → *Workbench Organizer*
2. Assign the ITS files to the change request.
  - a. Choose *Tools* → *Business Framework* → *Web development* → *Web object administration*.
  - b. In the *Service name* field, enter the service name.
  - c. Select the service and choose *Transport*.
    - You see the *Change Request Query* dialog.
  - d. Enter a change request number.
    - If desired, you can choose *Own requests* or *Create request* to branch to the Workbench Organizer.

## Result

The files are checked in and assigned for transport.

When you have checked the ITS files into ITS source control and assigned them to a change request in the R/3 System, you check them out for modification, or just get read access.

Checking ITS Services Into ITS Source Control



For further information about change requests and managing development objects in the Workbench Organizer, see [BC - Change and Transport Organizer \[Extern\]](#).