The Workflow Management Coalition Specification

Workflow Management Coalition
Workflow Client Application (Interface 2)
Application Programming Interface (WAPI)
Naming Conventions

Document Number WFMC-TC-1013

01-November-97
Version 1.4

Copyright (C) 1993, 1996 The Workflow Management Coalition

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission of the Workflow Management Coalition except that reproduction, storage or transmission without permission is permitted if all copies of the publication (or portions thereof) produced thereby contain a notice that the Workflow Management Coalition and its members are the owners of the copyright therein.

This Specification has been authored by Workflow Management Coalition members.
Workflow Application Programmer's Interface (WAPI) Naming Conventions

The "WfMC" logo and "Workflow Management Coalition" name are service marks of the Workflow Management Coalition.

Neither the Workflow Management Coalition nor any of its members make any warranty of any kind whatsoever, express or implied, with respect to the Specification, including as to non-infringement, merchantability or fitness for a particular purpose. This Specification is provided “as is”.

First printing, November 1995
Second printing, version 1.1, May 1996
Third printing, version 1.2, November 1996
Fourth printing, version 1.3, March 1997
1. Purpose ........................................................................................................................................... 4
2. Audience......................................................................................................................................... 4
3. Guidelines ....................................................................................................................................... 4
   3.1. Readability .................................................................................................................................. 4
   3.2. Portability ................................................................................................................................. 4
   3.3. Usability ..................................................................................................................................... 5
   3.4. Compile Time Name Space Resolution .................................................................................... 5
   3.5. Link Time Name Space Resolution .......................................................................................... 5
   3.6. Implementation ........................................................................................................................ 5
4. Named Entities ................................................................................................................................. 6
   4.1. Functions ................................................................................................................................... 6
   4.2. Data Type Definitions ............................................................................................................... 6
   4.3. Variables .................................................................................................................................... 6
   4.4. Structures .................................................................................................................................. 7
   4.5. Static Defines ............................................................................................................................ 7
   4.6. Files ............................................................................................................................................ 7
5. Include Files & Examples ................................................................................................................. 8
   5.1. Include File - wmbasic.h ........................................................................................................... 8
   5.2. Include File - wmapi2.h ........................................................................................................... 9
   5.3. Include File - wmerror.h .......................................................................................................... 19
1. Purpose

This document suggests guidelines and solutions for naming conventions suitable for the Workflow Management Application Programming Interface (WAPI). This document also includes the common header files for the ‘C’ programming language.

2. Audience

The intended audience of this document includes all participants in the workflow industry. It is particularly suited for the vendors involved in developing ‘C’ language bindings for the APIs they are developing. It is also intended to serve as a reference for ‘C’ programmers making use of WfMC APIs.

3. Guidelines

Successful naming conventions for the WAPI should cover the following points:
- readability
- portability
- usability
- compile time name space resolution
- link time name space resolution
- implementation

3.1. Readability

Naming conventions should be designed to facilitate readability. This can be achieved by a number of simple steps. Using long symbol names that convey the purpose of the symbol in the WAPI facilitates readability. Clearly separating the words in symbol names by using underscores or capitalizing the first letter of each word also improves readability.

The proposed naming convention spells out symbol words fully, and separates different words in the symbol name by capitalizing the first letter of each word.

3.2. Portability

Portability issues usually discourage designers from thinking about readability. Some platforms allow only limited symbol name length, and usage of characters in only one case. This in turn will make symbol names short, encrypted, and discourage the use of underscores or capitalization to separate the words in symbol names.

Most modern environments support symbol names of greater than 32 characters with at least 32 significant characters. These are the environments for which the current API set is most suitable.

The proposed solution should be carefully modified for improved portability. If deemed necessary the API would be modified specifically for those environments where symbol name lengths are restricted. This modification would start by identifying the set of platforms and operating systems supported by the coalition members, and then modifying the naming convention and the APIs to accommodate restrictions of each of those systems, trying to avoid sacrificing readability.
3.3. Usability

The WAPI naming convention should allow WAPI users to write applications that can be linked against any vendor's library with minimal changes to the source code. All WAPI calls are intended to be usable by all the prevalent programming environments currently in use. Although the this convention is biased toward 'C' and 'C++' users, the convention should allow the functions implemented to be called by environments such as Visual Basic, Smalltalk, and various 4GL and other environments.

The proposed solution requires all vendors to name the corresponding symbols with the same name. This will allow users to change WAPI vendor libraries without modifying the application source code.

The proposed solution assumes that the WAPI vendor libraries will be implemented as DLLs (dynamic linked libraries).

3.4. Compile Time Name Space Resolution

Compile time name space issues can be resolved by reserving a prefix for all WAPI symbols. This prefix should be short, and each symbol should start with the prefix. In the case that a particular WAPI vendor or user has a conflict between a WAPI symbol name and an internal symbol name, compile time symbols can be easily re-mapped with the help of C preprocessor or by editing the header files. The naming conventions should be designed so that the need for such interventions is minimized.

This solution chooses 'WM' as a prefix to all WAPI symbol names. Furthermore, different classes of symbol names have an additional letter identifying the class that the symbol belongs to, thus improving readability. For example, all WAPI types start with 'WMT'.

3.5. Link Time Name Space Resolution

Unlike compile time symbols, conflicts between link time symbols, such as multiple defined function names, are not easily correctable. Thus, the major and most important goal of a successfully designed naming convention is to minimize the chance of link time symbol conflicts. A well designed naming convention will not only allow WAPI vendors and users to link WAPI against their internal libraries without conflicts, but it will enable users to link their applications against multiple vendor WAPIs at the same time as well. This ought to be achieved without sacrificing usability. For example, requiring that each vendor uses their own prefix before each WAPI function would facilitate link time name space resolution by elimination of multiple defined symbols between different vendors’ functions linked into the same executable, but at the same time it would decrease the usability of the WAPI, since users would have to customize their application source code for each vendor's WAPI separately.

3.6. Implementation

The proposed solution assumes that the coalition will provide a common header file as a starting point for vendor implementations. This is a curtail step, since the definition of a common WAPI programming vocabulary is essential for usability issues, and at the same time it eliminates many possible misinterpretations among different vendors.
4. Named Entities

In general all of the named entities within the scope of the Workflow Management Coalition specifications should have names which start with the letters ’WM’.

The sections to come will outline specific naming conventions for the following named entities:

<table>
<thead>
<tr>
<th>Functions</th>
<th>The callable routines specified by the WfMC APIs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Type Definitions</td>
<td>The “typedefs” used to declare and create variables for the API</td>
</tr>
<tr>
<td>Variables</td>
<td>The necessary variables required to support the APIs, i.e. parameters for those functions.</td>
</tr>
<tr>
<td>Structures</td>
<td>The data structures required for passing information to and from the APIs</td>
</tr>
<tr>
<td>Files</td>
<td>The files required to support the compilation and linking of programs written with the APIs</td>
</tr>
</tbody>
</table>

4.1. Functions

All functions in the Workflow Management Coalition’s API suite should be preceded with the ‘WM’ prefix. All functions should also return the same return value as specified by the WMTErrRetType typedef. The following is an example of a function declaration:

```c
WMTErrRetType WMOpenProcessDefinitionsList (  
  in  WMTPSessionHandle psession_handle,  
  in  WMTPFilter pproc_def_filter,  
  in  WMTBoolean count_flag,  
  out WMTPQueryHandle pquery_handle,  
  out WMTPInt32 pcount);
```

4.2. Data Type Definitions

There are a few basic data types which must first be defined. The convention will be to create “typedefs” which will then be used to declare variables required by the API. The convention for “typedefs” will be to start their names with the prefix ‘WMT’ to denote Workflow Management Types.

The basic data types from which all other types are derived are defined as follows:

- WMTInt8: signed 8 bit value representation
- WMTInt16: signed 16 bit value representation
- WMTInt32: signed 32 bit value representation
- WMTUInt8: unsigned 8 bit value representation
- WMTUInt16: unsigned 16 bit value representation
- WMTUInt32: unsigned 32 bit value representation

The convention for dealing with pointer “typedefs” will be to use the prefix ‘WMTP’. For example:

```c
typedef *WMTConnectInfo WMTPConnectInfo;
```

4.3. Variables

Naming of variables in the API set should be descriptive but are not required to have a special prefix. For variables which are pointers it is recommended that ‘p’ be used as a prefix. It is presumed that for those environments that support multiple pointer dimensions that the pointers be defined as the largest supported size, e.g. for Intel based compilers the “FAR” pointer type will be used.
For variables which have a scope outside of a module, those variables should be prefixed with ‘WMGV’ to denote Workflow Management Global Variables. For various reasons the use of global variables will be discouraged but this document includes a naming specification for completeness.

4.4. Structures

Naming of structure typedefs should be done following the conventions already given for data type definitions. Naming of the actual structures should follow the conventions for naming variables.

4.5. Static Defines

All static definitions required by the WfMC APIs such as Error Values should have names beginning with ‘WM’ and be all caps with underscores where appropriate.

4.6. Files

All files required by the WfMC APIs such as ‘C’ include files should have names beginning with ‘WM’ and be limited to 8 characters with appropriate 3 character extensions.
5. Include Files & Examples

The following sections show the include files as defined to support the Workflow Management Coalition APIs in general and the Interface 2 APIs specifically. These files serve also as examples for the purpose of illustrating the usage of the naming conventions described here in this document.

5.1. Include File - wmbasic.h

File: wmbasic.h

```c
/*
 * This file is provided by the vendor for each specific platform environment.
 * It contains definitions of the basic Workflow Management types that are
 * operating system or platform dependent.
 */

#ifndef WMTBASIC_H
#define WMTBASIC_H

typedef char WMTInt8;
typedef short WMTInt16;
typedef long WMTInt32;
typedef unsigned char WMTUInt8;
typedef unsigned short WMTUInt16;
typedef unsigned long WMTUInt32;

/********************
 * WMTPointer - generic pointer representation, points to the smallest
 * addressable unit of information.
*/
typedef WMTUInt8 * WMTPointer;
define WMNULL ((WMTPointer)0)

/********************
 * WMTText - since different vendors may represent textual information in
 * different formats for language independence purposes, we treat text as
 * an opaque generic pointer.
*/
typedef WMTInt8 WMTText;
typedef WMTText *WMTFText;
typedef WMTInt8 *WMTFInt8;
typedef WMTInt16 *WMTFInt16;
typedef WMTInt32 *WMTFInt32;

/********************
 * WMTBoolean - boolean type and value definitions.
*/
typedef WMTInt8 WMTBoolean;

#define WMFalse 0
#define WMTrue (!WMFalse)

#define NAME_STRING_SIZE 64
#define UNIQUE_ID_SIZE 64

#endif
```
5.2. Include File - wmapi2.h

File: wmapi2.h

```c
/*
 * WMAPI.H
 *
 * This is the file supplied by the Workflow Management Coalition.
 * It contains standard parameters, function and value definitions.
 */
#endif WMAPI_H
#define WMAPI_H

/********************
 WM* - all global Workflow Management symbols start with WM
 WMT* - all type definitions start with WMT
 */

/********************
 Basic Type Definitions:
 Before including this file, the following types have to be type defined
 by the vendor according to the current platform definitions. All other
 WORKFLOW MANAGEMENT types are derived from these basic types.

 WMTInt8   signed 8 bit value representation
 WMTInt16  signed 16 bit value representation
 WMTInt32  signed 32 bit value representation
 WMTUInt8  unsigned 8 bit value representation
 WMTUInt16 unsigned 16 bit value representation
 WMTUInt32 unsigned 32 bit value representation
 WMTPInt8  8 bit pointer value representation
 WMTPInt16 16 bit pointer value representation
 WMTPInt32 32 bit pointer value representation
 */
#include "wmtbasic.h"

/********************
 This section defines all the structures used by the interface 2 API.
 */

/****************************
 WMTErrRetType
 */
typedef struct {
    WMTInt16      main_code;
    WMTInt16      sub_code;
} WMTErrRetType;

/****************************
 WMTConnectInfo
 */
typedef struct {
    WMTText    user_identification[NAME_STRING_SIZE];
    WMTText    password[NAME_STRING_SIZE];
    WMTText    engine_name[NAME_STRING_SIZE];
    WMTText    scope[NAME_STRING_SIZE];
    WMTConnectInfo
} WMTConnectInfo;

typedef WMTConnectInfo *WMTPConnectInfo;
```
typedef struct
{
    WMTUInt32   session_id;
    WMTPText   pprivate;
} WMTSessionHandle;

typedef WMTSessionHandle *WMTPSessionHandle;

/******************************************************
WMTFilter
*/

typedef struct
{
    WM TInt32  filter_type;
    WM TInt32  filter_length;
    WM TText   attribute_name [NAME_STRING_SIZE];
    WM TUInt32 comparison;
    WM TText   filter_string;
} WMTFilter;

typedef WMTFilter *WMTPFilter;

/** The first 255 filter types (0x00000001 to 0x000000FF) will be reserved. These
will be used for filtering on attributes of process control data and process
relevant data. */

/******************************************************
WMTQueryHandle
*/

typedef struct
{
    WMTUInt32 query_handle;
} WMTQueryHandle;

typedef WMTQueryHandle *WMTPQueryHandle;

/******************************************************
WMTWflParticipant
*/

typedef struct
{
    WM TText   wf_participant[NAME_STRING_SIZE];
} WMTWflParticipant;

typedef WMTWflParticipant *WMTPWflParticipant;

/******************************************************
WMTProcDefID
*/

typedef struct
{
    WM TText   proc_def_id[UNIQUE_ID_SIZE];
} WMTProcDefID;

typedef WMTProcDefID *WMTPProcDefID;

/******************************************************
WMTActivityID
*/

typedef struct
{
    WM TText   activity_id[UNIQUE_ID_SIZE];
} WMTActivityID;

typedef WMTActivityID *WMTPActivityID;
/**************************************************
WMTProcDefState
*/
typedef struct
{
    WMTText proc_def_state[NAME_STRING_SIZE];
} WMTProcDefState;
typedef WMTProcDefState *WMTPProcDefState;
/*-----------------------------------------------*/
WMTProcDef
*/
typedef struct
{
    WMTText process_name[NAME_STRING_SIZE];
    WMTProcID proc_def_id;
    WMTProcDefState state;
} WMTProcDef;
typedef WMTProcDef *WMTPProcDef;
/*-----------------------------------------------*/
WMTProcInstID
*/
typedef struct
{
    WMTText proc_inst_id[UNIQUE_ID_SIZE];
} WMTProcInstID;
typedef WMTProcInstID *WMTPProcInstID;
/*-----------------------------------------------*/
WMTProcInstState
*/
typedef struct
{
    WMTText proc_inst_state[NAME_STRING_SIZE];
} WMTProcInstState;
typedef WMTProcInstState *WMTPProcInstState;
/*-----------------------------------------------*/
WMTProcInst
*/
typedef struct
{
    WMTText process_name[NAME_STRING_SIZE];
    WMTProcInstID proc_inst_id;
    WMTProcID proc_def_id;
    WMTProcInstState state;
    WMTProcDep priority;
    WMTProcInstProcPart proc_participants[20];
} WMTProcInst;
typedef WMTProcInst *WMTPProcInst;
/*-----------------------------------------------*/
WMTActivityInstID
*/
typedef struct
{
    WMTText activity_inst_id[UNIQUE_ID_SIZE];
} WMTActivityInstID;
typedef WMTActivityInstID *WMTPActivityInstID;
/*-----------------------------------------------*/
typedef struct
{
    WMTText     activity_inst_state[NAME_STRING_SIZE];
} WMTActivityInstState;

typedef WMTActivityInstState *WMTPActivityInstState;

typedef struct
{
    WMTText   activity_name[NAME_STRING_SIZE];
    WMTActivityInstID activity_inst_id;
    WMTProcInstID     proc_inst_id;
    WMTActivityInstState state;
    WMTInt32          priority;
    WMTPActivityInst activity_participants[10];
} WMTActivityInst;

typedef WMTActivityInst *WMTPActivityInst;

typedef struct
{
    WMTText     work_item_id[UNIQUE_ID_SIZE];
} WMTWorkItemID;

typedef WMTWorkItemID *WMTPWorkItemID;

typedef struct
{
    WMTText            work_item_name[NAME_STRING_SIZE];
    WMTWorkItemID      work_item_id;
    WMTActivityInstID activity_inst_id;
    WMTProcInstID      proc_inst_id;
    WMTInt32          priority;
    WMTPActivityInst participant;
} WMTWorkItem;

typedef WMTWorkItem *WMTPWorkItem;

typedef WMTAttrName *WMTPAttrName;

typedef WMTAttrName[NAME_STRING_SIZE];

typedef WMTAttrName *WMTPAttrName;
WAPI - Interface 2 function declarations.

The definitions include specification of parameters with indication of the
direction of data passing:

in         for parameters with data being passed to the API function
out        for parameters with data being passed from the API function

# define in
# define out

WMTErrRetType WMConnect (  
in  WMTPConnectInfo pconnect_info,  
out WMTPSessionHandle psession_handle);
WMTErrRetType WMDisconnect (  
in  WMTPSessionHandle psession_handle);

WMTErrRetType WMOpenProcessDefinitionsList (  
in  WMTPSessionHandle psession_handle,  
in  WMTPFilter pproc_def_filter,  
in  WMTBoolean count_flag,  
out WMTPQueryHandle pquery_handle,  
out WMTPInt32 pcount);

WMTErrRetType WMFetchProcessDefinition (  
in  WMTPSessionHandle psession_handle,  
in  WMTPQueryHandle pquery_handle,  
out WMTPProcDef pproc_def_buf_ptr);

WMTErrRetType WMCloseProcessDefinitionList (  
in  WMTPSessionHandle psession_handle,  
in  WMTPQueryHandle pquery_handle);

WMTErrRetType WMOpenProcessDefinitionStatesList (  
in  WMTPSessionHandle psession_handle,  
in  WMTPProcDefID pproc_def_id,  
in  WMTPFilter pproc_def_state_filter,  
in  WMTBoolean count_flag,  
out WMTPQueryHandle pquery_handle,  
out WMTUInt32 pcount);

WMTErrRetType WMFetchProcessDefinitionState (  
in  WMTPSessionHandle psession_handle,  
in  WMTPQueryHandle pquery_handle,  
out WMTPProcDefState pproc_def_state);

WMTErrRetType WMCloseProcessDefinitionStatesList (  
in  WMTPSessionHandle psession_handle,  
in  WMTPQueryHandle pquery_handle);

WMTErrRetType WMChangeProcessDefinitionState (  
in  WMTPSessionHandle psession_handle,  
in  WMTPProcDefID pproc_def_id,  
in  WMTPProcDefState pproc_def_state);

WMTErrRetType WMCreateProcessInstance (  
in  WMTPSessionHandle psession_handle,  
in  WMTPText pproc_inst_name,  
out WMTPProcInstID pproc_inst_id);

WMTErrRetType WMStartProcess (  
in  WMTPSessionHandle psession_handle,  
in  WMTPProcInstID pproc_inst_id,  
out WMTPProcInstID pnew_proc_inst_id );

WMTErrRetType WMTerminateProcessInstance (  
in  WMTPSessionHandle psession_handle,  
in  WMTPProcInstID pproc_inst_id);
WMTErrRetType WMOpenProcessInstanceStatesList (in WMTPSessionHandle psession_handle,
in WMTPProcInstID pproc_inst_id,
in WMTPFilter pproc_inst_state_filter,
in WMTPBoolean count_flag,
out WMTPQueryHandle pquery_handle,
out WMTPInt32 pcount);

WMTErrRetType WMFetchProcessInstanceState (in WMTPSessionHandle psession_handle,
in WMTPQueryHandle pquery_handle,
out WMTPProcInstState pproc_inst_state);

WMTErrRetType WMCloseProcessInstanceStatesList (in WMTPSessionHandle psession_handle,
in WMTPQueryHandle pquery_handle);

WMTErrRetType WMChangeProcessInstanceState (in WMTPSessionHandle psession_handle,
in WMTPProcInstID pproc_inst_id,
in WMTPProcInstState pproc_inst_state);

WMTErrRetType WMOpenProcessInstanceAttributesList (in WMTPSessionHandle psession_handle,
in WMTPProcInstID pproc_inst_id,
in WMTPFilter pproc_inst_attr_filter,
in WMTPBoolean count_flag,
out WMTPQueryHandle pquery_handle,
out WMTPInt32 pcount);

WMTErrRetType WMFetchProcessInstanceAttribute (in WMTPSessionHandle psession_handle,
in WMTPQueryHandle pquery_handle,
out WMTPAttrName pattribute_name,
out WMTPInt32 pattribute_type,
out WMTPInt32 pattribute_length,
out WMTPText pattribute_value,
in WMTPInt32 buffer_size);

WMTErrRetType WMCloseProcessInstanceAttributesList (in WMTPSessionHandle psession_handle,
in WMTPQueryHandle pquery_handle);

WMTErrRetType WMGetProcessInstanceAttributeValue (in WMTPSessionHandle psession_handle,
in WMTPProcInstID pproc_inst_id,
in WMTPAttrName pattribute_name,
out WMTPInt32 pattribute_type,
out WMTPInt32 pattribute_length,
out WMTPText pattribute_value,
in WMTPInt32 buffer_size);

WMTErrRetType WMAssignProcessInstanceAttribute (in WMTPSessionHandle psession_handle,
in WMTPAttrName pattribute_name,
in WMTPInt32 attribute_type,
in WMTPText pattribute_value);

WMTErrRetType WMOpenActivityInstanceStatesList (in WMTPSessionHandle psession_handle,
in WMTPProcInstID pproc_inst_id,
in WMTPActivityInstID pactivity_inst_id,
in WMTPFilter pact_inst_state_filter,
in WMTPBoolean count_flag,
out WMTPQueryHandle pquery_handle,
out WMTPInt32 pcount);

WMTErrRetType WMFetchActivityInstanceState (in WMTPSessionHandle psession_handle,
in WMTPQueryHandle pquery_handle,
out WMTPActivityInstState pactivity_inst_state);
WMTErrRetType WMCloseActivityInstanceStatesList (  
in WMTPSessionHandle psession_handle,  
in WMTPQueryHandle pquery_handle);

WMTErrRetType WMChangeActivityInstanceState (  
in WMTPSessionHandle psession_handle,  
in WMTPProcInstID pproc_inst_id,  
in WMTPActivityInstID pactivity_inst_id,  
in WMTPActivityInstState pactivity_inst_state);

WMTErrRetType WMOpenActivityInstanceAttributesList (  
in WMTPSessionHandle psession_handle,  
in WMTPProcInstID pproc_inst_id,  
in WMTPActivityInstID pactivity_inst_id,  
in WMTPFilter pact_inst_attr_filter,  
in WMTBoolean count_flag,  
out WMTPQueryHandle pquery_handle,  
out WMTPInt32 pcount);

WMTErrRetType WMChangeActivityInstanceAttribute (  
in WMTPSessionHandle psession_handle,  
in WMTPProcInstID pproc_inst_id,  
in WMTPActivityInstID pactivity_inst_id,  
in WMTPFilter pact_inst_attr_filter,  
in WMTBoolean count_flag,  
out WMTPQueryHandle pquery_handle,  
out WMTPInt32 pcount);

WMTErrRetType WMGetActivityInstanceAttributeValue (  
in WMTPSessionHandle psession_handle,  
in WMTPProcInstID pproc_inst_id,  
in WMTPActivityInstID pactivity_inst_id,  
in WMTPFilter pact_inst_attr_filter,  
in WMTBoolean count_flag,  
out WMTPQueryHandle pquery_handle,  
out WMTPInt32 pcount);

WMTErrRetType WMAssignActivityInstanceAttribute (  
in WMTPSessionHandle psession_handle,  
in WMTPProcDefID pproc_def_id,  
in WMTPProcInstID pproc_inst_id,  
in WMTPActivityInstID pactivity_inst_id,  
in WMTPAttrName pattr_name,  
in WMTPAttrType pattr_type,  
in WMTPAttrLength pattr_length,  
in WMTPText pattr_value,  
in WMTPInt32 buffer_size);

WMTErrRetType WMOpenProcessInstancesList (  
in WMTPSessionHandle psession_handle,  
in WMTPFilter pproc_inst_filter,  
in WMTBoolean count_flag,  
out WMTPQueryHandle pquery_handle,  
out WMTPInt32 pcount);

WMTErrRetType WMGetProcessInstance (  
in WMTPSessionHandle psession_handle,  
in WMTPProcInstID pproc_inst_id);

WMTErrRetType WMCloseProcessInstancesList (  
in WMTPSessionHandle psession_handle,  
in WMTPQueryHandle pquery_handle);

WMTErrRetType WMGetProcessInstance (  
in WMTPSessionHandle psession_handle,  
in WMTPProcInstID pproc_inst_id,  
out WMTPProcInst pproc_inst);
WMTErrRetType WMOpenActivityInstancesList ( 
in WMTPSessionHandle psession_handle, 
in WMTPFilter pactivity_inst_filter, 
in WMTPBoolean count_flag, 
out WMTPQueryHandle pquery_handle, 
out WMTPInt32 pcount);

WMTErrRetType WMGetActivityInstance ( 
in WMTPSessionHandle psession_handle, 
in WMTPActivityInst pactivity_inst);

WMTErrRetType WMCloseActivityInstancesList ( 
in WMTPSessionHandle psession_handle, 
in WMTPQueryHandle pquery_handle);

WMTErrRetType WMTErrRetType WMGetActivityInstance ( 
in WMTPSessionHandle psession_handle, 
in WMTPProcInstID pproc_inst_id, 
in WMTPActivityInstID pactivity_inst_id, 
out WMTPActivityInst pactivity_inst );

WMTErrRetType WMOpenWorkList ( 
in WMTPSessionHandle psession_handle, 
in WMTPFilter pworklist_filter, 
in WMTPBoolean count_flag, 
out WMTPQueryHandle pquery_handle, 
out WMTPInt32 pcount);

WMTErrRetType WMFetchWorkItem ( 
in WMTPSessionHandle psession_handle, 
in WMTPQueryHandle pquery_handle, 
out WMTPWorkItem pwork_item);

WMTErrRetType WMCloseWorkList ( 
in WMTPSessionHandle psession_handle, 
in WMTPQueryHandle pquery_handle);

WMTErrRetType WMGetWorkItem ( 
in WMTPSessionHandle psession_handle, 
in WMTPProcInstID pproc_inst_id, 
in WMTPWorkItemID pwork_item_id, 
out WMTPWorkItem pwork_item );

WMTErrRetType WMCompleteWorkItem ( 
in WMTPSessionHandle psession_handle, 
in WMTPProcInstID pproc_inst_id, 
in WMTPWorkItemID pwork_item_id);

WMTErrRetType WMReassignWorkItem ( 
in WMTPSessionHandle psession_handle, 
in WMTPWorkflowParticipant psource_user, 
in WMTPWorkflowParticipant ptarget_user, 
in WMTPProcInstID pproc_inst_id, 
in WMTPWorkItemID pwork_item_id);

WMTErrRetType WMOpenWorkItemAttributesList ( 
in WMTPSessionHandle psession_handle, 
in WMTPProcInstID pproc_inst_id, 
in WMTPWorkItemID pwork_item_id, 
in WMTPFilter pwork_item_attr_filter, 
in WMTPBoolean count_flag, 
out WMTPQueryHandle pquery_handle, 
out WMTPInt32 pcount);

WMTErrRetType WMFetchWorkItemAttribute ( 
in WMTPSessionHandle psession_handle, 
in WMTPQueryHandle pquery_handle, 
out WMTPAttrName pattribute_name, 
out WMTPInt32 pattribute_type, 
out WMTPInt32 pattribute_length, 
out WMTPText pattribute_value, 
in WMTPInt32 buffer_size);
WMTErrRetType WMCloseWorkItemAttributesList (  
in WMTPSessionHandle psession_handle,  
in WMTPQueryHandle pquery_handle);

WMTErrRetType WMGetWorkItemAttributeValue (  
in WMTPSessionHandle psession_handle,  
in WMTPProcInstID pproc_inst_id,  
in WMTPWorkItemID pwork_item_id,  
in WMTPAttrName pattribute_name,  
out WM TInt32 pattribute_type,  
out WM TInt32 pattribute_length,  
out WMTPText pattribute_value,  
in WM TInt32 buffer_size);

WMTErrRetType WMAssignWorkItemAttribute (  
in WMTPSessionHandle psession_handle,  
in WMTPProcInstID pproc_inst_id,  
in WMTPWorkItemID pwork_item_id,  
in WMTPAttrName pattribute_name,  
in WM TInt32 attribute_type,  
in WM TInt32 attribute_length,  
in WMTPText pattribute_value);

WMTErrRetType WMExecuteWorkItem (  
in WMTPSessionHandle psession_handle,  
in WMTPProcInstID pproc_inst_id,  
in WMTPWorkItemID pwork_item_id )

WMTErrRetType WMChangeProcessInstancesState (  
in WMTPSessionHandle psession_handle,  
in WMTPProcDefID pproc_def_id,  
in WMTPFilter pproc_inst_filter,  
in WMTPProcInstState pproc_inst_state);

WMTErrRetType WMChangeActivityInstancesState (  
in WMTPSessionHandle psession_handle,  
in WMTPProcDefID pproc_def_id,  
in WMTPActivityID pactivity_def_id,  
in WMTPFilter pactivity_inst_filter,  
in WMTPActivityInstState pactivity_inst_state);

WMTErrRetType WMTerminateProcessInstances (  
in WMTPSessionHandle psession_handle,  
in WMTPProcDefID pproc_def_id,  
in WMTPFilter pproc_inst_filter);

WMTErrRetType WMAssignProcessInstancesAttribute (  
in WMTPSessionHandle psession_handle,  
in WMTPProcDefID pproc_def_id,  
in WMTPFilter pproc_inst_filter,  
in WMTPAttrName attribute_name,  
in WM TInt32 attribute_type,  
in WM TInt32 attribute_length,  
in WMTPText pattribute_value);

WMTErrRetType WMAssignActivityInstancesAttribute (  
in WMTPSessionHandle psession_handle,  
in WMTPProcDefID pproc_def_id,  
in WMTPActivityID pactivity_def_id,  
in WMTPFilter pactivity_inst_filter,  
in WMTPAttrName attribute_name,  
in WM TInt32 attribute_type,  
in WM TInt32 attribute_length,  
in WMTPText pattribute_value);

WMTErrRetType WMAbortProcessInstances (  
in WMTPSessionHandle psession_handle,  
in WMTPProcDefID pproc_def_id,  
in WMTPFilter pproc_inst_filter);

WMTErrRetType WMAbortProcessInstance (  
in WMTPSessionHandle psession_handle,  
in WMTPProcInstID pproc_inst_id);
End of wmAPI.h
*/
#endif
5.3. Include File - wmerror.h

File: wmerror.h

```c
/*
WMERROR.H

This is the file supplied by the Workflow Management Coalition.
It contains defines for the error codes support by the WAPI.
*/

#ifndef WMERR_H
#define WMERR_H

#define WM_MAIN_ERR_MASK 0xFFFF0000
#define WM_SUB_ERR_MASK 0x0000FFFF
#define WM_SUCCESS 0
#define WM_CONNECT_FAILED 0x00100000
#define WM_INVALID_PROCESS_DEFINITION 0x00200000
#define WM_INVALID_ACTIVITY_NAME 0x00300000
#define WM_INVALID_PROCESS_INSTANCE 0x00400000
#define WM_INVALID_ACTIVITY_INSTANCE 0x00500000
#define WM_INVALID_PROCESS_INSTANCE 0x00600000
#define WM_INVALID_ATTRIBUTE 0x00700000
#define WM_ATTRIBUTE_ASSIGNMENT_FAILED 0x00800000
#define WM_INVALID_STATE 0x00900000
#define WM_TRANSITION_NOT_ALLOWED 0x00A00000
#define WM_INVALID_SESSION_HANDLE 0x00B00000
#define WM_INVALID_QUERY_HANDLE 0x00C00000
#define WM_INVALID_SOURCE_USER 0x00D00000
#define WM_INVALID_TARGET_USER 0x00E00000
#define WM_INVALID_FILTER 0x00F000000
#define WM_LOCKED 0x00F10000
#define WM_NOT_LOCKED 0x00F20000
#define WM_NO_MORE_DATA 0x00F30000
#define WM_INSUFFICIENT_BUFFER_SIZE 0x00F40000
#define WM_EXECUTE_FAILED 0x00F50000

#endif
*/
End of wmerror.h
*/
#endif
```

Include File - wmconf.h

File: wmconf.h

```c
/*
WMCONF.H

This is the file supplied by the Workflow Management Coalition.
It contains function definitions for operations related to conformance profiles.
*/

ifndef WMCONF_H
#define WMCONF_H

#include "wmtbasic.h"

/**************************************************************************
End of wmconf.h
*/
#endif
```

Version 1.4
Copyright © 1993 - 1997, The Workflow Management Coalition
WAPI - Interface 2 conformance function declarations.

*/
WMTErrRetType WMIsWorklistHandlerProfileSupported ();
WMTErrRetType WMIsProcessDefinitionProfileSupported ();
WMTErrRetType WMIsProcessControlStatusProfileSupported ();
WMTErrRetType WMIsProcessAdminProfileSupported ();
WMTErrRetType WMIsActivityControlStatusProfileSupported ();
WMTErrRetType WMIsActivityAdminProfileSupported ();
WMTErrRetType WMIsAuditRecordProfileSupported ();

/*******************************************************************/
End of WMCONF.h
*/
#endif