File API

Important

It is recommended to make the Methods OS agnostic to use them with different operating systems. The main incompatibility issues in operating systems that need to be considered when implementing the method are related to file path case-sensitivity, file path separators, OS-specific line-endings, OS-specific code. For more information about cross-platform development please see section "Cross-platform development" in "Aras Innovator 40 - Programmer's Guide" document.

Item.setFileProperty()

The Item function `setFileProperty()` takes in 2 parameters: property and file path.

- ***Property*** **–** The property of type file for which you want to set the file item for add
- ***File path*** **–** The file path to the file on the file system

Example:

```
Item cadItem = inn.newItem("CAD", "add");
cadItem.setFileProperty("native_file", filePath);
...
```

Item.fetchFileProperty()

The function `fetchFileProperty()` takes in 3 parameters; property, file path, and mode. It returns *Item* of type *File*.

- ***Property*** **–** The property of type file for which you want to download the file
- ***Target path*** **–** The file path on the file system where to place the file
  - A file name can be specified as to change the name of the file i.e., C:\ArasTest\MyNewFileName.txt
- ***Mode*** **–** The mode of operation for checkout
  - `FetchFileMode.Normal -` In this mode, method performs file downloading to a defined target path. The return item has the `checkedout_path` property set to the value of the path where the file is downloaded.
  - `FetchFileMode.Dry -` In this mode, the method performs a “dry” run without downloading the physical file but checks if a file with same name exists in the specified

target path. The return item has the `checkedout_path` property set to the value of the path to where the file would be downloaded. You can then use this path to determine if a file already exists in this path for further logic handling. No file is downloaded in this mode.

Example:

```
Item cadItem = inn.newItem("CAD", "add");
Item myFile = cadItem.fetchFileProperty("native_file", filePath, FetchFileMode.Normal);
...
```

replaceFile method

A server method called replaceFile can be executed by File Administrators and replaces all references of the specified old file with the new file you specify.

This server method can be called with the following syntax:

```
<AML>
<Item type="File" action="replaceFile" id="oldFileID" newId="newFileID" />
</AML>
```

JavaScript functions

To upload a File using method code on the client-side, the following examples may be referenced.

To download a File through javascript, use the downloadFile function. For example:

```
aras.downloadFile(myFileItem.node);
```

Client-side Action to prompt for File:

```
function processAddingFile(file) {
var newFile = aras.newItem("File", file);
if (newFile) {
aras.itemsCache.addItem(newFile);
var res = aras.saveItemEx(newFile);
aras.unlockItemEx(res);
}
}
aras.vault.selectFile().then(processAddingFile.bind(this));
```

Form onClick event for Button:

```
function processAddingFile(file) {
      var newFile = aras.newItem("File", file);
      if (newFile) {
            aras.itemsCache.addItem(newFile);
            window.handleItemChange("myFileProperty", newFile);
      }
}
aras.vault.selectFile().then(processAddingFile.bind(this));
```

Client-side setFileProperty Example:

```
aras.vault.selectFile().then(function (fileObject)
{
    var d = aras.IomInnovator.newItem('CAD', 'add');
    d.setProperty('item_number', 'CAD_001');
    d.setFileProperty('native_file', fileObject);
    d.applyAsync();
});
```