Services
WorkXpress

WORKXPRESS™

717-609-0029 | sales@workxpress.com | www.workxpress.com

Table of Contents

Introduction

Services WorkXpress is a Pear package for PHP that makes communicating with the WorkXpress API easier. The package allows one to make API calls using common data structures such as objects and arrays. All of the functionality of the WorkXpress API is exposed through this package. This means that anything available through the API is supported. This document refers to Services WorkXpress version 1.0.

Services WorkXpress requires PHP 5.2+ with the base PEAR library installed. For more information on PEAR, please visit http://pear.php.net. You can download Services WorkXpress from http://www.workxpress.com/sites/default/files/api/Services_WorkXpress.tar.gz.

License

Services WorkXpress is licensed under the BSD License as follows:

Getting Started

Before making any calls to the API an instance of Services WorkXpress must be instantiated and properly configured. First, make sure that your pear directory is set in your php.ini. You can do this at run time using set_include_path(). Next instantiate the object:

```
$workxpress = new Services_WorkXpress();
```

After instantiating the object there are three options that must be configured. The first is the API version. The current API version is 1. Next is the authentication key. A different auth key must be provided for each different application role – projects, testing and production. To learn more about creating an auth key, please see the API documentation. The final option is the remote host. This should be set to the address of your application.

```
$workxpress->setAPIVersion(1);
$workxpress->setAuthKey($auth_key);
$workxpress->setRemoteHost('http://example.workxpress.com');
```

Once the object has been properly configured, it's time to start constructing the request. There are four types of requests that can be made using the WorkXpress API:

- LookupData – Searches and pulls data from a WorkXpress application.
- AddItem – Adds new Items to a WorkXpress application.
- UpdateItem – Updates existing Items inside of a WorkXpress application.
- ExecuteAction – Run Actions that exist in a WorkXpress application.

All of the requests can be made either by using many method calls on the request object or passing a single array to the request object. To build the request, a request object must be instantiated using the Services_WorkXpress::loadRequest() method:

```
$request = $workxpress->loadRequest('LookupData');
/* @var $request Services_WorkXpress_Request_LookupData */
```

Each request can be made up of many data sets. A data set is a definition of a single operation. This allows for multiple lookups, updates, action executions, etc. to be made using a single call. However, request types cannot be mixed. Although you can have a single call with five different lookups, you cannot have a single call that performs a lookup and executes an Action. Each request type is detailed in the sections that follow.

LookupData Request

LookupData is a function for reading from the WorkXpress database. It allows you to define the item types and relation types that you wish to lookup, as well as filter those results. If you already know the Item id of a specific Item(s), you can also pull data from the Item(s) without performing a search.

Request Object Methods

The LookupData request object provides six methods to construct the request to the WorkXpress API:

addDataSet() - Adds a single data set to the request.

addDataSet Arguments:

$reference *(Optional)*	String	An identifier that will be returned with the response to identify each data set. If the reference is empty a random one will be created.
$data_set_array *(Optional)*	Array	An optional array that describes the contents of the data set. This argument can be used instead of using the other methods on this object.

addItem() - Adds a single item to the lookup.

addItem Arguments:

$itemId	String	The id of the Item to lookup.

addMap() - Adds a map to the request that will search for Items instead of defining individual Items. For more information on building maps, please see the "Map Builder" section of the API documentation.

addMap Arguments:

$definition	String	XML for the map definition. Services WorkXpress will pass the string through htmlentities() to make it safe for inclusion in the SOAP call.

addField() - Adds a Field to pull data from on the Items that are found.

addFIeld Arguments:

$fieldId	String	The if of the Field to pull data from.
$format *(Optional)*	String	One of the Services_WorkXpress::FIELD_FORMAT_* constants. These values correspond to the different formats that can be returned by the API. STORED (Default): Returns the data as it is stored in the WorkXpress database. TEXT: Returns the plain text format of the Field as it would be displayed inside of WorkXpress. HTML: Returns the Field exactly how it would be displayed inside of WorkXpress, including and HTML.
$format_formula *(Optional)*	String	A string of display format parts used to create a custom format. For more information, see the display format parts section of the API documentation.
$reference *(Optional)*	String	An identifier that will be returned with the response to identify each Field. If the reference is empty a random one will be created.

addRelation() - Adds a relation to the request.

addRelation Arguments:

$relationType	String	Id of the Relation Type for the Relation.
$from	String	Defines which side of the Relation to start from. Valid values are: base – The base side of the Relation Type. target – The target side of the Relation Type.
$fields	Array	An array of Fields to pull from the Relation, formated as follows: array('fieldId' => String, 'format' => String stored|text|html, 'reference' => String *(Optional)*, 'formatFormula' => String *(Optional)*,);

$reference *(Optional)*	String	An identifier that will be returned with the response to identify each Relation Type. If the reference is empty a random one will be created.

call() - Executes the API call

Example Using Array

Below is an example of a LookupData request using an array as input:

```
<?php
// load the Services_WorkXpress object
$workxpress  = new Services_WorkXpress();
$workxpress->setAPIVersion(1);
$workxpress->setAuthKey($auth_key);
$workxpress->setRemoteHost('http://example.workxpress.com');

// load the request object
$request = $workxpress->loadRequest('LookupData');
/* @var $request Services_WorkXpress_Request_LookupData */

// we'll use a map to find our items
$map_definition = '<?xml version="1.0" encoding="UTF-8"?><wxQuery
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="wxQuery.xsd"
id="root"><startingTypes><startingType>a1</startingType></startingTypes>
<data for="root" /></wxQuery>';

// build the data set array
$data_set_array = array(
    'items' => array (
        // we can also define any items that we know
        array('itemId' => 'u4'),
        array('definition' => $map_definition),
    ),
    'fields' => array (
        array(
            'fieldId' => 'a14',
            'reference' => 'Mailing Address',
            'format' => 'Services_WorkXpress::FIELD_FORMAT_TEXT,
```

```
            'formatFormula' =>  'Street - City, State ZipCode';
        ),
        array(
            'fieldId' => 'a26',
            'reference' => 'PhoneNumber',
            'format' => 'stored'
        ),
    ),
    'relations' => array(
        array(
            'relationType' => 'a47',
            'from' => 'target',
            'reference' => 'relation',
            'fields' => array(
                array(
                    'fieldId' => 'a56',
                    'format' => Services_WorkXpress::FIELD_FORMAT_HTML,
                )
            ),
        ),
    )
);
$request->addDataSet('A', $data_set_array);

/** make the API call **/
try
{
    // make the call and get the data array
    $response = $request->call();
    /* @var $response Services_WorkXpress_Response_LookupData */
    $items = $response->getDataArray(
        Services_WorkXpress::DATA_ARRAY_FORMAT_FULLY_COLLAPSED);

    // show the results
    echo '<pre>'.print_r($items,  true).'</pre>';
} // end try
catch (Services_WorkXpress_Exception $e)
{
```

```
    echo '<h1>Error</h1><pre>'.$e->getMessage().'</pre>';
} // end catch Services_WorkXpress_Exception
```

Example Using Methods

Below is an example of a LookupData request using the methods provided by Services WorkXpress:

```
<?php
// load the Services_WorkXpress object
$workxpress  = new Services_WorkXpress();
$workxpress->setAPIVersion(1);
$workxpress->setAuthKey($auth_key);
$workxpress->setRemoteHost('http://example.workxpress.com');

// load the request object
$request = $workxpress->loadRequest('LookupData');
/* @var $request Services_WorkXpress_Request_LookupData */

// we'll use a map to find our items
$map_definition = '<?xml version="1.0" encoding="UTF-8"?><wxQuery
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="wxQuery.xsd"
id="root"><startingTypes><startingType>a1</startingType></startingTypes>
<data for="root" /></wxQuery>';

// add the first data set
$request->addDataSet('A');
$request->addItem('u4');
$request->addField('a26', Services_WorkXpress::FIELD_FORMAT_STORED,
    null, 'PhoneNumber');

// define fields for the relation
$relation_fields = array(
    'fieldId' => 'a56',
       'format' => Services_WorkXpress::FIELD_FORMAT_HTML,
) // end $relation_fields

// add a second data set
$request->addDataSet('B');
$request->addMap($map_definition);
```

```
$request->addField('a14',  Services_WorkXpress::FIELD_FORMAT_TEXT,
    'Street - City, State ZipCode', 'MailingAddress');
$request->addField('a26', Services_WorkXpress::FIELD_FORMAT_STORED,
    null, 'PhoneNumber');
$request->addRelation('a47', 'target', $relation_fields, 'Relation');

/** make the API call **/
try
{
    // make the call and get the data array
    $response = $request->call();
    /* @var $response Services_WorkXpress_Response_LookupData */
    $items = $response->getDataArray(
        Services_WorkXpress::DATA_ARRAY_FORMAT_FULLY_COLLAPSED);

    // show the results
    echo '<pre>'.print_r($items,  true).'</pre>';
} // end try
catch (Services_WorkXpress_Exception $e)
{
    echo '<h1>Error</h1><pre>'.$e->getMessage().'</pre>';
} // end catch Services_WorkXpress_Exception
```

Add Item Request

AddItem is a function for creating new Items inside of WorkXpress. When adding Items through the WorkXpress API, any appropriate Item, Field and Relation Actions will be executed.

Request Object Methods

The AddItem request object provides four methods to construct the request to the WorkXpress API:

addDataSet() - Adds a single data set to the request.

addDataSet Arguments:

$reference *(Optional)*	String	An identifier that will be returned with the response to identify each data set. If the reference is empty a random one will be created.
$data_set_array *(Optional)*	Array	An optional array that describes the contents of the data set. This argument can be used instead of using the other methods on this object.

addField() - Adds a Field to set data into on the new Item.

addFIeld Arguments:

$fieldId	String	The id of the Field to set.
$value	String	The value to set into the field. See the API documentation for information on the data formats of some Fields.

addRelation() - Adds a Relation to be added with the Item

addRelation Arguments:

$relationType	String	Id of the Relation Type for the Relation.
$startingSide	String	Defines which side of the Relation the Item being added will be on. Valid values are: base – The base side of the Relation Type. target – The target side of the Relation Type.
$oppositeItemId	String	The Item id of the Item that you wish to relate the Item you are adding to.

$fields *(Optional)*	Array	An array of Fields to set on the Relation, formated as follows: array('fieldId' => String, 'value' => String,);
$reference *(Optional)*	String	An identifier that will be returned with the response to identify each Relation. If the reference is empty a random one will be created.

call() - Executes the API call

Example Using Array

Below is an example of an AddItem request using an array as input:

```
<?php
// load the Services_WorkXpress object
$workxpress  = new Services_WorkXpress();
$workxpress->setAPIVersion(1);
$workxpress->setAuthKey($auth_key);
$workxpress->setRemoteHost('http://example.workxpress.com');

// load the request object
$request = $workxpress->loadRequest('AddItem');
/* @var $request Services_WorkXpress_Request_AddItem */

// build the data set array
$data_set_array = array(
    'item' => array (
        array('itemTypeId' => 'a22'),
    ),
    'fields' => array (
        array(
            'fieldId' => 'a37',
            'value' => 123,
        ),
        array(
            'fieldId' => 'a39',
```

```
            'value' => 'Heres some Text!!',
        ),
    ),
    'relations' => array(
        array(
            'relationType' => 'a68',
            'startingSide' => 'target',
            'reference' => 'relation',
            'oppositeItemId' => 'u2',
            'fields' => array(
                array(
                    'fieldId' => 'a75',
                    'value' => 'From an array...',
                ),
            ),
        ),
    ),
); // end $data_set_array
$request->addDataSet('First Item', $data_set_array);

/** make the API call **/
try
{
    // make the call and get the data array
    $response = $request->call();
    /* @var $response Services_WorkXpress_Response_AddItem */
    $items = $response->getDataArray(
        Services_WorkXpress::DATA_ARRAY_FORMAT_FULLY_COLLAPSED);

    // show the results
    echo '<pre>'.print_r($items, true).'</pre>';
} // end try
catch (Services_WorkXpress_Exception $e)
{
    echo '<h1>Error</h1><pre>'.$e->getMessage().'</pre>';
} // end catch Services_WorkXpress_Exception
```

Example Using Methods

Below is an example of an AddItem request using the methods provided by Services WorkXpress:

```
<?php
// load the Services_WorkXpress object
$workxpress  = new Services_WorkXpress();
$workxpress->setAPIVersion(1);
$workxpress->setAuthKey($auth_key);
$workxpress->setRemoteHost('http://example.workxpress.com');

// load the request object
$request = $workxpress->loadRequest('AddItem');
/* @var $request Services_WorkXpress_Request_AddItem */

// add the data set
$request->addDataSet('First Item');

// add the item
$request->addItem('a22');
$request->addField('a37', 123);
$request->addField('a39', 'Heres some Text!!');

// setup the relation fields
$relation_fields = array(
    array(
        'fieldId' => 'a75',
        'value' => 'From an array...'
    ),
); // end $relation_fields

// add the relation
$request->addRelatoin('a68', 'target', 'u2', $relation_fields,
    'relation');

/** make the API call **/
try
```

```
{
    // make the call and get the data array
    $response = $request->call();
    /* @var $response Services_WorkXpress_Response_AddItem */
    $items = $response->getDataArray(
        Services_WorkXpress::DATA_ARRAY_FORMAT_FULLY_COLLAPSED);

    // show the results
    echo '<pre>'.print_r($items, true).'</pre>';
} // end try
catch (Services_WorkXpress_Exception $e)
{
    echo '<h1>Error</h1><pre>'.$e->getMessage().'</pre>';
} // end catch Services_WorkXpress_Exception
```

UpdateItem Request

UpdateItem is called to perform a number of different tasks on existing Items in WorkXpress.

These tasks include:

- Set Fields on Items & Relations
- Recycle Items & Relations
- Restore previously recycled Items & Relations
- Delete Items & Relations
- Create Relations to Items

The Request XML defines Fields to store, as well as Relations to create. The engine reads this definition, performs its' tasks and then returns an item node for each Item effected, along with relation nodes for each Relation that was added or updated. Actions attached to any Items, Fields or Relations affected by the call will be run.

Request Object Methods

The UpdateItem request object provides six methods to construct the request to the WorkXpress API:

addDataSet() - Adds a single data set to the request.

addDataSet Arguments:

$string	String	One of the Services_WorkXpress::DATA_SET_ACTION_* constants corresponding to the action that should be performed on the Items in the data set. DELETE – Deletes the Item from WorkXpress. Items that have been deleted cannot be recovered. RECYCLE – Recycles the Item in WorkXpress. Items that have been recycled can be resroted. RESTORE – Restores a previously recycled Item in WorkXpress. UPDATE – Updates and existing Item in WorkXpress.

$reference *(Optional)*	String	An identifier that will be returned with the response to identify each data set. If the reference is empty a random one will be created.
$data_set_array *(Optional)*	Array	An optional array that describes the contents of the data set. This argument can be used instead of using the other methods on this object.

addItem() - Adds a single item to the request.

addItem Arguments:

$itemId	String	The id of the Item to update.

addMap() - Adds a map to the request that will search for Items instead of defining individual Items. For more information on building maps, please see the "Map Builder" section of the API documentation.

addMap Arguments:

$definition	String	XML for the map definition. Services WorkXpress will pass the string through htmlentities() to make it safe for inclusion in the SOAP call.

addField() - Adds a Field to set data into on the Item(s).

addFIeld Arguments:

$fieldId	String	The id of the Field to set.
$value	String	The value to set into the field. See the API documentation for information on the data formats of some Fields.

addRelation() - Adds a Relation to be added or updated.

addRelation Arguments:

$action	String	Action to be performed on the Relation. add – Creates a new Relationship. update – Updates an existing Relationship. delete – Deleted Relationships are completely removed from WorkXpress and cannot be retrieved. recycle – Recycled Relationships are *not* removed from WorkXpress and can be restored. restore – Restores a previously recycled Relationship.
$relationType	String	Id of the Relation Type for the Relation.
$startingSide	String	Defines which side of the Relation the Item being updated will be on. Valid values are: base – The base side of the Relation Type. target – The target side of the Relation Type.
$oppositeItemId *(Optional)*	String	The Item id of the Item that you wish to relate the Item you are updating to. If no Item id is provided, all Relations of the appropriate type and direction for the Item will be found. An Item id is required for the add action.
$fields *(Optional)*	Array	An array of Fields to set on the Relation, formated as follows: array('fieldId' => String, 'value' => String,);
$reference *(Optional)*	String	An identifier that will be returned with the response to identify each Relation. If the reference is empty a random one will be created.

call() - Executes the API call

Example Using Array

Below is an example of an UpdateItem request using an array as input:

```
// load the Services_WorkXpress object
$workxpress  = new Services_WorkXpress();
$workxpress->setAPIVersion(1);
$workxpress->setAuthKey($auth_key);
$workxpress->setRemoteHost('http://example.workxpress.com');

// load the request object
$request = $workxpress->loadRequest('UpdateItem');
/* @var $request Services_WorkXpress_Request_UpdateItem */

// we'll use a map to find our items
$map_definition = '<?xml version="1.0" encoding="UTF-8"?><wxQuery
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="wxQuery.xsd"
id="root"><startingTypes><startingType>a1</startingType></startingTypes>
<data for="root" /></wxQuery>';

// create the first data set array
$data_set_array = array(
    'items' => array(
        array('itemId' => 'u4'),
    ),
    'fields' => array(
        array(
            'fieldId' => 'a18',
            'value' => 'Test Value',
        ),
    ),
    'relations' => array(
        array(
            'action' => 'add',
            'relationType' => 'a47',
            'startingSide' => 'target',
            'oppositeItemId' => 'u8',
        ),
    )
```

```
); // end $data_set_array

// add the fist data set
$request->addDataSet(Services_WorkXpress::DATA_SET_ACTION_UPDATE, 'A',
    $data_set_array);

// create the second data set array
$data_set_array = array(
    'items' => array(
        array('definition' => $map_definition),
    ),
    'fields' => array(
        array(
            'fieldId' => 'a32',
            'value' => 'Lots of text...',
        ),
    ),
    'relations' => array(
        // recycle all relations of type a47 on the items in this
        // data set
        array(
            'action' => 'recycle',
            'relationType' => 'a47',
            'startingSide' => 'target',
            'oppositeItemId' => 'u8',
         'fields' => array(
                'fieldId' => 'a56',
                'value' => 'for ALL of them?'
            ),
            'reference' => 'relationAdd',
        ),
        // add a relation between the item and item u8 with no fields
        array(
            'action' => 'add',
            'relationType' => 'a47',
            'startingSide' => 'target',
            'reference' => 'relationRecycle',
        ),
```

```
        // update all relations of type a32 on the items in this
        // data set
        array(
            'action' => 'update',
            'relationType' => 'a32',
            'startingSide' => 'base',
         'fields' => array(
                'fieldId' => 'a56',
                'value' => 'for ALL of them?'
            ),
            'reference' => 'relationUpdate',
        ),
        // delete all relations of type a32 on the items in this
        // data set
        array(
            'action' => 'delete',
            'relationType' => 'a65',
            'startingSide' => 'base',
            'reference' => 'relationDelete',
        ),
        // restore all relations of type a32 on the items in this
        // data set
        array(
            'action' => 'restore',
            'relationType' => 'a15',
            'startingSide' => 'base',
            'reference' => 'relationRestore',
        ),
    )
); // end $data_set_array

// add the second data set
$request->addDataSet(Services_WorkXpress::DATA_SET_ACTION_UPDATE, 'B',
    $data_set_array);

/** make the API call **/
try
{
```

```
    // make the call and get the data array
    $response = $request->call();
    /* @var $response Services_WorkXpress_Response_UpdateItem */
    $items = $response->getDataArray(
        Services_WorkXpress::DATA_ARRAY_FORMAT_FULLY_COLLAPSED);

    // show the results
    echo '<pre>'.print_r($items, true).'</pre>';
} // end try
catch (Services_WorkXpress_Exception $e)
{
    echo '<h1>Error</h1><pre>'.$e->getMessage().'</pre>';
} // end catch Services_WorkXpress_Exception
```

Example Using Methods

Below is an example of an UpdateItem request using the methods provided by Services WorkXpress.

```
// load the Services_WorkXpress object
$workxpress  = new Services_WorkXpress();
$workxpress->setAPIVersion(1);
$workxpress->setAuthKey($auth_key);
$workxpress->setRemoteHost('http://example.workxpress.com');

// load the request object
$request = $workxpress->loadRequest('UpdateItem');
/* @var $request Services_WorkXpress_Request_UpdateItem */

// we'll use a map to find our items
$map_definition = '<?xml version="1.0" encoding="UTF-8"?><wxQuery
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="wxQuery.xsd"
id="root"><startingTypes><startingType>a1</startingType></startingTypes>
<data for="root" /></wxQuery>';

// add the fist data set
$request->addDataSet(Services_WorkXpress::DATA_SET_ACTION_UPDATE, 'A');
$request->addItem('u4');
$request->addField('a18', 'Test Value');
```

```
// add a relation between the item and item u8 with no fields
$request->addRelation('add', 'a47', 'target', 'u8');

// add the second data set
$request->addDataSet(Services_WorkXpress::DATA_SET_ACTION_UPDATE, 'B');
$request->addMap($map_definition);
$request->addField('a32', 'Lots of text...');

// recycle all relations of type a47 on the items in this data set
$request->addRelation('recycle', 'a47', 'target', null, array(),
    'relationRecycle');

// setup the relation fields
$relation_fields = array(
    array(
        'fieldId' => 'a56',
        'value' => 'for ALL of them?',
    ),
); // end $relation_fields

// add a new relation to all of the items in this data set, with data
// on the relationship(s)
$request->addRelation('add', 'a47', 'target', 'u8', $relation_fields,
    'relationAdd');

// update all relations of type a32 on the items in this data set
$request->addRelation('update', 'a32', 'base', null, $relation_fields,
    'relationUpdate');

// delete all relations of type a65 on the items in this data set.
$request->addRelation('delete', 'a65', 'base', null, array(),
    'relationDelete');

// restore all relations of type a15 on the items in this data set.
$request->addRelation('restore', 'a15', 'base', null, array(),
    'relationRestore');

/** make the API call **/
```

```
try
{
    // make the call and get the data array
    $response = $request->call();
    /* @var $response Services_WorkXpress_Response_UpdateItem */
    $items = $response->getDataArray(
        Services_WorkXpress::DATA_ARRAY_FORMAT_FULLY_COLLAPSED);

    // show the results
    echo '<pre>'.print_r($items, true).'</pre>';
} // end try
catch (Services_WorkXpress_Exception $e)
{
    echo '<h1>Error</h1><pre>'.$e->getMessage().'</pre>';
} // end catch Services_WorkXpress_Exception
```

ExecuteAction Request

ExecuteAction is called to run Actions that already exist in the WorkXpress Application on a set of Items. The request XML defines Items and individual Actions to execute. The engine reads this definition, performs its' tasks and then returns each Item that the Action was run on.

Request Object Methods

The ExecuteAction request object provides five methods to construct the request to the WorkXpress API:

addDataSet() - Adds a single data set to the request.

addDataSet Arguments:

$reference *(Optional)*	String	An identifier that will be returned with the response to identify each data set. If the reference is empty a random one will be created.
$data_set_array *(Optional)*	Array	An optional array that describes the contents of the data set. This argument can be used instead of using the other methods on this object.

addItem() - Adds a single item to the action execution.

addItem Arguments:

$itemId	String	The id of the Item to run the Action on.

addMap() - Adds a map to the request that will search for Items instead of defining individual Items. For more information on building maps, please see the "Map Builder" section of the API documentation.

addMap Arguments:

$definition	String	XML for the map definition. Services WorkXpress will pass the string through htmlentities() to make it safe for inclusion in the SOAP call.

addAction() - Adds a an Action to be run on the Items.

addFIeld Arguments:

$actionId	String	The id of the Action to run.

call() - Executes the API call

Example Using Array

Below is an example of an Execute Action request using an array as input:

```
<?php
// load the Services_WorkXpress object
$workxpress  = new Services_WorkXpress();
$workxpress->setAPIVersion(1);
$workxpress->setAuthKey($auth_key);
$workxpress->setRemoteHost('http://example.workxpress.com');

// load the request object
$request = $workxpress->loadRequest('ExecuteAction');
/* @var $request Services_WorkXpress_Request_ExecuteAction */

// we'll use a map to find our items
$map_definition = '<?xml version="1.0" encoding="UTF-8"?><wxQuery
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="wxQuery.xsd"
id="root"><startingTypes><startingType>a121</startingType></startingType
s><data for="root" /></wxQuery>';

// build the data set array
$data_set_array = array(
    'items' => array (
        array('itemId' => 'u4'),
        array('itemId' => 'u7'),
        array('definition' => $map_definition),
    ),
    'actions' => array (
        array(
            'actionId' => 'a286',
        ),
    ),
); // end $data_set_array
$request->addDataSet('A', $data_set_array);

/** make the API call **/
try
{
```

```
    // make the call and get the data array
    $response = $request->call();
    /* @var $response Services_WorkXpress_Response_LookupData */
    $items = $response->getDataArray(
        Services_WorkXpress::DATA_ARRAY_FORMAT_NOT_COLLAPSED);

    // show the results
    echo '<pre>'.print_r($items, true).'</pre>';
} // end try
catch (Services_WorkXpress_Exception $e)
{
    echo '<h1>Error</h1><pre>'.$e->getMessage().'</pre>';
} // end catch Services_WorkXpress_Exception
```

Example Using Methods

Below is an example of an ExecuteAction request using the methods provided by Services WorkXpress.

```
<?php
// load the Services_WorkXpress object
$workxpress  = new Services_WorkXpress();
$workxpress->setAPIVersion(1);
$workxpress->setAuthKey($auth_key);
$workxpress->setRemoteHost('http://example.workxpress.com');

// load the request object
$request = $workxpress->loadRequest('ExecuteAction');
/* @var $request Services_WorkXpress_Request_ExecuteAction */

// we'll use a map to find our items
$map_definition = '<?xml version="1.0" encoding="UTF-8"?><wxQuery
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="wxQuery.xsd"
id="root"><startingTypes><startingType>a121</startingType></startingType
s><data for="root" /></wxQuery>';

// add the data set
$request->addDataSet('A');
$request->addItem('u4');
$request->addItem('u7');
```

```
$request->addMap($map_definition);
$request->addAction('a286');

/** make the API call **/
try
{
    // make the call and get the data array
    $response = $request->call();
    /* @var $response Services_WorkXpress_Response_LookupData */
    $items = $response->getDataArray(
        Services_WorkXpress::DATA_ARRAY_FORMAT_NOT_COLLAPSED);

    // show the results
    echo '<pre>'.print_r($items, true).'</pre>';
} // end try
catch (Services_WorkXpress_Exception $e)
{
    echo '<h1>Error</h1><pre>'.$e->getMessage().'</pre>';
} // end catch Services_WorkXpress_Exception
```

Response Data Arrays

There are three different formats for the response arrays that may be retrieved after making a request. Each of these formats corresponds to one of three constants:

- Services_WorkXpress::DATA_ARRAY_FORMAT_NOT_COLLAPSED

```
    array(
    0 => array(
        'reference' => 'account',
        'items' => array(
            0 => array(
                'itemId' => 'u113',
                'fields' => array(
                    0 => array(
                        'fieldId => 'e52',
                        'reference' => 'username',
                        'value' => 'tuser',
                    ),
                ),
                'relations' => array(
                    0 => array(
                        'reference' => 'relation',
                        'id' => 'u115',
                        'relationType' => 'a41',
                        'baseItemTypeId' => 'e8',
                        'baseItemId' => 'u113',
                        'targetItemTypeId' => 'a28',
                        'targetItemId' => 'u114',
                        'fields' => array(
                            0 => array(
                                'fieldId => 'a118',
                                'reference' => 'position',
                                'value' => 'Developer',
                            ),
                        ),
                    ),
                ),
            ),
        ),
    ),
);
```

- Services_WorkXpress::DATA_ARRAY_FORMAT_FULLY_COLLAPSED

```
    'accounts' => array(
    'u113' => array(
        'fields' => array(
            'e52' => array(
                'username' => 'tuser'
            ),
        ),
        'relations' => array(
            'relation' => array(
                'u115' => array(
                    'relationType' => 'a41',
                    'baseItemTypeId' => 'e8',
                    'baseItemId' => 'u113',
                    'targetItemTypeId' => 'a28',
                    'targetItemId' => 'u114',
                    'fields' => array(
                        0 => array(
                            'fieldId => 'a118',
                            'reference' => 'position',
                            'value' => 'Developer',
                        ),
                    ),
                ),
            ),
        ),
    ),
);
```

- Services_WorkXpress::DATA_ARRAY_FORMAT_PARTIALLY_COLLAPSED

```
    'accounts' => array(
    'u113' => array(
        'fields' => array(
            'e52' => array(
                'username' => array(
                    'fieldId' =>'e52',
                    'reference' => 'username',
                    'value' => 'jarmes',
                ),
            ),
        ),
        'relations' => array(
            'u115' => array(
                'reference' => array(
                    'reference' => 'relation',
                    'id' => 'u115',
                    'relationType' => 'a41',
                    'baseItemTypeId' => 'e8',
                    'baseItemId' => 'u113',
                    'targetItemTypeId' => 'a28',
                    'targetItemId' => 'u114',
                    'field' => array(
                        0 => array(
                            'fieldId => 'a118',
                            'reference' => 'position',
                            'value' => 'Developer',
                        ),
                    ),
                ),
            ),
        ),
    ),
);
```