WebSphere Product Center Scripting Reference (wpc_5320-if2_03)

Table of Contents

Scripting Operations Reference

basic

add

- **void Array::add(elements)**
  To add elements to an Array

break-continue

- **break|continue]**
  To break/continue from a loop

for

- **for(init-statement; cond; each-statement) { t-statements }**
  Equivalent to doing init-statement; while(cond) {t-statements; each-statement;}

getStringValueForClassMember

- **String getStringValueForClassMember(String className, String memberName)**
  This operation uses java reflection mechanisms to return the value of the specified static member for the named class as a string. className needs to be the fully qualified name of the class, e.g. java.lang.String

if-else

- **if(Boolean cond) { t-statements } [else { f-statements }]**
  If cond is true t-statements are executed, otherwise f-statements are executed

remove

- **void Array::remove(int i)**
  To remove the element at the specified position

return

- **return e**
  Used in functions: returns e to the caller

runQuery

- **void runQuery(String qryString)**
  run query in qryString against default db

while

- **while(Boolean cond) { t-statements }**
  As long as cond is true, t-statements are executed

bidi

bidiTransform

- **public String bidiTransform(String srcStr, String direction, String typeOfText, String orientation, String swap, String numShapes, String textShapes)**
  If direction is "IMPORT", using the BiDi attributes specified in the parameters to create a BiDiText and then tranform it to BiDiText with default attributes. If direction is

"EXPORT", create a BiDiText using default attribute then tranform it to BiDiText with attributes specified in the parameters. typeOfText can be : "IMPLICIT", "VISUAL". orientation can be : "LTR", "RTL", "CONTEXTUAL_LTR", "CONTEXTUAL_RTL". swap can be : "YES", "NO". numShapes can be : "NOMINAL", "NATIONAL", "CONTEXTUAL", "ANY". textShapes can be : "NOMINAL", "SHAPED", "INITIAL", "MIDDLE", "FINAL", "ISOLATED". default value is: typeOfText:"IMPLICIT" orientation:"LTR" swap:"YES" numShapes:NOMINAL textShapes:NOMINAL

currency

getAllCurrencies

- **String[] getAllCurrencies()**
  This operation return all currency codes that wpc support.

getCompanyCurrencies

- **String[] getCompanyCurrencies()**
  This operation return currencies code selected in company attribute.

getCurrencyDescByCode

- **String getCurrencyDescByCode(String code)**
  This operation return currency description from currency code

getCurrencySymbolByCode

- **String getCurrencySymbolByCode(String code)**
  This operation return currency symbol from currency code,such as input "USD",currency symbol return will be "$".

setCompanyCurrencies

- **void setCompanyCurrencies(String listOfCodes[])**
  This operation set the list of codes to the company datebase.

date

addDate

- **Date Date::addDate(String field, Integer value)**
  Add the integer value given to the field specified. Allowed field values are : YEAR MONTH DATE HOUR MINUTE

formatDate

- **String Date::formatDate(String newFormat[,Locale locale])**
  Use to format a date as a human readable format. The newFormat string is a pattern whose format is identical to the format used by Java. Locale is optional, default is the UI locale.

getDateField

- **Integer Date::getDateField(String field)**
  Get the value of the field specified. Allowed field values are : YEAR MONTH DATE HOUR_OF_DAY MINUTE SECOND

getDateInputFormat

- **String getDateInputFormat()**
  Returns the date input format set in my setting

getDateOutputFormat

- **String getDateOutputFormat()**
  Returns the date output format set in my setting

getDateTimeInUserTimeZone

- **Date getDateTimeInUserTimeZone()**
  Returns the number of seconds since January 1, 1970, 00:00:00 GMT represented by this Date object

getTime

- **Integer Date::getTime()**
  Returns the number of seconds since January 1, 1970, 00:00:00 GMT represented by this Date object

isDateAfter

- **Boolean Date::isDateAfter(Date otherDate)**
  Returns true if and only if this date is after otherDate

isDateBefore

- **Boolean Date::isDateBefore(Date otherDate)**
  Returns true, if and only if this date is before otherDate

new$Date

- **new Date(String sFormat, String sDate[,Locale locale)**
  Builds a Date object from a String given a date format, if the locale is supplied that locale will be used to apply the given format, else default_locale from common.properties will be used

parseDate

- **Date parseDate(String value, String format[,Locale locale])**
  Use to parse a String value to a Date object. The format string is a pattern whose format is identical to the format used by Java. Locale is optional, the default value is the UI locale.

reformatDate

- **String reformatDate (String formattedDateString, String currentDateFormat, Locale currentLocale [, String newDateFormat, Locale newLocale])**
  Takes a date string formatted according to the pattern indicated by currentDateFormat and returns a new string formatted according to the newDateFormat provided.If currentDateFormat is null the default format for the locale is used. If currentLocale or newLocale is null the locale in the user setting is used. If newDateFormat is null the standard default pattern "EEE MMM dd HH:mm:ss zzz yyyy" is used.

setDateField

- **Date Date::setDateField(String field, Integer value)**
  Return a Date equal to the input Date, except that the specified field is set to the given value. Allowed field values are : YEAR MONTH DATE HOUR_OF_DAY MINUTE SECOND

setDateInputFormat

- **void setDateInputFormat(String format)**
  Set the Date input format

setDateOutputFormat

- **void setDateOutputFormat(String format)**
  Set the Date output format

today

- **Date today ()**
  Returns the current date and time

db

commit

- **void Connection::commit()**
  Commit a transaction using the DB connection

executeBatchUpdate

- **void Connection::executeBatchUpdate(String sql, Object[][] batchValues)**
  Executes a prepared statement for a batch update using the connection object. The Object[][] is a HashMap of HashMaps, each indexed by integer, whose value is the replacement for a '?' in the prepared statement for a given batch.

executeQuery

- **ResultSet Connection::executeQuery(String sql)**
  Execute the query using the Connection object. Returns the ResultSet.

executeUpdate

- **int Connection::executeUpdate(String sql)**
  Execute the query using the Connection object. Returns the number of rows inserted, updated, or deleted.

getColumn

- **Object ResultSet::getColumn(String colName)**
  Get the entry for the current result at column colName. Returns an object of type Integer, String, or Date (depending on the data type of the column).

getColumnAt

- **Object ResultSet::getColumnAt(Number colNumber)**
  Get the entry for the current result at column position. Returns an object of type Integer, String, or Date (depending on the data type of the column).

getWPCDBConnection

- **Connection DBContext::getWPCDBConnection()**
  Get a non-auto commit connection using the DB context

getWPCDBContext

- **DBContext getWPCDBContext()**
  Get the database context object

loadJar

- **Boolean loadJar(String jarName)**
  loadJar dynamically adds the jar file of name jarName to the SystemClassLoader. This allows subsequent script operations (such as createJavaMethod) to use those class files within the jar file. The jarName is specified as the fully qualified file name of the jar on the server. The operation returns false if the file cannot be accessed. The operation returns true if the dynamic load was successful. If two loadJar calls are issued with the same fully qualified jarName and the first was successful, then the second call will return true and will not add the jar file again.

next

- **boolean ResultSet::next()**
  Move the ResultSet iterator to the next result. Returns false if it has iterated past the last result.

openJDBCConnection

- **Connection openJDBCConnection(String driverName, String database, String userid, String password)**
  Get an (autoCommit on) SQL connection using JDBC Drivers

releaseJDBCConnection

- **void releaseJDBCConnection(Connection conn)**
  Rollback and release an SQL connection retrieved using JDBC

releaseWPCDBConnection

- **void DBContext::releaseWPCDBConnection(Connection conn)**
  Rollback and release a connection retrieved using the DB context

rollback

- **void Connection::rollback()**
  Rollback a transaction using the DB connection

error

catchError

- **catchError(String errMsg) { statements }**
  Analogous to a try-catch in Java, all statements are executed and errMsg is set to null in the absence of errors

logDebug

- **void logDebug (String message)**
  Logs the debug message with the debug log that is available from the schedule profile details screens. Use with caution because the debug log is maintained in memory.

logError

- **void logError(String itemId, String message)**
  Logs the error message with the corresponding item id to the location specified in the context

logWarning

- **void logWarning(String itemId, String message)**
  Logs the warning message with the corresponding item id to the location specified in the context

throwError

- **void throwError (String rejectionCause)**
  Use to throw a Java-like exception. This operation is usually used in conjunction with the catchError operation

excel

createExcelCell

- **IExcelCell ExcelRow::createExcelCell(index)**
  Returns am ExcelCell at the requested index within the ExcelRow.

createExcelCellStyle

- **ExcelCellStyle ExcelBook::createExcelCellStyle()**
  Returns a cell style associated with this ExcelBook. A style constains characteristics of a cell over and above the value such as the font and the fillPattern. A style is applied to a cell using ExcelCell::setExcelStyle.

createExcelSheet

- **ExcelSheet ExcelBook::createExcelSheet([String sSheetName])**
  Creates a sheet from the workbook. If a sheet name is supplied then the sheet is created with this name.

createFont
- **ExcelCellFont ExcelBook::createFont()**
  Returns a ExcelCellFont associated with this ExcelBook, the ExcelFont set methods should be used to setup the font as required. This font can then be used as the input parameter to ExcelCellStyle::setFont(font). The ExcelCellStyle can then be seton an cell using ExcelCell::setExcelStyle(cellStyle).

createRow
- **IExcelRow ExcelSheet::createRow(index)**
  Returns am ExcelRow at the requested index.

getCellObj
- **ExcelCell ExcelRow::getCellObj(Integer columnIndex)**
  Returns the Excel Obj at the given column index for further investigation.

getDateCellValue
- **Date ExcelCell::getDateCellValue()**
  Returns the value of this date cell as a date. Use this function only if it is pre determined using String ExcelCell::getExcelCellType() (unless known otherwise) that the cell is a date type.

getDateFromDoubleValue
- **Date getDateFromDoubleValue(Double dateAsDoubleValue)**
  Creates a Date Object from a given Double value.

getExcelCell
- **String ExcelSheet::getExcelCell(Integer iRow, Integer iCol)**
  Returns the value of the cell at given row and column indexes as a String value.

getExcelCellEncoding
- **String ExcelCell::getExcelCellEncoding()**
  Returns the encoding of the cell. Possible values can be COMPRESSED_UNICODE, UTF-16, UNKNOWN.

getExcelCellType
- **String ExcelCell::getExcelCellType()**
  Returns the type of this cell. Values can be NUMERIC, STRING, DATE, BLANK, UNKNOWN.

getExcelRow
- **ExcelRow ExcelSheet::getExcelRow(Integer iRow)**
  Returns the row at the specified index. Note: Rows are zero based.

getExcelSheet
- **ExcelSheet ExcelBook::getExcelSheet(Integer iSheetNumber/String sSheetName)**
  Returns a sheet from the workbook based on the arguments passed. If iSheetNumber is passed, then the sheet having the index specified by this argument is returned. If sSheetName is passed, then the sheet is retrieved by the name.

getExcelSheets
- **HashMap ExcelBook::getExcelSheets()**
  Returns a hashmap of excel sheets within the workbook. The hashmap is indexed by the sheet name.

getFirstCellNum
- **Integer ExcelRow::getFirstCellNum()**
  Returns the index of the first physical cell in this row. Note that columns are zero based.

getFirstRowNum
- **Integer ExcelSheet::getFirstRowNum()**
  Returns the index of the first physical row

getLastCellNum
- **Integer ExcelRow::getLastCellNum()**
  Returns the index of the last physical cell in this row.

getLastRowNum
- **Integer ExcelSheet::getLastRowNum()**
  Returns the index of the last physical row

getNbColumns
- **Integer ExcelSheet::getNbColumns()**
  Returns the number of physical columns in this sheet

getNbRows
- **Integer ExcelSheet::getNbRows()**
  Returns the number of physical rows in this sheet

getNumericCellValue
- **Double ExcelCell::getNumericCellValue()**
  Returns the value of this numeric cell as a double value. Use this function only if it is pre determined using String ExcelCell::getExcelCellType() (unless known otherwise) that the cell is a numeric type.

getStringCellValue
- **String ExcelCell::getStringCellValue()**
  Returns the value of this text cell as a String. Use this function only if it is pre determined using String ExcelCell::getExcelCellType() (unless known otherwise) that the cell is a string type.

new$ExcelBook
- **new ExcelBook([Doc docToRead])**
  This creates a new ExcelBook. ExcelBooks can be used in 2 ways, to read an existing ExcelBook or to create a new ExcelBook. It is not supported to update an existing ExcelBook. All the other Excel objects can be obtained either directly from this ExcelBook or indirectly from the objects obtained from this ExcelBook (apart from the ExcelParser object which has its own constructor). When the script op is used within an import job, then an existing ExcelBook is read from the docstore (the default import document or the one that is specified in the docToRead parameter). When an existing ExcelBook is read, it should not be updated, as any changes will not be written back to the docstore. To create an ExcelBook which will be updated, this script operation should be called outside of an import job, without the docToRead parameter; this creates an ExcelBook in memory which can then saved to the docstore using the ExcelBook::saveToDocStore script operation.

new$ExcelParser
- **new ExcelParser(ExcelSheet sheet)**
  Returns an excel parser to parse the given spreadsheet.

saveToDocStore
- **void ExcelBook::saveToDocStore([[String docStorePath],Boolean overwriteFlag]])**
  This saved an updated ExcelBook to the documentation store. If used in an export script with no operands, then the excel file will be written into the standard export directory with name CATALOG.XLS. If run with no operands outside of an export, then this script operation will fail with an exception. When a docStorePath argument

is supplied, then this is absolute path including the file name where the excel book will be written in the doc store. When the overWriteFlag is set to true, then any existing excel book at the supplied path will be overwritten, If the overWriteFlag is set to false and excelbook existing in the docstore path, an exception will be thrown. If the overwriteFlag is not supplied, then it will default to false. It is recommended that you do not specify a docstore path in export scripts, as subsequent runs of the export will attempt to write to the same file in the doc store (which will only succeed if the overwrite flag is set to true)

setAlignment

- **void ExcelCellStyle::setAlignment(String alignment)**
  Set the cell style alignment to that supplied. The valid alignments are ALIGN_GENERAL, ALIGN_LEFT, ALIGN_CENTER, ALIGN_RIGHT, ALIGN_FILL, ALIGN_JUSTIFY, ALIGN_CENTER_SELECTION.

setBoldWeight

- **void ExcelCellFont::setBoldWeight(String weight)**
  Set the cell font bold weight. Valid Strings are BOLDWEIGHT_NORMAL and BOLDWEIGHT_BOLD.

setBorderBottom

- **void ExcelCellStyle::setBorderBottom(String border)**
  Set the Bottom Border to the supplied border. The valid borders are BORDER_NONE, BORDER_THIN, BORDER_MEDIUM, BORDER_DASHED, BORDER_HAIR, BORDER_THICK, BORDER_DOUBLE, BORDER_DOTTED, BORDER_MEDIUM_DASHED, BORDER_DASH_DOT, BORDER_MEDIUM_DASH_DOT, BORDER_DASH_DOT_DOT, BORDER_MEDIUM_DASH_DOT_DOT, BORDER_SLANTED_DASH_DOT.

setBorderLeft

- **void ExcelCellStyle::setBorderLeft(String border)**
  Set the left Border to the supplied border. The valid borders are BORDER_NONE, BORDER_THIN, BORDER_MEDIUM, BORDER_DASHED, BORDER_HAIR, BORDER_THICK, BORDER_DOUBLE, BORDER_DOTTED, BORDER_MEDIUM_DASHED, BORDER_DASH_DOT, BORDER_MEDIUM_DASH_DOT, BORDER_DASH_DOT_DOT, BORDER_MEDIUM_DASH_DOT_DOT, BORDER_SLANTED_DASH_DOT.

setBorderRight

- **void ExcelCellStyle::setBorderRight(String border)**
  Set the Right Border to the supplied border. The valid borders are BORDER_NONE, BORDER_THIN, BORDER_MEDIUM, BORDER_DASHED, BORDER_HAIR, BORDER_THICK, BORDER_DOUBLE, BORDER_DOTTED, BORDER_MEDIUM_DASHED, BORDER_DASH_DOT, BORDER_MEDIUM_DASH_DOT, BORDER_DASH_DOT_DOT, BORDER_MEDIUM_DASH_DOT_DOT, BORDER_SLANTED_DASH_DOT.

setBorderTop

- **void ExcelCellStyle::setBorderTop(String border)**
  Set the Top Border to the supplied border. The valid borders are BORDER_NONE, BORDER_THIN, BORDER_MEDIUM, BORDER_DASHED, BORDER_HAIR, BORDER_THICK, BORDER_DOUBLE, BORDER_DOTTED, BORDER_MEDIUM_DASHED, BORDER_DASH_DOT, BORDER_MEDIUM_DASH_DOT, BORDER_DASH_DOT_DOT, BORDER_MEDIUM_DASH_DOT_DOT, BORDER_SLANTED_DASH_DOT.

setBottomBorderColor

- **void ExcelCellStyle::setBottomBorderColor(String color)**
  Set the cell style Bottom border color to that supplied. The valid colors are BROWN, OLIVE_GREEN,DARK_GREEN, DARK_TEAL,DARK_BLUE,

INDIGO,GREY_80_PERCENT, DARK_RED,ORANGE, DARK_YELLOW,GREEN, TEAL, BLUE,BLUE_GREY, GREY_50_PERCENT,RED, LIGHT_ORANGE,LIME, SEA_GREEN,AQUA, LIGHT_BLUE,VIOLET, GREY_40_PERCENT,PINK, GOLD, YELLOW,BRIGHT_GREEN, TURQUOISE,SKY_BLUE, PLUM,GREY_25_PERCENT, ROSE,TAN, LIGHT_YELLOW,LIGHT_GREEN, LIGHT_TURQUOISE,PALE_BLUE, LAVENDER,WHITE, CORNFLOWER_BLUE,LEMON_CHIFFON, MAROON,ORCHID, CORAL,ROYAL_BLUE, LIGHT_CORNFLOWER_BLUE.

setCellType

- **void ExcelCell::setCellType(String type)**
  Set the cell type. Valid values are BLANK, NUMERIC, STRING. Be aware that the NUMERIC type will change to a DATE type of cell if the value of the cell is a Date.

setColor

- **void ExcelCellFont::setColor(String color)**
  Set the cell fonts color to that supplied. The valid colors are BROWN, OLIVE_GREEN,DARK_GREEN, DARK_TEAL,DARK_BLUE, INDIGO,GREY_80_PERCENT, DARK_RED,ORANGE, DARK_YELLOW,GREEN, TEAL, BLUE,BLUE_GREY, GREY_50_PERCENT,RED, LIGHT_ORANGE,LIME, SEA_GREEN,AQUA, LIGHT_BLUE,VIOLET, GREY_40_PERCENT,PINK, GOLD, YELLOW,BRIGHT_GREEN, TURQUOISE,SKY_BLUE, PLUM,GREY_25_PERCENT, ROSE,TAN, LIGHT_YELLOW,LIGHT_GREEN, LIGHT_TURQUOISE,PALE_BLUE, LAVENDER,WHITE, CORNFLOWER_BLUE,LEMON_CHIFFON, MAROON,ORCHID, CORAL,ROYAL_BLUE, LIGHT_CORNFLOWER_BLUE.

setDataFormat

- **void ExcelCellStyle::setDataFormat(String format)**
  Set the cell style Data Format to that supplied. The valid formats are General, 0, 0.00, #, ##0, #, ##0.00, ($#, ##0_);($#, ##0), ($#, ##0_);[Red]($#, ##0), ($#, ##0.00);($#, ##0.00), ($#, ##0.00_);[Red]($#, ##0.00), 0%, 0.00%, 0.00E+00, # ?/?, # ??/??, m/d/yy, d-mmm-yy, d-mmm, mmm-yy, h:mm AM/PM, h:mm:ss AM/PM, h:mm, h:mm:ss, m/d/yy h:mm.

setDateCellValue

- **void ExcelCell::setDateCellValue(Date date)**
  Set a Date as the cell value

setExcelStyle

- **void ExcelCell::setExcelStyle(String cellStyle)**
  Set the cell style for this cell

setFillBackgroundColor

- **void ExcelCellStyle::setFillBackgroundColor(String color)**
  Set the cell style fill Background color to that supplied. The valid colors are BROWN, OLIVE_GREEN,DARK_GREEN, DARK_TEAL,DARK_BLUE, INDIGO,GREY_80_PERCENT, DARK_RED,ORANGE, DARK_YELLOW,GREEN, TEAL, BLUE,BLUE_GREY, GREY_50_PERCENT,RED, LIGHT_ORANGE,LIME, SEA_GREEN,AQUA, LIGHT_BLUE,VIOLET, GREY_40_PERCENT,PINK, GOLD, YELLOW,BRIGHT_GREEN, TURQUOISE,SKY_BLUE, PLUM,GREY_25_PERCENT, ROSE,TAN, LIGHT_YELLOW,LIGHT_GREEN, LIGHT_TURQUOISE,PALE_BLUE, LAVENDER,WHITE, CORNFLOWER_BLUE,LEMON_CHIFFON, MAROON,ORCHID, CORAL,ROYAL_BLUE, LIGHT_CORNFLOWER_BLUE.

setFillForegroundColor

- **void ExcelCellStyle::setFillForegroundColor(String color)**
  Set the cell style fill foreground color to that supplied. The valid colors are BROWN, OLIVE_GREEN,DARK_GREEN, DARK_TEAL,DARK_BLUE,

INDIGO,GREY_80_PERCENT, DARK_RED,ORANGE, DARK_YELLOW,GREEN, TEAL, BLUE,BLUE_GREY, GREY_50_PERCENT,RED, LIGHT_ORANGE,LIME, SEA_GREEN,AQUA, LIGHT_BLUE,VIOLET, GREY_40_PERCENT,PINK, GOLD, YELLOW,BRIGHT_GREEN, TURQUOISE,SKY_BLUE, PLUM,GREY_25_PERCENT, ROSE,TAN, LIGHT_YELLOW,LIGHT_GREEN, LIGHT_TURQUOISE,PALE_BLUE, LAVENDER,WHITE, CORNFLOWER_BLUE,LEMON_CHIFFON, MAROON,ORCHID, CORAL,ROYAL_BLUE, LIGHT_CORNFLOWER_BLUE.

setFillPattern

- **void ExcelCellStyle::setFillPattern(String pattern)**
  Set the cell style fill pattern to that supplied. The valid patterns are NO_FILL, SOLID_FOREGROUND, FINE_DOTS, ALT_BARS, SPARSE_DOTS, THICK_HORZ_BANDS, THICK_VERT_BANDS, THICK_BACKWARD_DIAG, THICK_FORWARD_DIAG, BIG_SPOTS, BRICKS, THIN_HORZ_BANDS, THIN_VERT_BANDS, THIN_BACKWARD_DIAG, THIN_FORWARD_DIAG, SQUARES, DIAMONDS.

setFont

- **void ExcelCellStyle::setFont(ExcelCellFont font)**
  Set the cell style font to that supplied.

setFontHeight

- **void ExcelCellFont::setFontHeight(Integer height)**
  Set the cell font height.

setFontName

- **void ExcelCellFont::setFontName(String fontName)**
  Set the font name in the ExcelCellFont. The font name is accepted if it is a non-null String. The fonts names that are valid are those that are installed on the windows system that the spreadsheet is opened on.

setIndention

- **void ExcelCellStyle::setIndention(indent)**
  Set the cell indent. The indent amount is the number of intented characters.

setItalic

- **void ExcelCellFont::setItalic(Boolean flag)**
  Set the cell font to italic by passing true in, or non-italic by passing false in.

setLeftBorderColor

- **void ExcelCellStyle::setLeftBorderColor(String color)**
  Set the cell style left border color to that supplied. The valid colors are BROWN, OLIVE_GREEN,DARK_GREEN, DARK_TEAL,DARK_BLUE, INDIGO,GREY_80_PERCENT, DARK_RED,ORANGE, DARK_YELLOW,GREEN, TEAL, BLUE,BLUE_GREY, GREY_50_PERCENT,RED, LIGHT_ORANGE,LIME, SEA_GREEN,AQUA, LIGHT_BLUE,VIOLET, GREY_40_PERCENT,PINK, GOLD, YELLOW,BRIGHT_GREEN, TURQUOISE,SKY_BLUE, PLUM,GREY_25_PERCENT, ROSE,TAN, LIGHT_YELLOW,LIGHT_GREEN, LIGHT_TURQUOISE,PALE_BLUE, LAVENDER,WHITE, CORNFLOWER_BLUE,LEMON_CHIFFON, MAROON,ORCHID, CORAL,ROYAL_BLUE, LIGHT_CORNFLOWER_BLUE.

setNumericCellValue

- **void ExcelCell::setNumericCellValue(Double number)**
  Set a number as the cell value

setRightBorderColor

- **void ExcelCellStyle::setRightBorderColor(String color)**
  Set the cell style Right border color to that supplied. The valid colors are BROWN,

OLIVE_GREEN,DARK_GREEN, DARK_TEAL,DARK_BLUE, INDIGO,GREY_80_PERCENT, DARK_RED,ORANGE, DARK_YELLOW,GREEN, TEAL, BLUE,BLUE_GREY, GREY_50_PERCENT,RED, LIGHT_ORANGE,LIME, SEA_GREEN,AQUA, LIGHT_BLUE,VIOLET, GREY_40_PERCENT,PINK, GOLD, YELLOW,BRIGHT_GREEN, TURQUOISE,SKY_BLUE, PLUM,GREY_25_PERCENT, ROSE,TAN, LIGHT_YELLOW,LIGHT_GREEN, LIGHT_TURQUOISE,PALE_BLUE, LAVENDER,WHITE, CORNFLOWER_BLUE,LEMON_CHIFFON, MAROON,ORCHID, CORAL,ROYAL_BLUE, LIGHT_CORNFLOWER_BLUE.

setStrikeout

- **void ExcelCellFont::setStrikeout(Boolean flag)**
  Set the cell style text to strikeout by passing true in, or non-strikeout by passing false in.

setStringCellValue

- **void ExcelCell::setStringCellValue(String string)**
  Set a String as the cell value

setTopBorderColor

- **void ExcelCellStyle::setTopBorderColor(String color)**
  Set the cell style Top border color to that supplied. The valid colors are BROWN, OLIVE_GREEN,DARK_GREEN, DARK_TEAL,DARK_BLUE, INDIGO,GREY_80_PERCENT, DARK_RED,ORANGE, DARK_YELLOW,GREEN, TEAL, BLUE,BLUE_GREY, GREY_50_PERCENT,RED, LIGHT_ORANGE,LIME, SEA_GREEN,AQUA, LIGHT_BLUE,VIOLET, GREY_40_PERCENT,PINK, GOLD, YELLOW,BRIGHT_GREEN, TURQUOISE,SKY_BLUE, PLUM,GREY_25_PERCENT, ROSE,TAN, LIGHT_YELLOW,LIGHT_GREEN, LIGHT_TURQUOISE,PALE_BLUE, LAVENDER,WHITE, CORNFLOWER_BLUE,LEMON_CHIFFON, MAROON,ORCHID, CORAL,ROYAL_BLUE, LIGHT_CORNFLOWER_BLUE.

setUnderline

- **void ExcelCellFont::setUnderline(String underline)**
  Set the cell font underline. Valid Strings are U_NONE, U_SINGLE, U_DOUBLE, U_SINGLE_ACCOUNTING and U_DOUBLE_ACCOUNTING.

setVerticalAlignment

- **void ExcelCellStyle::setVerticalAlignment(String valignment)**
  Set the cell style vertical alignment to that supplied. The valid vertical alignments are VERTICAL_TOP, VERTICAL_CENTER, VERTICAL_BOTTOM, VERTICAL_JUSTIFY .

setWrapText

- **void ExcelCellStyle::setWrapText(Boolean flag)**
  Set the cell style text to wrapping by passing true in , or non-wrapping by passing false in.

libxml

getXMLNode

- **XmlNode XMLNode::getXmlNode(String nodePath)**
  Returns the xmlnode selected by sPath relative to this node

getXMLNodeName

- **String XMLNode::getXMLNodeName()**
  Returns the name of the current XMLNode.

getXMLNodePath
- **String XMLNode::getXMLNodePath()**
  Returns the path of the current XMLNode. This path is not an XPath - it is the concatenation of all the names of the parent XMLNode's path, /, and the name of this XMLNode

getXMLNodes
- **XmlNode[] XMLNode::getXmlNodes(String sPath)**
  Returns the xmlnodes selected by sPath relative to this node

getXMLNodeValue
- **String XMLNode::getXMLNodeValue(String nodePath [, Boolean bRequired])**
  Returns the value of the current XMLNode. Default value of bRequired is false. It is set to throw AustinException

getXMLNodeValues
- **String[] XMLNode::getXMLNodeValues(String nodePath [, Boolean bRequired])**
  Returns the values of the XMLNode given by path. Default value of bRequired is false. It is set to throw AustinException

new$XmlDocument
- **new XmlDocument(Doc doc/String str)**
  Creates an XmlDocument from a docstore Doc instance or an xml string literal. If the string starts with "file://" then the xml document will be loaded from the file system according to the specified path.

setXMLNodeValue
- **void XMLNode::setXMLNodeValue(String sPath, String value)**
  Sets the value of the xmlNode given by sPath. Creates the node if it doesn't exists.

setXMLNodeValues
- **void XMLNode::setXMLNodeValue(String sPath, String[] values)**
  Sets the value of the xmlNode given by sPath. Creates the node if it doesn't exists.

validateXML
- **String validateXML(String docPath)**
  Validates an XmlDocument from a docstore Doc instance. Returns "Success" if its a valid XML Document. Returns "Document not found" if the XML Document not found in DocStore. Returns "Document is empty" if the XML Document is empty. Returns "Fatal Parsing Error" concatenated with the error description for a non-XML Document. Returns "Error" for any other error.

xmlDocToString
- **String XMLNode::xmlDocToString()**
  Returns the serialized xml document which this node is part of

math

max
- **Number max(Number a, Number b)**
  Return the max

min
- **Number min(Number a, Number b)**
  Return the min

rand
- **Integer rand(Integer max)**

Returns a random integer that is between 0 and max

reformatDouble

- **String reformatDouble (Double origDouble, Integer minDigitsBeforeDecPoint, Integer maxDigitsAfterDecPoint)**
  Returns a new String representing the number, reformatted to fit the criteria specified by minDigitsBeforeDecPoint and maxDigitsAfterDecPoint

toDouble

- **Double toDouble(String str)**
  Parses str as a Double

toInteger

- **Integer toInteger(String str)**
  Parses str as an Integer

number

formatNumber

- **String Number::formatNumber(String numberFormat, Locale loc)**
  Use to format an Number to a human readable format according to the locale specified in the parameter.If locale is null,it will use the locale of user setting. If numberFormat is null, it will use the default format of the locale.

formatNumberByLocPrecision

- **String formatNumberByLocPrecision(Double number, Locale loc, Integer precision)**
  This operation returns a string format along with defined precision and locale

formatNumberByPrecision

- **String formatNumberByPrecision(Double number,Integer precision)**
  This operation returns a string format along with defined precision

parseDouble

- **Double parseDouble(String str, Locale loc)**
  Pass string to double value based on locale

parseNumber

- **Number parseNumber(String str, String numberFormat, Locale locale)**
  Use to parse a String to Number by numberFormat and locale. If locale is null, it will use the locale of user setting .If numberFormat is null, it will use the default format of the locale. The numberFormat string is a pattern whose format is identical to the number format used by Java

operations

addLdapAttribute

- **void LdapEntry::addLdapAttribute(LdapAttribute attribute)**
  Adds an LdapAttribute Object to this LdapEntry.

addLdapEntry

- **void LdapEntrySet::addLdapEntry(LdapEntry entry)**
  Create a new Ldap Entry Set

addLdapObjectclass

- **void LdapEntry::addLdapObjectclass(LdapOperation objectClass)**
  Adds an LdapObjectclass Object to this LdapEntry.

createLDIFFile
- **createLDIFFile(String docstore_filename, LdapEntrySet entrySet)**
  A static method that creates an LDIF formatted file based upon an input Ldap entry set. The filename is a docstore reference

getLdapAttributes
- **LdapAttributes[] LdapEntry::getLdapAttributes()**
  Retrieves the LdapAttribute Objects for this LdapEntry.

getLdapAttributeType
- **String LdapAttribute::getLdapAttributeType()**
  Retrieves the attribute type or name of an LdapAttribute Object

getLdapAttributeValue
- **Object LdapAttribute::getLdapAttributeValue()**
  Retrieves the attribute value of an LdapAttribute Object

getLdapDistinguishedName
- **LdapAttribute LdapEntry::getLdapDistinguishedName()**
  Retrieves the distinguished name for an LdapEntry as an LdapAttribute Objects

getLdapEntries
- **LdapEntry[] LdapEntrySet::getLdapEntries()**
  Retrieves the LdapEntry Objects.

getLdapObjectclass
- **String LdapObjectclass::getLdapObjectclass()**
  Retrieves the name of a LdapObjectclass Object

getLdapObjectclasses
- **LdapObjectclass[] LdapEntry::getLdapObjectclasses()**
  Retrieves the LdapObjectclass Objects for this LdapEntry.

getLdapOperation
- **String[] LdapEntry::getLdapOperationDetails()**
  Retrieves the LdapOperation details strings for this LdapEntry.

isBinary
- **Boolean LdapAttribute::isBinary()**
  Indicates if the attribute represents a binary value encoded as a BASE64 string.

isExternal
- **Boolean LdapAttribute::isExternal()**
  Indicates if the attribute is a reference to an external file. For example and jpeg image

new$LdapAttribute
- **new LdapAttribute(String attributeType, Object attributeValue [, Boolean isBinary, Boolean isExternal)**
  Create a new Ldap Attribute. Optional parameters: isBinary represents a BASE64 encoded binary representation, default false; isExternal represents an external file reference, default false

new$LdapEntry
- **new LdapEntry()**
  Create a new Ldap Entry

new$LdapEntrySet
- **new LdapEntrySet()**

Create a new Ldap Entry Set

new$LdapObjectclass

- **new LdapObjectclass(String objclass)**
  Create a new Ldap objectclass object

parseLDIFFile

- **LdapEntrySet parseLDIFFile(String filename)**
  A static method that reads an LDIF file and instantiates an Ldap entry set based on it. The filename is a system reference

setLdapDistinguishedName

- **void LdapEntry::setLdapDistinguishedName(LdapAttribute dn)**
  Sets the single distinguished name for an LdapEntry as an LdapAttribute Objects

setLdapOperation

- **void LdapEntry::setLdapOperation(LdapOperation operation)**
  Adds an LdapOperation object to an LdapEntry.

operations_admin

flushScriptCache

- **::flushScriptCache()**
  Flushes the script cache on the local JVM. While this is normally done automatically, this script operation is provided in case there are any techniques that would cause the scripts to update in docstore, without properly updating the cache. This method may also be used to test the caching behavior of scrpts.

logActionableMessage

- **Integer logActionableMessage(String type, String action, String comment, IMessage msg, String state)**
  Logs a message in the alerts console for a message "msg". the Actionable "type" is primary heading or category under which an actionable is classified. Actionable "action" is known as the actionable topic. The topic is essentially a more specific version of the actionable type, it can be Accept or Reject. Actionable "comment" is information about the actionable. Actionable "state" sets the priority level of this actionable, the level can be set to either "INF" for informational, "ACT" for actionable or "ERR" for any error. It returns a unique ID for the message logged

new$SystemDB

- **new SystemDB()**
  Returns an object that represents the status of the current database

reportAllTableIndexes

- **String SystemDB::reportAllTableIndexes()**
  Reports all the tables and their indexes

reportChangedIndexes

- **String SystemDB::reportChangedIndexes()**
  Reports the list of indexes that have not been updated

reportExtraIndexes

- **String SystemDB::reportExtraIndexes()**
  Reports the list of indexes that are extra in the current database that sould not be there

reportIndexStatistics

- **String SystemDB::reportIndexStatistics()**

Reports all the indexes and their current statistics and whether or not they need to be rebuilt. Warning: This script op should not be used on a live sytems; using this script operation during normal WPC operations will have a detrimental effect on performance.

reportMissingIndexes

- **String SystemDB::reportMissingIndexes()**
  Reports the list of indexes that are missing in the current database that sould be there

operations_assert

assertEquals

- **void ::assertEquals(Object expectedValue, Object actualValue, [String message])**
  Throws a ScriptAssertException when the two object references do not refer to the same 'simple' value. Therefore, note that a value of 1.0 (of type double) will NOT equal a value of 1 (of type Integer). Note that 'complex' objects e.g. Catalog, Hierarchy etc. are not supported currently.

assertFalse

- **void ::assertFalse(Boolean condition, [String message])**
  Throws a ScriptAssertException unless condition is false

assertNotNull

- **void ::assertNotNull(Object obj, [String message])**
  Throws a ScriptAssertException unless the value is NOT null

assertNull

- **void ::assertNull(Object obj, [String message])**
  Throws a ScriptAssertException unless the value is null

assertTrue

- **void ::assertTrue(Boolean condition, [String message])**
  Throws a ScriptAssertException unless condition is true

operations_attr_group

addAttributeToAttrGroup

- **void AttrGroup::addAttributeToAttrGroup(String attrPath [, boolean throwError])**
  Adds an attribute to the attribute collection.Irrespective of throwError flag, throws an exception if called on a default generated attribute collection. If throwError is true, throws an exception for invalid attrPath. By default, value of throwError is false

addLocalesToAttrGroup

- **void AttrGroup::addLocalesToAttrGroup(String localesCSVString)**
  Adds the locales to the Attribute Collection.Throws an exception if called on default generated attribute collection.

addLocalizedNodeToAttrGroup

- **void AttrGroup::addLocalizedNodeToAttrGroup(Node node)**
  Associates this localized node with this attribute collection. Throws an exception if called on default generated attribute collection.

addSpecToAttrGroup

- **void AttrGroup::addSpecToAttrGroup(Spec spec, [boolean bDynamic])**

Associates all the nodes of the spec with this attribute collection. If the bDynamic flag is set to true then any additional nodes added to the spec, after the spec has been associated to the Attribute Collection, will become part of the Attribute Collection. If the bDynamic flag is set to false then only the nodes that are part of the spec at this time only will be added to the Attribute Collection. By default, value for bDynamic is true. Throws an exception if called on default generated attribute collection.

deleteAttrGroup

- **void AttrGroup::deleteAttrGroup()**
  Deletes this attribute collection. Throws an exception if called on default generated attribute collection.

getAllAttrGroupsForAttribute

- **AttrGroup[] getAllAttrGroupsForAttribute(String attrPath)**
  Returns an array of AttrGroups where the attrPath is included. Return null if attrPath is not included in any Attribut Group.

getAllAttributePathsFromAttrGroup

- **String[] AttrGroup::getAllAttributePathsFromAttrGroup()**
  Returns all the attribute paths associated with this attribute collection

getAttrGroupByName

- **AttrGroup getAttrGroupByName(String name)**
  Returns the attribute collection with the given name. Otherwise it becomes null.

getAttrGroupName

- **String AttrGroup::getAttrGroupName()**
  Returns the name of this attribute collection

getAttrGroupType

- **String AttrGroup::getAttrGroupType()**
  Returns the type of this attribute collection. Type can only be GENERAL.

getLocalesOfAttrGroup

- **String AttrGroup::getLocalesOfAttrGroup()**
  Returns the locales to the Attribute Collection as a single String of comma-separated values.

new$AttrGroup

- **new AttrGroup(String name, String type, [String desc])**
  Returns a new attribute collection with the given name, type and description. Type can only be GENERAL.

removeAttributeFromAttrGroup

- **void AttrGroup::removeAttributeFromAttrGroup(String attrPath)**
  Removes the attribute from the attribute collection.Throws an exception if called on default generated attribute collection.

removeLocalesFromAttrGroup

- **void AttrGroup::removeLocalesFromAttrGroup(String localesCSVString)**
  Removes the locales from the Attribute Collection.Throws an exception if called on default generated attribute collection.

removeSpecFromAttrGroup

- **void AttrGroup::removeSpecFromAttrGroup(Spec spec)**
  Disassociates all the nodes of the spec from this attribute collection.Throws an exception if called on default generated attribute collection.

operations_catalog

addCtgItemToSelection

- **(deprecated) see addEntryToSelection**
  Add the item to the basic selection (does nothing for an advanced selection)

addEntryToSelection

- **void Selection::addEntryToSelection(Entry entry)**
  Add the entry to the basic selection - the entry can be an item or a hierarchy node (does nothing for advanced selection).

addSecondaryCategoryTree

- **void Catalog::addSecondaryCategoryTree(CategoryTree categoryTree)**
  Add category tree as a secondary category tree

associateCategoryCacheToItemSet

- **void ItemSet::associateCategoryCacheToItemSet(CategoryCache catCache)**
  Associates the CategoryCache to the ItemSet so that when items are fetched, the corresponding categories are also fetched in bulk

buildTestCatalogData

- **buildTestCatalogData(Spec fileSpec, String sDocStorePath, Integer nbRows [, Integer firstSku])**
  Create a document at sDocStorePath for the file specification fileSpec with nbRows of random data, with the primary key starting at firstSku

containsByPrimaryKey

- **boolean Catalog::containsByPrimaryKey(String sPrimaryKey) - boolean ItemSet::containsByPrimaryKey(String sPrimaryKey)**
  Returns true, if the catalog or item set contains an item with the primary key sPrimaryKey

defineLocationSpecificData

- **void Catalog::defineLocationSpecificData(CategoryTree ctr, Spec spc, AttrGroup[] inhAttrGrps)**
  Sets up location specific data for a catalog. CTR is the category tree that contains the locations. SPC is the spec of the locations. INHATTRBRPS is an array of attribute groups containing the inheritable attributes.

deleteCatalog

- **(deleteCatalog(Catalog ctg)**
  Delete the catalog ctg. WARNING: Transactional disruption will occur: This script operation will roll back any existing transaction, and will leave the database connection in auto-commit. This script operation should be used carefully; for example, it should not be called in a catchError block.

deleteSelection

- **boolean Selection::deleteSelection()**
  Delete the selection. Return true if the deletion occured, false if selection was in use.

disableContainerProcessingOptions

- **void Container::disableContainerProcessingOptions(String[] options)**
  Disable the specified processing options - possible values: PRE_SCRIPT, POST_SCRIPT, POST_SAVE_SCRIPT, ENTRY_BUILD_SCRIPT, VALUE_RULES, VALIDATION_RULES, DEFAULT_VALUES, DEFAULT_VALUE_RULES, SEQUENCES, TYPE_VALIDATION, MERGE_WITH_OLD_VERSION, MIN_MAX_OCCURANCE, MIN_MAX_LENGTH, POSSIBLE_VALUES, PATTERN_VALIDATION, COL_AREA_LOCKS_VALIDATION, LOCK_CATEGORIES_FOR_ITEM_SAVE, LOCKING, UNIQUE_VALIDATION, ALL

exportCatalog
- **void Catalog::exportCatalog(Spec mktplaceSpec, SpecMap specMap)**
  Use to syndicate a catalog using mktplaceSpec and specMap

forEachCtgItem
- **forEachCtgItem([String sCatalogName,], Item item) { statements }**
  Executes the statements for each item in the catalog called sCatalogName

forEachItemSetElement
- **forEachItemSetElement(ItemSet is, Object oItem) { statements }**
  Executes the statements for each (oItem) map in the ItemSet

getAttributeGroupsToProcess
- **AttrGroup[] Container::getAttributeGroupsToProcess()**
  Return the list of attribute collections if any have been specified to restrict processing an retrieval from the database. If a null is returned, it means that retrieval and processing is not being restricted and all attributes are being processed

getCatalogAccessControlGroupName
- **String Catalog::getCatalogAccessControlGroupName()**
  Returns the Access Control Group for this catalog.

getCatalogAttribute
- **String[] Catalog::getCatalogAttribute(String sAttribName)**
  Returns a list of values for the attribute sAttribName

getCatalogAttributes
- **HashMap Catalog::getCatalogAttributes()**
  Returns a HashMap mapping attributes to their respective values. The attributes returned are "SCRIPT_NAME", "PRE_SCRIPT_NAME", "POST_SAVE_SCRIPT_NAME", "ENTRY_BUILD_SCRIPT", "DISPLAY_ATTRIBUTE", "USER_DEFINED_CORE_ATTRIBUTE_GROUP" and "SCRIPT_RESTRICT_LOCALES".

getCatalogCategoryTrees
- **HashMap Catalog::getCatalogCategoryTrees()**
  Return an array with category trees of this catalog

getCatalogId
- **Integer Catalog::getCatalogId()**
  Returns the ID of this catalog.

getCatalogItemCountInVersion
- **Integer Catalog::getCatalogItemCountInVersion(Version version)**
  Returns the number of items in the specified version of this catalog

getCatalogNamesList
- **String[] getCatalogNamesList([String filterByPrivilege])**
  Return the list of names of available catalogs filtered by catalog privileges LIST (list catalog), VIEW_ITEMS (view items in catalog), MODIFY_ITEMS (modify items in catalog). By default the catalog names for the catalogs with LIST privilege access are returned.

getCatalogsByAttributeValue
- **Catalog[] getCatalogsByAttributeValue(String attribute_name, String value)**
  Returns all catalogs that have the provided value for the attribute

getCatalogSpec

- **Spec Catalog::getCatalogSpec([Boolean bGetImmutableSpec])**
  Returns the spec this catalog. If the optional boolean bGetImmutableSpec is set to true, an immutable spec is retrieved (you can not modify the spec, but it is faster to retrieve). By default you get a mutable spec.

getCatalogVersion

- **Version Catalog::getCatalogVersion()**
  Returns the version of this catalog.

getCatalogVersionSummary

- **Versions[] Catalog::getCatalogVersionSummary()**
  Return an array with versions of this catalog - most recent first

getCategorizedItemCountInVersion

- **Integer Catalog::getCategorizedItemCountInVersion(Version version, CategoryTree ctgtree)**
  Returns the number of items categorized in the specified category tree for the specified version of this catalog

getCheckedOutEntryColAreasByPrimaryKey

- **String[] Container::getCheckedOutEntryColAreasByPrimaryKey(String sPrimaryKey)**
  Return a list of collaboration area names in which the entry for the given primary key is checked out. Returns empty list if entry is not checked out.

getContainerId

- **Integer Container::getContainerId()**
  Returns the ID of this container.

getContainerLocalesForRole

- **String Container::getContainerLocalesForRole(Role role)**
  Gets the locales that are allowed for this container specifically for the particular role.

getContainerType

- **String Container::getContainerType()**
  Returns the type of the container. Types can be one of the following: CATALOG, CATEGORY_TREE

getCtgByName

- **Catalog getCtgByName([String name])**
  Returns the catalog object with the corresponding name. If no name is provided, return the default catalog (if defined).

getCtgCategorySpecs

- **HashMap Catalog::getCtgCategorySpecs()**
  Returns the category specs for this catalog

getCtgItemByAttributeValue

- **ItemSet Catalog::getCtgItemByAttributeValue(String sNodePath, String sValue)**
  Returns ItemSet of items from the catalog that have the provided value for the attribute. Use "" or null value for searching EMPTY values. An excpetion is throw if the attribute does not exist or it is not indexed.

getCtgItemByPrimaryKey

- **Item Catalog::getCtgItemByPrimaryKey(String sPrimaryKey)**
  Method deprecated. Use Container::getEntryByPrimaryKey. Returns the item from the catalog with the given primary key - this method cannot be used to retrieve newly created items that have not been saved yet.

getCtgItemIdByPrimaryKey
- **Integer Catalog::getCtgItemIdByPrimaryKey(String sPrimaryKey)**
  Returns an item id by primary key

getCtgName
- **String Catalog::getCtgName()**
  Returns the name of this catalog

getCtgSpec
- **Spec Catalog::getCtgSpec([Boolean bGetImmutableSpec])**
  Returns the spec this catalog. If the optional boolean bGetImmutableSpec is set to true, an immutable spec is retrieved (you can not modify the spec, but it is faster to retrieve). By default you get a mutable spec.

getDefaultCatalogName
- **(deprecated) String getDefaultCatalogName()**
  See getCtgByName(). Returns the name of the catalog being used for an aggregation/syndication.

getDestinationCatalog
- **Catalog getDestinationCatalog()**
  Returns the destination catalog for catalog to catalog exports.

getDynamicSelectionQueryString
- **String DynamicSelection::getDynamicSelectionQueryString()**
  Returns the query string for this dynamic selection

getEntrySetForPrimaryKeys
- **EntrySet Container::getEntrySetForPrimaryKeys(Array pkeys, Boolean bOptimize)**
  Returns an EntrySet of the entries in this container for the given primary keys - set bOptimize to true if you don't plan on changing the entries, the entry set is then optimized but these entries don't keep track of changed attributes

getEntrySetSize
- **Integer EntrySet::getEntrySetSize()**
  Returns the number of entries in an entry set

getExportItemsCount
- **Integer getExportItemsCount()**
  Returns the number of items being syndicated

getExportItemSets
- **ItemSet[] getExportItemSets()**
  Returns an array of ItemSets being syndicated

getHierarchyNodeSetForSelection
- **HierarchyNodeSet Selection::getHierarchyNodeSetForSelection()**
  Return the hierarchy nodes in that selection as a HierarchyNodeSet

getItemBySku
- **(deprecated) see getCtgItemByPrimaryKey**

getItemPrimaryKeysForCategory
- **String[] Category::getItemPrimaryKeysForCategory(Catalog ctg [, Boolean ordered])**
  Returns an array of Strings containing the primary keys of the items in this category. The option Boolean 'ordered' being set to true makes the operation return the ordered children of this category if the catalog is set up to use ordering

getItemSetForCatalog
- **ItemSet Catalog::getItemSetForCatalog()**
  Returns an ItemSet of the items in this catalog

getItemSetForCategory
- **ItemSet Category::getItemSetForCategory(Catalog ctg [, Boolean ordered])**
  Returns an ItemSet of the items in this category. The option Boolean 'ordered' being set to true makes the operation return the ordered children of this category if the catalog is set up to use ordering

getItemSetForPrimaryKeys
- **ItemSet Catalog::getItemSetForPrimaryKeys(Array pkeys, Boolean bOptimize)**
  Returns an ItemSet of the items in this catalog for the given primary keys - set bOptimize to true if you don't plan on changing the items, the item set is then optimized but these items don't keep track of changed attributes

getItemSetForSelection
- **ItemSet Selection::getItemSetForSelection()**
  Return the items in that selection as a ItemSet

getItemSetForUnassigned
- **ItemSet Catalog::getItemSetForUnassigned(CategoryTree ctr, boolean readOnly)**
  Returns an ItemSet of the items in this catalog which are not assigned to any of the categories of given category tree

getItemSetSize
- **Integer ItemSet::getItemSetSize()**
  Returns the number of items in an item set

getItemsInCategory
- **Item[] Catalog::getItemsInCategory(Category [, Boolean ordered])**
  Returns an array of the items in this category. The option Boolean 'ordered' being set to true makes the operation return the ordered children of this category if the catalog is set up to use ordering

getPrimaryCategoryTree
- **CategoryTree Catalog::getPrimaryCategoryTree()**
  Returns the primary category tree of this catalog

getSelectionAccessControlGroupName
- **String Selection::getSelectionAccessControlGroupName()**
  Returns the Access Control Group for this selection.

getSelectionByName
- **Selection getSelectionByName(String sName)**
  Return the selection called sName

getSelectionCatalog
- **Catalog Selection::getSelectionCatalog()**
  Returns the selection's catalog

getSelectionHierarchy
- **Hierarchy Selection::getSelectionHierarchy()**
  Returns the selection's hierarchy.

getSelectionHierarchyNodeCount
- **Integer Selection::getSelectionHierarchyNodeCount()**

Returns the number of hierarchy nodes in a selection - returns 0 for advanced selections.

getSelectionItemCount
- **Integer Selection::getSelectionItemCount()**
  Returns the number of items in a selection

getSelectionName
- **String Selection::getSelectionName()**
  Returns the name of this selection

getSelectionNamesList
- **String[] getSelectionNamesList(Catalog catalog)**
  Return the list of names of available selections for catalog

getSequenceByName
- **Sequence getSequenceByName(String name)**
  Gets the sequence object with the corresponding name where name is defined by the name of the catalog/category tree + "_" + "CTG" / "CATTREE" + "_" + the path of the node the sequence is defined for.

getSequenceCurrentValue
- **String Sequence::getSequenceCurrentValue()**
  Returns the current value of this sequence. This number is less than or equal to the last preallocated value stored in the DB. Note, because of sequence caching, this number will vary depending on the JVM. If the JVM has not called getNextValue, then this method will return the last allocated value in the JVM. If the sequence row does not exist in the DB, this method will reuturn -1.

getSequenceNextValue
- **String Sequence::getSequenceNextValue()**
  Returns the next value of this sequence

getSourceCatalog
- **Catalog getSourceCatalog()**
  Returns the source catalog for catalog to catalog exports.

getVersionDate
- **Date Version::getVersionDate()**
  Returns the date of this version

getVersionName
- **String Version::getVersionName()**
  Returns the name of this version

getVersionType
- **String Version::getVersionType()**
  Returns the type of this version

hasAccessToPrivilegeForEntry
- **boolean Entry::hasAccessToPrivilegeForEntry(String priv_name)**
  Checks if the entry has access to the given privilege.

hasCtgListPermission
- **Boolean Catalog::hasCtgListPermission()**
  Returns true if the current user has permission to list this catalog, false otherwise

hasCtrListPermission
- **Boolean CategoryTree::hasCtrListPermission()**

Returns true if the current user has permission to list this category tree, false otherwise

insertNewVersion

- **Version Container::insertNewVersion(String sName)**
  Add a version called sName on this container.

isEntryCheckedOutForPrimaryKey

- **Boolean Container::isEntryCheckedOutForPrimaryKey(String sPrimaryKey)**
  Returns true if the entry for the given primary key is checked out into a collaboration area otherwise it returns false.

isOrdered

- **Boolean Catalog::isOrdered()**
  Returns the value of catalog's Use Ordering attribute.

linkCatalog

- **void Catalog::linkCatalog(Catalog dstCatalog, INode srcAttribute[, INode dstAttribute])**
  Links catalog to another using source and destination attributes. The dstAttribute is optional

loadCatalog

- **void Catalog/CategoryTree::loadCatalog(String docStorePathForFileToLoad, Spec fileSpec, SpecMap specMap, String feedType [itm|icm|ctr])**
  (Deprecated) This script operation has been deprecated and should not be used, and is scheduled to be removed. Use createImport() and startAggregationByName() instead. Loads data from the specified File Spec and Spec Map, into the catalog or category tree upon which this operation is called. The feedType must be "itm" for item to catalog feeds, "ctr" for category to category tree feeds, and "icm" for item to category mapping feeds. WARNING: Transactional disruption will occur: This script operation will rollback any existing transaction, undoing prior operations, and will leave the database connection in autocommit mode. This script operation should be used with caution - it should not be called in a catchError block for example.

new$BasicSelection

- **new BasicSelection(Catalog catalog, String name)**
  Returns an empty basic selection (Selection) on catalog

new$Catalog

- **new Catalog(Spec catalogSpec, String name, CategoryTree categoryTree [,Hashmap optionalArgs])**
  Returns a new catalog with the given spec and name. Pass optional args in the map with these keys "displayAttribute" (path of node), "accessControlGroup" (pass the ACG object), "isLookupTable" (default is false--set to true to create a lookup table and the Default Lookup Table Hierarchy is used as the category tree). If the displayAttribute is not set, the pk attribute is used.

new$DynamicSelection

- **new DynamicSelection(String selectionName, String queryString)**
  Returns a dynamic selection named selectionName and corresponding to the query queryString

new$LookupTable

- **new LookupTable(Spec spec, String name)**
  Returns a new lookuptable with the given spec and name

removeLocationSpecificData

- **void Catalog::removeLocationSpecificData(CategoryTree ctr)**

Sets up location specific data for a catalog. CTR is the category tree that contains the locations.

resetContainerLocalesForRole

- **void Container::resetContainerLocalesForRole(Role role)**
  Deletes the locales that are allowed for this container specifically for the particular role. After this the default list of locales for this role will be applicable.

saveCatalog

- **void Catalog::saveCatalog()**
  Saves this catalog. This is used to save new attributes that have been set on the catalog.

saveSelection

- **void Selection::saveSelection()**
  Save the basic or advanced selection to the database

setAttributeGroupsToProcess

- **void Container::setAttributeGroupsToProcess(AttrGroup[] aAttrGroups)**
  Only retrieve attributes that belong to one of the attribute collections specified in the list aAttrGroups for the given container

setCatalogAccessControlGroupName

- **void Catalog::setCatalogAccessControlGroupName(String acgName)**
  Sets the Access Control Group to the given name for this catalog.

setContainerAttribute

- **void Container::setContainerAttribute(String sAttribName, String[] sAttribValues)**
  Sets the value for the container attribute with sAttribName to the array of values

setContainerLocalesForRole

- **void Container::setContainerLocalesForRole(Role role, String localesCSVString)**
  Sets the locales that are allowed for this container for the particular role

setContainerProperties

- **void Container::setContainerProperties(HashMap properties)**
  The properties specified in the PROPERTIES hashmap are set for the container in question. The hashmap keys can be one of "PRE_SCRIPT_NAME", "POST_SCRIPT_NAME", "POST_SAVE_SCRIPT_NAME", "ENTRY_BUILD_SCRIPT", "DISPLAY_ATTRIBUTE", "USER_DEFINED_CORE_ATTRIBUTE_GROUP", "SCRIPT_RESTRICT_LOCALES" or "SCRIPT_NAME"(Deprecated). The values are required to be string names. The value for "SCRIPT_RESTRICT_LOCALES" must be "true" or "false". Enforcement of locale restrictions on script operations is based on the value of "SCRIPT_RESTRICT_LOCALES". "SCRIPT_NAME" is now deprecated and "POST_SCRIPT_NAME" should be used in its place.

setDynamicSelectionQueryString

- **void DynamicSelection::setDynamicSelectionQueryString(String queryString)**
  Sets the query string for this dynamic selection

setItemSetFetchLinkedItems

- **void ItemSet::setItemSetFetchLinkedItems(Boolean b)**
  Sets the item set to fetch or not fetch master linked items

setItemSetFetchSize

- **void ItemSet::setItemSetFetchSize(Integer i)**

Sets the item set fetch size (i.e. the number of items gotten in bulk each time)

setOrdered

- **Boolean Catalog::setOrdered(Boolean bOrder)**
  Alters the catalog to allow ordering or not. Returns a flag on whether the update is successful or not.

setSelectionAccessControlGroupName

- **void Selection::setSelectionAccessControlGroupName(String acgName)**
  Sets the Access Control Group to the given name for this selection.

setSelectionHierarchy

- **void Selection::setSelectionHierarchy(Hierarchy hier)**
  Sets the selection's hierarchy. Only applicable to basic selections

setSelectionName

- **void Selection::setSelectionName(String name)**
  Returns the name of this selection

setSequenceValueForMigration

- **void setSequenceValueForMigration(String sequenceName, String objName, String objType, Integer newValue)**
  This operation is only there for migration of environments. Do not use for any other purpose.

sortItemSet

- **void ItemSet::sortItemSet()**
  Sorts the ItemSet for performance

operations_category

addCategoryTreeMapping

- **void CategoryTreeMap::addCategoryTreeMapping(Category cat1, Category cat2)**
  Add a map between the two categories cat1 and cat2

addChildCategory

- **Boolean Category::addChildCategory(Category childCategory)**
  Adds childCategory as a child of this category

addItemSecondarySpecToCategory

- **void Category::addItemSecondarySpecToCategory(String sSpecName, [Catalog[] ctgs])**
  Associates a secondary item spec to this Category - if ctgs are passed, only those catalogs are affected by the spec.

addSecondarySpecToCategory

- **void Category::addSecondarySpecToCategory(String sSpecName, [Boolean bAdd])**
  Associates a secondary spec defining this categories attrs. The optional parameters allows for the Spec to be associated with the category but does not build out the EntryNode structure, useful to improve performance on imports

buildCategory

- **Category CategoryTree::buildCategory(String path, [String delimiter], [String primaryKey])**
  Returns a new category object when given the complete path of the new category and the delimiter that separates the categories in the path. If the delimiter is not

specified, it defaults to '/' (except if a filespec is used during an import). If the primary key is not specified, then it should either be automatically set via a sequence or value rule, or it should be set after creation. If used in workflows and the category path already exists in the source category tree, the category will be checked out.

deleteCategory

- **Boolean Category::deleteCategory()**
  Deletes this category from it's category tree

deleteCategoryTree

- **ValidationError[] deleteCategoryTree(CategoryTree ctr)**
  Delete the category tree ctr. Returns Validation Error array if any validation errors occured. Null if successful. WARNING: Transactional disruption will occur: This script operation will roll back any existing transaction, and will leave the database connection in auto-commit. This script operation should be used carefully; for example, it should not be called in a catchError block.

forEachCategorySetElement

- **forEachCategorySetElement(CategorySet categorySet, Object oCategory) { statements }**
  Executes the statements for each (oCategory) in the categorySet

getCategoryAttrib

- **Object Category::getCategoryAttrib(String sAttribPath)**
  Returns the value of the attribute sAttribPath (spec_name/attribute_name) of this category

getCategoryByPath

- **Category CategoryTree::getCategoryByPath (String sNamePath, String sDelim [, boolean bLight, boolean bReadOnly])**
  Returns the category with a full name path equivalent to sNamePath. sNamePath is expected to be delimited by sDelim. sNamePath should not contain the name of the root category, since we are already restricted to a spcific category tree. If bLight is true, not all data for the category is retrieved. If bReadOnly is true, a read only copy of the category is retrieved - bReadOnly should be used in exports, for example

getCategoryByPathNoCfp

- **Category CategoryTree::getCategoryByPathNoCfp (String sNamePath, String sDelim [, boolean bLight, boolean bReadOnly])**
  Returns the category with a full name path equivalent to sNamePath. sNamePath is expected to be delimited by sDelim. sNamePath should not contain the name of the root category, since we are already restricted to a spcific category tree. If bLight is true, not all data for the category is retrieved. If bReadOnly is true, a read only copy of the category is retrieved - bReadOnly should be used in exports, for example

getCategoryCache

- **CategoryCache CategoryTree::getCategoryCache(Integer size, Boolean get_all_categories)**
  Returns a CategoryCache for this CategoryTree. The cache will be empty if get_all_categories is false and the size will be the given size, or 100, whichever is the greater. If get_all_categories is true then the cache will contain all the categories for the given category tree and the size arguments will be ignored. The size of the cache in the latter case will be the greater of the number of categories in the tree or 100

getCategoryChildren

- **Category[] Category::getCategoryChildren([Boolean ordered, Catalog catalog, Boolean restrictToSubtreeWithItems])**
  Returns the categories immediately below this category. The option Boolean 'ordered' being set to true makes the operation return the ordered children of this category if

the catalog (if not specified, the default catalog) is set up to use ordering. The option restrictToSubtreeWithItems being set to true only returns categories that have items in their sub-trees

getCategoryChildrenUsingCache

- **Category[] Category::getCategoryChildrenUsingCache (CategoryCache cat_cache)**
  Returns this category's children, possibly making use of the cache provided

getCategoryCode

- **String Category::getCategoryCode()**
  Returns the code of this category.

getCategoryHasChildren

- **Boolean Category::getCategoryHasChildren()**
  Returns true if the category has children.

getCategoryLevels

- **Integer[] Category::getCategoryLevels()**
  Returns the levels of this category in an array of Integers.

getCategoryOrganizations

- **Organization[] Category::getCategoryOrganizations()**
  Return the all organizations this category is mapped to,

getCategoryParent

- **Category Category::getCategoryParent ([CategoryCache cat_cache])**
  Returns this category's parent. If there are multiple parents, only the first one is returned.

getCategoryParents

- **Category[] Category::getCategoryParents ()**
  Returns the parent categories of this Category

getCategoryParentsUsingCache

- **Category[] Category::getCategoryParentsUsingCache (CategoryCache cat_cache)**
  Returns this category's parents, possibly making use of the cache provided

getCategorySet

- **CategorySet CategoryTree::getCategorySet([Boolean bReadonly])**
  Returns a CategorySet for this CategoryTree.

getCategorySetByAttributeValue

- **CategorySet CategoryTree::getCategorySetByAttributeValue(String attribPath, Object attribValue, [Boolean bReadOnly])**
  Returns a CategorySet with all categories in the category tree which have the given AttribPath and AttribValue. Use "" or null AttribValue for searching EMPTY values. An exception is thrown if the attribPath is for non indexed attribute.

getCategorySetByFullNamePath

- **CategorySet CategoryTree::getCategorySetByFullNamePath(String[] fullNamePaths, String delimiter)**
  Returns an CategorySet of the categories in the category tree from the given full name paths. Do not include the category tree name in the full name paths

getCategorySetByItemSecondarySpec

- **CategorySet CategoryTree::getCategorySetByItemSecondarySpec(String specName)**

Returns an CategorySet that is a subset of the categories of this tree having the specified spec in their item secondary spec list

getCategorySetByLevel

- **CategorySet CategoryTree::getCategorySetByLevel(Integer level, [Boolean bReadOnly])**
  Returns an CategorySet of the categories in the category tree at a particular level

getCategorySetByPrimaryKey

- **CategorySet CategoryTree::getCategorySetByPrimaryKey(String primaryKey, [Boolean bReadOnly])**
  Returns a CategorySet with the categories in the category tree which have match the primary key

getCategorySetByStandAloneSpec

- **CategorySet CategoryTree::getCategorySetByStandAloneSpec(String specName)**
  Returns an CategorySet that is a subset of the categories of this tree having the specified spec in their stand alone spec list

getCategorySetSize

- **Integer CategorySet::getCategorySetSize()**
  Returns the number of categories in a category set

getCategoryTree

- **CategoryTree Category::getCategoryTree()**
  Returns the category tree object this category belongs to. Use getCategoryTreeByName() to get the category tree being used for an aggregation/syndication.

getCategoryTreeByName

- **CategoryTree getCategoryTreeByName([String name])**
  Returns the category tree object with the corresponding name. If name is not provided, return the category tree being used for the aggregation/syndication.

getCategoryTreeMap

- **CategoryTreeMap getCategoryTreeMap(CategoryTree ctr1, CategoryTree ctr2)**
  Returns the category tree map between the two category trees ctr1 and ctr2

getCategoryTreeName

- **String CategoryTree::getCategoryTreeName()**
  Returns the name of this categoryTree.

getCategoryTreeNamesList

- **String[] getCategoryTreeNamesList([String filterByPrivilege])**
  Return the list of names of available category trees filtered by category tree privileges LIST (list category tree), VIEW_ITEMS (view items in category tree), MODIFY_CATEGORY_ATTRIBUTES (modify category attributes in category tree). By default the category tree names for the category tree with LIST privilege access are returned.

getCategoryTreeSpec

- **Spec CategoryTree::getCategoryTreeSpec()**
  Returns the spec of this category tree

getCurrentLocation

- **ICategory getCurrentLocation()**
  Returns the category that identifies the current location if the script is running the context of a location

getDefaultCategoryTreeName

- **(deprecated) String getDefaultCategoryTreeName()**
  See getCategoryTreeByName(). Returns the name of the category tree being used for an aggregation/syndication. Use getCategoryTreeByName() to get the category tree being used for the aggregation/syndication.

getDescendentCategorySetForCategory

- **CategorySet Category::getDescendentCategorySetForCategory([Boolean bReadonly])**
  Returns a CategorySet consisting of all the descendents of this category

getEntryPosition

- **Integer Category::getEntryPosition(Catalog ctg, Entry child)**
  Allows users to get the position of a child Entry within a parent category. This will only work in an ordered catalog. Returns the position (if it works) or null (if it fails).

getFullPaths

- **String[] Category::getFullPaths ([String sDelimiter], [boolean bWithRootName])**
  Returns the full name paths of this Category, using the sDelimiter as the delimiter if provided. The full path returned includes the root categories name if bWithRootName is true.

getHierarchyType

- **String Hierarchy::getHierarchyType()**
  Returns the type of the hierarchy. Types can be one of the following: CATEGORY_TREE, ORGANIZATION_TREE, COLLABORATION_AREA

getItemSecondarySpecsForCategory

- **Spec[] Category::getItemSecondarySpecsForCategory([Catalog ctg])**
  Returns the item secondary specs associated with this category

getMappedCategories

- **Category[] Category::getMappedCategories(CategoryTree ctr)**
  Returns the categories in ctr (if any) to which this category is mapped

getPathValue

- **String Category::getPathValue()**
  Returns the path attribute value of this category. Note, this is not the full path.

getSecondarySpecsForCategory

- **Spec[] Category::getSecondarySpecsForCategory()**
  Returns the secondary specs defining this categories attrs

mapCategoryToOrganizations

- **void Category::mapCategoryToOrganizations(Category[] categories [, boolean bAdd])**
  Maps the category to all the organizations provided. If bAdd is true, the old mappings are added to otherwise they are overwritten to be the new set of organizations

new$Category

- **new Category(CategoryTree ctr, String path, [String delimiter], [String primaryKey]**
  Returns a new category object when given the complete path of the new category and the delimiter that separates the categories in the path. If the delimiter is not specified, it defaults to '/' (except if a filespec is used during an import). If the primary key is not specified, then it should either be automatically set via a sequence or value rule, or it should be set after creation. If used in workflows and the category path already exists in the source category tree, the category will be checked out.

new$CategoryTree

- **new CategoryTree(Spec spec, String name [,HashMap optionalArgs]])**
  Returns a new category tree with the given primary spec and name. Pass optional args in the map with these keys "useInheritance" (default is false), "displayAttribute" (Node object), "pathAttribute" (Node object), "accessControlGroup" (pass the ACG object), "isOrganizationTree" (default is false--set to true to create an organization tree). If the pathAttribute is not set, the primary key will be used. If the displayAttribute is not set, the pathAttribute is used.

removeCategoryTreeMapping

- **void CategoryTreeMap::removeCategoryTreeMapping(Category cat1, Category cat2)**
  Remove a map between the two categories cat1 and cat2

removeChildCategory

- **void Category::removeChildCategory(String categoryName)**
  Remove childCategory from this categorie's children. Only allowed if childCategory has at least another parent.

removeItemSecondarySpecFromCategory

- **void Category::removeItemSecondarySpecFromCategory(String sSpecName))**
  Disassociates a secondary item spec to from this Category.

removeSecondarySpecFromCategory

- **void Category::removeSecondarySpecFromCategory(String sSpecName))**
  Disassociates a secondary spec defining this categories attrs.

reorderEntry

- **Integer Category::reorderEntry(Catalog ctg, Entry child, Integer position [, Boolean bInsertBefore])**
  Allows users to adjust the ordering of a child Entry within a parent category in catalog ctg. Entry child is moved before (bInsertBefore=true) or after (bInsertBefore=false) the position (zero is the first element) specified. Returns the ordered entry id (if it works) or null (if it fails). This method should not be used in conjunction with a transaction. The Boolean flag is optional and if not specified defaults to true.

saveCategoryTree

- **ValidationError[] CategoryTree::saveCategoryTree ()**
  Saves this category tree. DO NOT USE in AGGREGATION if you are in a item-to-category feed or a category tree feed. The category tree you are aggregating to gets saved automatically at the end of an aggregation. However, if you side affect another category tree, then call this operation to capture the changes you made. Returns Validation Error array if any validation errors occured. Null if successful

saveCategoryTreeMap

- **void CategoryTreeMap::saveCategoryTreeMap()**
  Save this category tree map

setCategoryAttrib

- **void Category::setCategoryAttrib(String sAttribPath, Object sValue)**
  Sets the attribute sAttribPath (spec_name/attribute_name) of this category to sValue

setCategoryCacheFetchSize

- **void CategoryCache::setCategoryCacheFetchSize(Integer i)**
  Sets the category cache fetch size (i.e. the number of categories gotten in bulk each time). This is only applicable if the category cache is associated with an ItemSet.

operations_col_area

addEntryIntoColArea

- **boolean CollaborationArea::addEntryIntoColArea(Entry entry, String stepPath)**
  Used for importing new items into a collaboration area. This script operation will post a message to add the newly constructed entry to the given stepPath of the collaboration area. Returns a boolean. If the entry exists in the collaboration area or the source catalog, false will be returned. True indicates the message was successfully posted. The import will occur after the current transaction is committed. This operation returns false if the entry is a category.

checkOutEntries

- **HashMap CollaborationArea::checkOutEntries(EntrySet entrySet, [String stepPath], [boolean waitForStatus)**
  Checks out the entries in the entrySet into the collaboration area. If stepPath is not specified the entries will be checked-out into the Initial step. If waitForStatus is true, the checkout will take place immediately and the statuses will be returned. Otherwise the checkout will not take place immediately, instead a message will be posted to perform the operation after the current transaction is committed. Returns a HashMap of entry primary key to the status of the checkout (or null if waitforStatus is false). Checkout status could be one of the following: CHECKOUT_SUCCESSFUL, CHECKOUT_FAILED, ALREADY_CHECKED_OUT, ENTRY_LOCKED and ATTRIBUTE_LOCKED. ATTRIBUTE_LOCKED indicates one or more attributes required for that collaboration area are checked out to another collaboration area. waitForStatus is false by default.

checkOutEntry

- **HashMap CollaborationArea::checkOutEntry(Entry entry, [String stepPath], [boolean waitForStatus)**
  Checks out the entry into the collaboration area. If stepPath is not specified the entry will be checked-out into the Initial step. If waitForStatus is true, the checkout will take place immediately and the status will be returned. Otherwise the checkout will not take place immediately, instead a message will be posted to perform the operation after the current transaction is committed. Returns a HashMap of entry primary key to the status of the checkout (or null if waitforStatus is false). Checkout status could be one of the following: CHECKOUT_SUCCESSFUL, CHECKOUT_FAILED, ALREADY_CHECKED_OUT, ENTRY_LOCKED and ATTRIBUTE_LOCKED. ATTRIBUTE_LOCKED indicates one or more attributes required for that collaboration area are checked out to another collaboration area. waitForStatus is false by default.

dropEntries

- **void CollaborationArea::dropEntries(EntrySet entrySet)**
  Posts a message to drop the entries in the entrySet from the collaboration area and to unlock the attributes which were locked in the source container for those entries. The drop will not take place until after the current transaction has committed.

dropEntry

- **void CollaborationArea::dropEntry(Entry entry)**
  Posts a message to drop the entry from the collaboration area and to unlock the attributes which were locked in the source container for that entry. The drop will not take place until after the current transaction has committed.

getColAreaAdminRoles

- **String[] CollaborationArea::getColAreaAdminRoles()**
  Returns the admin role names for the collaboration area.

getColAreaAdminUsers

- **String[] CollaborationArea::getColAreaAdminUsers()**
  Returns the admin user names for the collaboration area.

getColAreaByName
- **CollaborationArea getColAreaByName(String colAreaName)**
  Return a collaboration area object if exists otherwise null.

getColAreaContainer
- **Container CollaborationArea::getColAreaContainer()**
  Returns the collaboration area as a container.

getColAreaEntryHistory
- **ColAreaEntryHistory[] getColAreaEntryHistory(String colAreaName, String wflName, String primaryKey)**
  Return the entire history of the entry in the given collaboration area.

getColAreaHistoryByTimePeriod
- **ColAreaEntryHistory[] getColAreaHistoryByTimePeriod(String colAreaName, Date beginDate, Date endDate)**
  Return the entire history given collaboration area for the given time period.

getColAreaHistoryDate
- **Date ColAreaEntryHistory::getColAreaHistoryDate()**
  Returns the date for the given collaboration area history event.

getColAreaHistoryEntryKey
- **String ColAreaEntryHistory::getColAreaHistoryEntryKey()**
  Returns the entry key for the given collaboration area history event.

getColAreaHistoryEventAttribute
- **String ColAreaEntryHistory::getColAreaHistoryEventAttribute(String attrName)**
  Returns the attribute value for the given collaboration area history event type attribute name. attrName could be one of the following: COMMENT, EXIT_VALUE, ENTRY_DIFFERENCES

getColAreaHistoryEventType
- **String ColAreaEntryHistory::getColAreaHistoryEventType()**
  Returns the event type for the given collaboration area history event. Event types could be one of the following: CHECKOUT, CHECKIN, BEGINSTEP, ENDSTEP, SAVEENTRY, DROP, TIMEOUT.

getColAreaHistoryStepPath
- **String ColAreaEntryHistory::getColAreaHistoryStepPath()**
  Returns the step path for the given collaboration area history event.

getColAreaHistoryUser
- **String ColAreaEntryHistory::getColAreaHistoryUser()**
  Returns the username for the given collaboration area history event.

getColAreaHistoryWorkflow
- **String ColAreaEntryHistory::getColAreaHistoryWorkflow()**
  Returns the workflow name for the given collaboration area history event.

getColAreaId
- **int CollaborationArea::getColAreaId()**
  Returns the internal Id for the Collaboration Area.

getColAreaName
- **String CollaborationArea::getColAreaName()**
  Returns the name of the collaboration area.

getColAreaNames
- **String[] getColAreaNames()**
  Returns all of the Collaboration Area Names for the current Company

getColAreaNamesForRole
- **String[] getColAreaNamesForRole(String role_name)**
  Returns a list of collaboration area names that are applicable to a particular role.

getColAreaNamesForUser
- **String[] getColAreaNamesForUser()**
  Returns a list of collaboration area names that are applicable to the user.

getColAreaSrcContainer
- **Container CollaborationArea::getColAreaSrcContainer()**
  Returns the source container which this collaboration area is tied to.

getColAreaStepHistory
- **ColAreaEntryHistory[] getColAreaStepHistory(String colAreaName, String wflName, String stepPath)**
  Return the entire history of the step in the given collaboration area.

getColAreaWorkflow
- **Workflow CollaborationArea::getColAreaWorkflow()**
  Returns the workflow which this collaboration area is tied to.

getCountOfEntriesInColArea
- **int CollaborationArea::getCountOfEntriesInColArea()**
  Returns the entries currently in ALL the steps of the collaboration area.

getCountOfEntriesInColAreaStep
- **int CollaborationArea::getCountOfEntriesInColAreaStep(String stepPath)**
  Returns the entries currently in the given stepPath of the collaboration area.

getEntries
- **EntrySet CollaborationArea::getEntries()**
  Returns the entry set for the entries currently in the collaboration area.

getEntriesInfoXMLInStep
- **String CollaborationArea::getEntriesInfoXMLInStep(String stepName,HashMap itemSet, HashMap hmAttrPaths[, int entriesCount,String dateFormat])**
  Returns xml representation of workflow step entries for the given step name. The dateFormat is used for formating the date values. The attribute information of attributes present in the hmAttrPaths are included in the xml.

getEntriesInStep
- **EntrySet CollaborationArea::getEntriesInStep(String stepPath)**
  Returns the entry set for the entries currently in the step of the collaboration area. The format of the stepPath is Stepname

getEntryInStep
- **Entry CollaborationArea::getEntryInStep(String stepPath)**
  Returns one entry that is currently in the step of the collaboration area, if there is at least one. If there is more than one entry currently in the step, then it is undetermined which particular one will be returned by a call to this operation. The format of the stepPath is Stepname

getItemsInStepByAttribute
- **HashMap CollaborationArea::getItemsInStepByAttribute(String stepName,String searchAttributePath,String searchAttrValue,String**

sSortColumn[,boolean isAscending,int startIndex, int endIndex,String categoryPKList])
Returns a HashMap of items for the current pagination with the key ITEMSET and total no of matched items with the key ITEMCOUNT. By default isAscending is taken as true. startIndex as 0 and endIndex as total number of resultant items. categoryPKList is optional argument having comma separated list of category primary key for example :'1','3','7' . It is used for filtering items

getItemsInStepBySelection

- **HashMap CollaborationArea::getItemsInStepBySelection(String stepName,String selectionString,String sSortColumn[,boolean isAscending,int startIndex, int endIndex])**
  Returns a HashMap of items for the current pagination with the key ITEMSET and total no of matched items with the key ITEMCOUNT. By default isAscending is taken as true. startIndex as 0 and endIndex as total number of resultant items.

getItemSubset

- **ItemSet CollaborationArea::getSubset(IItemSet items, Integer start_point [, Integer end_point])**
  Returns an IItemSet which is a subset cloned from the supplied IItemSet restrained by start and optional end index positions. A start point of -1 is interpreted as 0. If the end index is omitted, all items are retrieved from the start point.

getReservedEntriesInStep

- **EntrySet CollaborationArea::getReservedEntriesInStep(String stepPath)**
  Returns the entry set for the reserved entries currently in the step of the collaboration area. The format of the stepPath is Stepname

getStepEntryTimeout

- **Date CollaborationArea::getStepEntryTimeout(Entry entry, String stepPath)**
  Expects the given entry to actually be in the given collaboration area's specified stepPath. If the entry's really there, the script op returns the moment in time when it will time out. If any of the assumptions are not met (collaboration area has no such stepPath, entry not in that stepPath, etc.), the operation simply does nothing, i.e. no Exception thrown. The operation doesn't modify the collaboration area's underlying workflow at all. It should be thought of as operating on an entry in a collaboration area, that is expected to be in a particular stepPath at the moment in time when the op is executed.

getStepsForEntry

- **String[] CollaborationArea::getStepsForEntry(Entry entry)**
  Returns all the steps that the entry is currently in for the given collaboration area. The return values is a string array containing the stepPaths. Entry should be retrieved using Collaboration Area as Container.

getUsernameForReservedEntryInStep

- **String CollaborationArea::getUsernameForReservedEntryInStep(Entry entry, String stepPath)**
  Returns the username of the user who locked the entry in a wfl step for a given collaboration area, otherwise it returns null.

getWflStepsForRole

- **String[][] CollaborationArea::getWflStepsForRole(String roleName)**
  Returns workflow step paths along with the number of entries in it for which the role has access.

getWflStepsForUser

- **String[][] CollaborationArea::getWflStepsForUser()**
  Returns workflow step paths along with the number of entries in it for which the user

has access.

getWflStepsXML

- **String CollaborationArea::getWflStepsXML([String roleName])**
  Returns xml representation of workflow steps accessible by the given role name. if role name is not provided the xml representation of workflow steps accessible by the current user is returned.

getWflStepsXMLByAttrValue

- **String CollaborationArea::getWflStepsXMLByAttrValue(String attrPath, String attrValue[, String roleName])**
  Returns xml representation of workflow steps accessible by the given role name. if role name is not provided the xml representation of workflow steps accessible by the current user is returned.

isColAreaLocked

- **Boolean CollaborationArea::isColAreaLocked()**
  Returns true if the collaboration area is locked, otherwise it returns false.

isEntryReserved

- **Boolean CollaborationArea::isEntryReserved(IEntry entry)**
  Returns true if the entry is reserved in a given collaboration area, otherwise it returns false.

isEntryReservedInStep

- **Boolean CollaborationArea::isEntryReservedInStep(IEntry entry, String stepPath)**
  Returns true if the entry is reserved in a wfl step for a given collaboration area, otherwise it returns false.

lockColArea

- **Boolean CollaborationArea::lockColArea()**
  Locks the Collaboration Area so that no more entries can be checked out into it. Returns true or false depending on whether the lock was successfully applied or not.

moveEntriesToColArea

- **boolean CollaborationArea::moveEntriesToColArea(EntrySet entrySet, String destColAreaName)**
  Applies to items only. Moves the entrySet of entries in the collaboration area to another collaboration area. destColAreaName specifies the name of the destination collaboration area, into whose Initial step the entries will be checked into. This operation is asynchronous which means a message is posted to complete the move at some time after the current transaction is committed. Returns a boolean depending on whether the message to move the entry was successfully posted. Returns false if any of the entries are categories.

moveEntriesToNextStep

- **HashMap CollaborationArea::moveEntriesToNextStep(EntrySet entrySet, String stepPath, String exitValue)**
  Posts a message to move the entries in the entrySet from the specified stepPath to the next step for the given exitValue. Returns a map of Entry primary key to String of validation errors. Only valid entries are moved to the next step. The move will not take place until after the current transaction has committed.

moveEntryToColArea

- **boolean CollaborationArea::moveEntryToColArea(Entry entry, String destColAreaName)**
  Applies to items only. Moves the entrySet of entries in the collaboration area to another collaboration area. destColAreaName specifies the name of the destination

collaboration area, into whose Initial step the entries will be checked into. This operation is asynchronous which means a message is posted to complete the move at some time after the current transaction is committed. Returns a boolean depending on whether the message to move the entry was successfully posted. Returns false if the entry was a category.

moveEntryToNextStep

- **HashMap CollaborationArea::moveEntryToNextStep(Entry entry, String stepPath, String exitValue)**
  Posts a message to move the entry from the specified stepPath to the next step for the given exitValue. Returns a map of Entry primary key to String of validation errors. Only valid entries are moved to the next step. The move will not take place until after the current transaction has committed.

new$CollaborationArea

- **new CollaborationArea(String colAreaName, Workflow wfl, Container srcContainer)**
  Create a new collaboration area with the given name, wfl and srcContainer

publishEntriesToSrcContainer

- **boolean CollaborationArea::publishEntriesToSrcContainer(EntrySet entrySet)**
  Posts a message to publish the current attribute values for each entry in the entrySet in the collaboration area back to the source catalog or category tree, leaving those entries which are able to be published out to the source container unchanged and undisturbed in the collaboration area. Entries which cannot, for whatever reason, be published out to the source container will be moved to the Fixit step. This is also known as an interim checkin. The publish will occur after the current transaction completes. Returns a boolean. True indicates that the entire entrySet was valid and a publish message was successfully posted.

releaseEntryInStep

- **Boolean CollaborationArea::releaseEntryInStep(Entry entry, String stepPath)**
  Returns true if the entry was unlocked in a wfl step for a given collaboration area, otherwise it returns false. Operation runs synchronously.

reserveEntryInStep

- **Boolean CollaborationArea::reserveEntryInStep(IEntry entry, String stepPath, [String username])**
  Returns true if the entry was reserved in a wfl step for a given collaboration area, otherwise it returns false. Operation runs synchronously.

runStepJob

- **void CollaborationArea::runStepJob(EntrySet entrySet, String stepPath, String jobName, String jobType, Date date [, String username])**
  entrySet is an arbitrary EntrySet whose elements are expected to be in the workflow step specified by stepPath. jobName is the name of an existing job. jobType is either "IMPORT" or "EXPORT". date specifies the time of a one-time run. The optional username specifies the User in whose name the job will be run. Default is the User who created the job. When the job finishes, if it completes ok, the "JOB_SUCCESSFUL" exit value will be set on the elements of entrySet and an EndStep event posted. If it completes with error, the "JOB_FAILED" exit value will be set on elements of entrySet and a EndStep event posted. No harm done if the elements of entrySet are not in stepPath (but are still in the Collaboration Area): the EndStepEvent posted when the job finishes presumably will be meaningless and harmless. If one or more of the elements of entrySet are no longer in the Collaboration Area: they will be excluded from the entrySet of the posted EndStepEvent. If no elements of entrySet are still in the CollaborationArea, no EndStepEvent will be posted.

saveColArea

- **void CollaborationArea::saveColArea()**
  Saves the collaboration area.

setColAreaAccessControlGroup
- **void CollaborationArea::setColAreaAccessControlGroup(String acgName)**
  Sets the Access Control Group to the given name for this collaboration area.

setColAreaAdminRoles
- **void CollaborationArea::setColAreaAdminRoles(String[] roles)**
  Sets the admin roles for the collaboration area.

setColAreaAdminUsers
- **void CollaborationArea::setColAreaAdminUsers(String[] users)**
  Sets the admin users for the collaboration area.

setStepEntryTimeout
- **void CollaborationArea::setStepEntryTimeout(Entry entry, String stepPath, Date date)**
  Expects the entry to actually be in the given collaboration area's specified stepPath. Provided the entry is found to actually be in the step, its timeout is set to be the moment in time specified by the date argument. If any of the assumptions are not met (collaboration area has no such stepPath, entry are not in that stepPath, etc.), the operation simply does nothing, i.e. no Exception thrown. The operation doesn't modify the collaboration area's underlying workflow at all. It should be thought of as operating on an entry in a collaboration area, that is expected to be in a particular stepPath at the moment in time when the op is executed.

unlockColArea
- **Boolean CollaborationArea::unlockColArea()**
  Unlocks the Collaboration Area so that entries can be checked out into it again. Returns true or false depending on whether the unlock was successful or not.

operations_ctgaccprv

getCtgAccessPrvByRole
- **CtgAccessPrv Container::getCtgAccessPrvByRole(String sRoleName)**
  Returns the catalog access privilege for the catalog and role. Returns catalog access privilege with full access if none was found.

getCtgAccessPrvPermission
- **String CtgAccessPrv::getCtgAccessPrvPermission(String attributeCollectionName)**
  Returns the permission [E-editable|V-viewable] for the given attribute collection in the current catalog access prv.

new$CtgAccessPrv
- **new CtgAccessPrv(Container container, String roleName)**
  Builds a new ctg access privilege object

saveCtgAccessPrv
- **Boolean CtgAccessPrv::saveCtgAccessPrv()**
  Saves the current catalog access prv to the database

setAccessPrv
- **Boolean CtgAccessPrv::setAccessPrv(String attrGroupName, String permission)**
  Returns an access privilege object with the new permissions set for the attrGroupName. Permission is [V|E|null], and in case the permission is NULL the path

is removed from the access Privilege. Returns TRUE if successful, FALSE if not

setCtgAccessPrv

- **CtgAccessPrv CtgAccessPrv::setCtgAccessPrv(String[] attrGroups, String[] permissions)**
  Returns a catalog access privilege object with the permissions set according to the attribute collections. Permissions are [V|E]

operations_ctgview

addCtgTab

- **void CtgView::addCtgTab(CtgTab tab)**
  Add container tab object to the catalog view. The tab is added to the end of the list of tabs already defined for the container ctg view.

deleteCtgView

- **void deleteCtgView(CtgView ctgView)**
  Delete the catalog view ctgView.

getCtgTabAttrGroupsList

- **String[] CtgTab::getCtgTabAttrGroupsList()**
  Returns a list of ordered attribute collections for the catalog view tab.

getCtgTabByName

- **CtgTab CtgView::getCtgTabByName(String name)**
  Returns the tab with the given name, or null if there is no such tab.

getCtgTabName

- **String CtgTab::getCtgTabName()**
  Returns the name of the tab.

getCtgTabRow

- **CtgTabRow[] CtgTab::getCtgTabRow()**
  get the set of rows in the current container tab object

getCtgTabs

- **CtgTab[] CtgView::getCtgTabs()**
  Returns an ordered array of container tab objects for the particular container view

getCtgViewAttrGroupsList

- **String[] CtgView::getCtgViewAttrGroupsList()**
  Returns list of ordered attribute collections for the catalog view.

getCtgViewAttribsList

- **String[] CtgView::getCtgViewAttribsList()**
  Returns list of ordered attribute paths for the catalog view.

getCtgViewByName

- **CtgView Container::getCtgViewByName([String viewName, String viewType])**
  Returns the view with the corresponding name. If no name is specified, returns the default view. Use '[System Default]' to refer to the default view. The viewType can be 'ITEM_LIST', 'ITEM_POPUP', 'ITEM_LOCATION, 'BULK_EDIT', 'ITEM_EDIT', 'CATEGORY_EDIT' or 'CATEGORY_BULK_EDIT'. By default ITEM_EDIT/CATEGORY_EDIT is used. If the view is not found, it returns null.

getCtgViewPermission

- **String CtgView::getCtgViewPermission(String attrGroupName)**
  Returns the permission [E-editable|V-viewable] for the attribute collection name in the

current view.

getCtgViewType
- **String CtgView::getCtgViewType()**
  Returns the type of the view in question as a string. The values can be 'ITEM_LIST', 'ITEM_POPUP', 'ITEM_LOCATION, 'BULK_EDIT', 'ITEM_EDIT', 'CATEGORY_EDIT', 'CATEGORY_BULK_EDIT', 'HIERARCHY_RICH_SEARCH', 'ITEM_RICH_SEARCH'

getCurrentCtgViewName
- **String getCurrentCtgViewName()**
  Returns name of current catalog view (only in Data Entry scripts). Returns an empty string outside of the Data Entry scripts.

getDefaultCtgViewName
- **String Catalog::getDefaultCtgViewName()**
  Returns name of default catalog view.

getDefaultCtrViewName
- **String CategoryTree::getDefaultCtrViewName()**
  Returns name of default category tree view.

getListOfCtgViewNames
- **String[] Catalog::getListOfViewNames()**
  Returns an array of view names available for this catalog. An entry with '[System Default]' is always included as the first entry.

getNewCtgTab
- **CtgTab CtgView::getNewCtgTab(String name, AttrGroup[] rows)**
  Builds a new container tab object with the given name and returns it. The tab needs to be added to the catalog view in order to save it

getTabRowPath
- **String CtgTabRow::getTabRowPath()**
  Returns the attribute path for this tab row

insertCtgTabAt
- **void CtgView::insertCtgTabAt(CtgTab tab, Integer index)**
  Insert container tab object to the catalog view at the index position(zero base). If index is invalid, tab is added to the end of the list

new$CtgTabRow
- **CtgTabRow CtgTabRow(String path)**
  Builds a new container tab row object for the node specified by the path.

new$CtgView
- **new CtgView(Container container, String name)**
  Builds a new Ctg View

removeCtgTabAt
- **void CtgView::removeCtgTabAt(Integer index)**
  Remove container tab object to the catalog view at the index position(zero base).

saveCtgTabs
- **void CtgView::saveCtgTabs()**
  save the container tab objects that were new/modified in the container view

saveCtgView
- **Boolean CtgView::saveCtgView()**
  Saves the current ctgview to the database

setCtgTabRow

- **void CtgTab::setCtgTabRow(AttrGroup[] rows)**
  Sets the current container tab object to the new set of rows

setCtgView

- **CtgView CtgView::setCtgView(String viewType, String[] attrGroupNames, String[] permissions)**
  Sets the container view object with the given name/catalog and returns it. The viewType can be 'ITEM_LIST', 'ITEM_POPUP', 'ITEM_LOCATION, 'BULK_EDIT' or 'ITEM_EDIT'. By default ITEM_EDIT is used. Permissions are [V|E]

setDefaultCtgView

- **void Catalog::setDefaultCtgView(CtgView ctgView)**
  Sets the ctgview as the default catalog view.

setDefaultCtrView

- **void CategoryTree::setDefaultCtrView(CtgView ctrView)**
  Sets the ctrview as the default category tree view.

operations_distribution

createDataSource

- **String createDataSource(String name, String type, [HashMap extraAttribs])**
  Creates a Data Source of the type ("PULL_FTP", "PULL_FTP", "PUSH_WWW", "DOC_STORE") with given name. Will not create if a source with same name already exists. extraAttribs can be used to set other attributes of the datasource like "SERVER_ADDRESS", "SERVER_PORT", "USERNAME", "PASSWORD", "FILENAME", "DIRECTORY", "DOC_STORE_PATH"

getDistributionByName

- **Distribution getDistributionByName(String name)**
  Gets the distribution with the specified name if one exists, otherwise returns null

getFtp

- **HashMap/Boolean getFtp(String hostname, String port, String userid, String password, String path, String filename, String sDocStorePath, Boolean deleteRemoteFile, [Boolean detailedTransferStatus, Boolean loggingEnabled])**
  Use to get a file via FTP. The seventh parameter set where Trigo will store the retrieved file. The eighth and the ninth paramters together are optional. The eigth parameter gets the FTP Operation Status and the ninth paramter ensures that the FTP operations are logged. Returns the result as true/false if the eighth and the ninth are not specified otherwise a HashMap is returned. If a true/false is returned, it indicates if the ftp was a success/failure. If the size of the retrieved file is not the same as the size of the remote file the status is set to false. If a HashMap is returned, the first paramater is the true/false which indicates success/failure, the second paramater is the message string of the FTP Operation Status and the third parameter is the FTP Operation error code

getFullHTTPResponse

- **HashMap getFullHTTPResponse(String url, HashMap hmRequestProperites, HashMap hmParameters, String sRequestMethod , [String sEncoding, Doc doc, String sContentType, boolean bGetResponseReader, boolean bPostUserInfo, String sDocStorePath])**
  Returns a HashMap (with RESPONSE_READER and RESPONSE_HEADER_FIELDS) for the response for posting hmParameters or a doc of sContentType against the server at url, Use hmRequestProperties to send specific header information. An optional parameter bGetReader could be used to

specify if the function needs to also return the response reader (default is true). An optional parameter bPostUserInfo could be used to specify if the function would need to post the invoking user information (default is false). The response is optionally stored into a document at sDocStorePath in the docstore.

getHTTPResponse

- **BufferedReader getHTTPResponse(String url, HashMap hmRequestProperites, HashMap hmParameters, String sRequestMethod [,String sEncoding])**
  Returns a reader for the response for posting hmParameters against the server at url, Use hmRequestProperties to send specific header information

new$Distribution

- **new Distribution(String name, String type, [HashMap extraAttribs])**
  Creates a distribution with the provided name. The type can be: "FTP", "EMAIL", "POST", "ARIBA_CATALOG_UPLOAD", "INCOMING_FTP", "CUSTOM". extraAttribs should contain the parameters required for the different types which include "email", "hostname", "userid", "password", "path", "from", "to", "sub", "localpath"

saveMultipartRequestData

- **Doc[] saveMultipartRequestData(String saveDir, [String charset])**
  Saves the documents sent through multipart post in the docstore at location: /archives/uploaded/multipart/saveDir/. If a charset is given, that is used. Otherwise, the default_charset_value as specified in common.properties is used.

sendEmail

- **void sendEmail(String emailTos, String emailSubject, String emailBody, [Doc emailAttachment | Doc[] emailAttatchments])**
  Use to send an asynchronous email. The tos parameter is a list of email addresses seperated with the semicolon character (;). The 4th parameter could be an Doc or an array of Doc

sendFtp

- **void sendFtp(String hostname, String port, String userid, String password, String path, Doc doc | Doc[] docs)**
  Use to send a file via FTP. The 6th parameter could be an Doc or an array of Doc.

sendHTMLEmail

- **void sendHTMLEmail(String emailTos, String emailSubject, String emailBody, [Doc emailAttachment | Doc[] emailAttatchments])**
  Same as sendEmail operation, however will allow HTML anchor tags in the message body. Use to send an asynchronous email. The tos parameter is a list of email addresses seperated with the semicolon character (;). The 4th parameter could be an IDoc or an array of IDoc

sendHttp

- **void sendHttp(String url, [Doc doc, [String contentType]])**
  Use to do GET or POST to a URL.

sendHttpString

- **void sendHttpString(String url, String content, String contentType)**
  Use to do GET or POST to a URL.

sendMultipartPost

- **HashMap sendMultipartPost(String url, HashMap hmRequestProperites, HashMap hmParameters, String sRequestMethod, HashMap hmDocs)**
  Sends one or more documents of any content type and/or a set of hmParameters using multipart post against the server at url, Use hmRequestProperties to send specific header information. The hmDocs is the list of pairs ['document path', 'document content type'] for the documents of a particular content type (eg: text/plain,

image/gif etc.). Also returns a HashMap (with RESPONSE_READER and RESPONSE_HEADER_FIELDS) for the response.

setHttpServletResponseHeader

- **void setHttpServletResponseHeader(HashMap hmNameValuePairs)**
  Sets the name, value pairs specified in the hashMap into the header for the current HttpServletResponse

setHttpServletResponseStatus

- **void setHttpServletResponseStatus(String statusCode)**
  Sets the status (takes one of the values from the set #[#'SC_ACCEPTED', 'SC_OK', 'SC_CONTINUE', 'SC_PARTIAL_CONTENT', 'SC_CREATED', 'SC_SWITCHING_PROTOCOLS', 'SC_NO_CONTENT'#]#) for the current HttpServletResponse

operations_docstore

copyDoc

- **Doc Doc::copyDoc(sPath)**
  Copy this document to the specified sPath in the docstore. If the path ends with a '/' it is assumed that the doc needs to be copied to the specified directory with its current name

deleteDoc

- **void Doc::deleteDoc()**
  Delete this document from the docstore

forEachDocument

- **forEachDocument([Doc[] docs_list,], Doc doc) { statements }**
  Executes the statements for each document (used in distribution scripts). If the optional docs_list parameter is provided, however, the statements are executed for each element of docs_list

getDocAttribute

- **String Doc::getDocAttribute(String sAttributeName)**
  Return the attribute sAttributeName from this document

getDocAttributes

- **HashMap Doc::getDocAttributes()**
  Return the attributes of this document

getDocByPath

- **Doc getDocByPath(String sPath)**
  Return the document with path sPath

getDocContentAsString

- **String Doc::getDocContentAsString()**
  Return the content of this document as a string. WARNING - this means that the entire content of the document, however big, will be returned in a string so the uer needs to make sure that any call of this operation is not going to be used in a situation where the content of the document is too big (too big being defined by the amount of memory available to the process this operation is running in).

getDocLastModifiedTimeStamp

- **Date Doc::getDocLastModifiedTimeStamp()**
  Returns the date/time this document was last modified

getDocLength

- **Integer Doc::getDocLength([Boolean bBytes])**
  Returns the length of the document in kilo bytes. If bBytes is true, value is returned in bytes instead of Kb. Important when smaller files are concerned

getDocListByPaths
- **Doc[] getDocListByPaths(String[] sPaths)**
  Return the document at each path specified in sPaths

getDocPath
- **String Doc::getDocPath()**
  Return this document path

getDocStoreDirectoriesInDirectory
- **String[] getDocStoreDirectoriesInDirectory(String sPath)**
  Return the list of paths of directories under the directory sPath

getDocStoreFilesInDirectory
- **String[] getDocStoreFilesInDirectory(String sPath)**
  Return the list of paths of documents under the directory sPath

getDocStoreSubtreeList
- **String[] getDocStoreSubtreeList(String sPath)**
  Return the list of documents under sPath

getHrefForDocPath
- **String getHrefForDocPath(String sDocPath)**
  Return a absolute path for the document with path sDocPath. This can be used in an HTML reference to provide a link to the document.

moveDoc
- **Doc Doc::moveDoc(sPath)**
  Move this document to the specified sPath in the docstore. If the path ends with a '/' it is assumed that the doc needs to be moved to the specified directory with the same doc name as the source

setDocAttribute
- **void Doc::setDocAttribute(String sAttributeName, String sAttributeValue)**
  Set the attribute sAttributeName to sAttributeValue for this document

operations_entry

deleteEntryNode
- **void EntryNode::deleteEntryNode()**
  Remove this entrynode from the Entry. This operation will only work on Mutli-Occurance attributes, an error will be thrown if used on a non-mulit-occurance entryNode.

displayEntryAttrib
- **String Entry::displayEntryAttrib(String sAttribPath)**
  Returns the html string for displaying entry attribute specified by attribute path

forEachEntrySetElement
- **forEachEntrySetElement(EntrySet entrySet, Object oEntry) { statements }**
  Executes the statements for each (oEntry) in the entrySet

getAddedAttributePathsNewEntry
- **String[] EntryChangedData::getAddedAttributePathsNewEntry([Category location])**

Returns the paths of all attributes in LOCATION that (1) are not present in the old entry and (2) are present in the new entry from which this EntryChangedData object was created. If LOCATION is not specified or is null, then the comparison is done for global attributes.

getCheckedOutEntryColAreas

- **String[] Entry::getCheckedOutEntryColAreas()**
  Return a list of collaboration area names in which the entry is checked out. Returns empty list if entry is not checked out.

getCtgItemLocationAttribsForKeys

- **String Item::getCtgItemLocationAttribsForKeys(Category location, Object[] aAttribPath [, String sDelimiter])**
  Gets the attributes for an item based upon the passed location category and a Object[] (declared: var aAttribs = [];) of attribute keys (paths). The resultant values are loaded into the value pair of the mapping. If the value for one key is unset, it defaults to a blank string. If the key does not correspond with an attribute, a null is entered instead.

getDeletedAttributePathsOldEntry

- **String[] EntryChangedData::getDeletedAttributePathsOldEntry([Category location])**
  Returns the paths of all attributes in LOCATION that (1) are not present in the new entry and (2) are present in the old entry from which this EntryChangedData object was created. If LOCATION is not specified or is null, then the comparison is done for global attributes.

getDestinationEntrySetForRelatedEntries

- **EntrySet Entry::getDestinationEntrySetForRelatedEntries(Container filterContainer)**
  Returns EntrySet with all entries this entry is related to filtering by container if filterContainer is provided.

getDisplayValue

- **String Entry::getDisplayValue(Locale locale)**
  Returns the display value for the entry. If no display value is available then the primary key value is returned.

getEntry

- **Entry EntryNode::getEntry()**
  Returns the Entry behind the EntryNode.

getEntryAttrib

- **Object Entry::getEntryAttrib(String sAttribPath)**
  Returns the value of the attribute sAttribPath (spec_name/attribute_name) of this entry

getEntryAttribs

- **(deprecated)HashMap Entry::getEntryAttribs()**
  Returns an HashMap mapping the paths (spec_name/attribute_name) of attributes to their respective values

getEntryAttribsList

- **String[] Entry::getEntryAttribsList()**
  Returns an array of String containing the paths (spec_name/attribute_name) of all the attributes of this entry

getEntryAttribValues

- **Object[] Entry::getEntryAttribValues(String sAttribPath)**

Returns the values of the multi-value attribute given by sAttribPath (spec_name/attribute_name) of this entry

getEntryByPrimaryKey

- **Object Container::getEntryByPrimaryKey(String primaryKey)**
  Gets the entry given the primary key.

getEntryChangedData

- **EntryChangedData ::getEntryChangedData(Entry oldEntry, Entry newEntry)**
  Return an EntryChangedData object encapsulating the changes in data and locations between two entries at a point in time at which the EntryChangedData object is created,such that the returned object is a static object. Note, this script operation is very CPU intensive on large items (many locations and many attributes). Please see script operation getEntryChangedDataSinceLastSave.

getEntryChangedDataSinceLastSave

- **EntryChangedData Entry::getEntryChangedDataSinceLastSave()**
  Return an EntryChangedData object encapsulating the changes in data and locations between this entry and the value since the last save (including save as draft). Note, this script operation should be much faster than 'getEntryChangedData(oldEntry, newEntry).

getEntryContainer

- **Object Entry::getEntryContainer()**
  Gets the holding container for this Entry. Could be a catalog or category tree. Use isEntryAnItem to determine which one.

getEntryId

- **Integer Entry::getEntryId()**
  Returns this entry's id

getEntryNode

- **EntryNode EntryNode::getEntryNode(String sPath)**
  Return the entryNode with path sPath relative to EntryNode. If the path is not already built a NULL will be returned. Use the Entry::setEntryAttrib script operation to create a path that might not exist.

getEntryNodeChildren

- **EntryNode[] EntryNode::getEntryNodeChildren()**
  Return the children of this EntryNode

getEntryNodeExactPath

- **String EntryNode::getEntryNodeExactPath()**
  Returns the exact path of this entry node - the following is always true: rootNode.getEntryNode(entryNode.getEntryNodeExactPath()) == entryNode

getEntryNodeInheritedValue

- **Object EntryNode::getEntryNodeInheritedValue()**
  If this EntryNode inherits its value, return the value. Otherwise, return null.

getEntryNodeParent

- **EntryNode EntryNode::getEntryNodeParent()**
  Return the parent of this EntryNode. If it is the root node, NULL is returned.

getEntryNodePath

- **String EntryNode::getEntryNodePath()**
  Returns the Spec Node path of this entry node, NOT the relative path of this attr.

getEntryNodes

- **EntryNode[] EntryNode::getEntryNodes(String sPath)**
  Return the entry nodes matching the path sPath

getEntryNodeType

- **String EntryNode::getEntryNodeType()**
  return "V" = value, "G" = Grouping or top level of spec directory, "M" = Multi-directory (contains multiple occurrances of values or groupings))

getEntryNodeValue

- **Object EntryNode::getEntryNodeValue()**
  Return the value of this EntryNode

getEntryRelatedItemInfo

- **String[] Entry::getEntryRelatedItemInfo(int iItemId)**
  (Deprecated, see String[] Entry::getEntryRelationshipAttrib(String sEntryAttrib)) Returns an array of length 2 containing: [0]=Related Item's Catalog's Name, [1]=Related Item's Primary Key, for the related item represented by the given internal unique item id, at the browsing version of the catalog of the given entry

getEntryRelationshipAttrib

- **String[] Entry::getEntryRelationshipAttrib(String sAttribPath)**
  Given a relationship attribute path, returns an array of length 2 containing: [0]=Related Item's Catalog's Name, [1]=Related Item's Primary Key, for the related item. Exception will be thrown if attribute sAttribPath doesn't exist or it's not of relationship type

getEntrySaveResult

- **String Entry::getEntrySaveResult()**
  Returns the result of the last save called on this entry. Returns one of the following strings {ADDED,DELETED,MODIFIED,UNKNOWN}

getEntryStatus

- **String Entry::getEntryStatus()**
  Returns the status of the entry

getEntryXMLRepresentation

- **String Entry::getEntryXMLRepresentation(Spec spec, boolean includePrimaryKeyValue[, boolean addNameSpace, String dateFormat, Locale[] aLocales, AttrGroup[] aAttrGroups])**
  Returns the XML representation of this entry specific to the given spec which can be consumed by WPC-WPS integration WPS portion's XML parser. The optional parameter; addNameSpace allows the user to specify that the XML returned is in a format that can immediately be read in using the setEntryAttributesFromXMLRepresentation function. The default of false for this parameter allows user to prepend and append to the XML as required. The date format is the pattern by which dates should be converted. A Simple Date Format is normally used. If aLocales is specified, restrict to those locales. If AttrGroup[] is specified only return attribute belonging to one of the AttrGroup

getFlatEntryNodes

- **EntryNode[] Entry::getFlatEntryNodes([Boolean skipEmptyGrouping])**
  Returns an array of flat EntryNodes of this entry.

getFlatEntryNodesOf

- **EntryNode[] getFlatEntryNodesOf(EntryNode en)**
  Returns an array of all the entrynodes under this entrynode in depth first order

getFlatPrimaryEntryNodes

- **EntryNode[] Entry::getFlatPrimaryEntryNodes([Boolean skipEmptyGrouping])**

Returns an array of flat primary EntryNodes of this entry

getFlatSecondaryEntryNodes

- **EntryNode[] Entry::getFlatSecondaryEntryNodes([Boolean skipEmptyGrouping])**
  Returns an array of flat secondary EntryNodes of this entry

getItemLocationAttrib

- **Object Item::getItemLocationAttrib(Category location, String sAttribPath)**
  Gets a location attribute for an item. Note -- if you specify an occurence (grouping or value) that does not exist, then an exception is thrown.

getItemUsingEntryRelationshipAttrib

- **Item Entry::getItemUsingEntryRelationshipAttrib(String sAttribPath)**
  return the related item object for given relationship attribute path. Exception will be thrown if attribute sAttribPath doesn't exist or it's not of relationship type

getLocation

- **Category Object::getLocation()**
  If the caller object is the entry node, return the location (category) or null if a global entry node; if the caller object is a SearchResultSet, return the value of the designated column in the current row of this SearchResultSet object as a Location (Category).

getLocationPathForInheritedValue

- **String EntryNode::getLocationPathForInheritedValue(String delimiter)**
  Returns the path for the location from which this EntryNode inherits, or null if it does not inherit its value. The given delimiter is used.

getLocationsAddedAvailability

- **CategorySet EntryChangedData::getLocationsAddedAvailability(CategoryTree locationHierarchy)**
  Returns all locations in LOCATIONHIERARCHY that are available in the new entry but not the old entry from which this EntryChangedData object was created.

getLocationsChangedToHaveData

- **CategorySet EntryChangedData::getLocationsChangedToHaveData(CategoryTree locationHierarchy)**
  Returns all locations in LOCATIONHIERARCHY that (1) are available in both the old entry and the new entry from which this EntryChangedData object was created, and (2) contain no data in the old entry but do contain data in the new entry. Note, Override is considered as having data, and Inherit is considered as not having data.

getLocationsChangedToHaveNoData

- **CategorySet EntryChangedData::getLocationsChangedToHaveNoData(CategoryTree locationHierarchy)**
  Returns all locations in LOCATIONHIERARCHY that (1) are available in both the old entry and the new entry from which this EntryChangedData object was created, and (2) contain no data in the new entry but do contain data in the old entry. Note, Override is considered as having data, and Inherit is considered as not having data.

getLocationsHavingChangedData

- **CategorySet EntryChangedData::getLocationsHavingChangedData(CategoryTree locationHierarchy)**
  Returns all locations in LOCATIONHIERARCHY that (1) are available in both the old entry and the new entry from which this EntryChangedData object was created, and

(2) have at least one attribute path for which the old and new entries contain different values. Note that this operation will return a superset of all locations returned by the getLocationsChangedToHaveData and getLocationsChangedToHaveNoData script operations.

getLocationsRemovedAvailability

- **CategorySet EntryChangedData::getLocationsRemovedAvailability(CategoryTree locationHierarchy)**
  Returns all locations in LOCATIONHIERARCHY that are available in the old entry but not the new entry from which this EntryChangedData object was created.

getModifiedAttributePathsNewEntry

- **String[] EntryChangedData::getModifiedAttributePathsNewEntry([Category location])**
  Returns the paths of all attributes in LOCATION that (1) are present in both the old entry and the new entry from which this EntryChangedData object was created, and (2) contain different data in the old and new entries. It is possible for an attribute to have different attribute paths across the old entry and the new entry, for example because a multioccurrence sibling has been deleted. In this case, we return the attribute path for the new entry. If LOCATION is not specified or is null, then the comparison is done for global attributes.

getModifiedAttributePathsOldEntry

- **String[] EntryChangedData::getModifiedAttributePathsOldEntry([Category location])**
  Returns the paths of all attributes in LOCATION that (1) are present in both the old entry and the new entry from which this EntryChangedData object was created, and (2) contain different data in the old and new entries. It is possible for the same attribute to have different attribute paths across the old entry and the new entry, for example because a multioccurrence sibling has been deleted. In this case, we return the attribute path for the old entry. If LOCATION is not specified or is null, then the comparison is done for global attributes.

getNodeFromEntryNode

- **Node EntryNode::getNodeFromEntryNode()**
  Returns the Node object for this entry node.

getOriginalEntry

- **Entry Entry::getOriginalEntry()**
  Returns the original picture of the entry as stored in the database. If the entry is new or deleted, this operation returns null.

getPipeDelimitedCSVRepresentation

- **String Entry::getPipeDelimitedCSVRepresentation()**
  Returns a CSV representation of this entry with fields that are name value pairs separated by the pipe character. All attribute values have the exact path of the attribute with occurrence numbers as the name. All category paths have CATEGORY or PATH as the name for items and categories respectively.

getPossibleEntryNodeValues

- **String[] EntryNode::getPossibleEntryNodeValues()**
  Returns the possible values of string enumeration, number enumeration, timezone or lookuptable entrynode.For other type of entrynodes an empty array would be returned.

getPrimaryKey

- **String Entry::getPrimaryKey()**
  Returns the primary key value of this entry.

getRootEntryNode
- **EntryNode Entry::getRootEntryNode()**
  Return the root entry node for this entry

getSourceEntrySetForRelatedEntries
- **EntrySet Entry::getSourceEntrySetForRelatedEntries(Container filterContainer)**
  Returns EntrySet with all entries that have an attribute related to this entry filtering by container if filterContainer is provided.

getXMLRepresentation
- **String Entry::getXMLRepresentation()**
  Returns an XML representation of this entry with the structure Nodes/Node/Name, Nodes/Node/Value, Paths/Path.

hasInheritedValue
- **Boolean EntryNode::hasInheritedValue()**
  Returns TRUE if this entry node WOULD inherit some non-null value if set to do so.

hasNonInheritedValue
- **Boolean EntryNode::hasNonInheritedValue()**
  Returns TRUE if there is a non-null non-inherited value. The presence or absence of inherited values makes no difference.

isEntryAnItem
- **Boolean Entry::isEntryAnItem()**
  Returns TRUE if this entry is an Item, FALSE if it is a Category.

isEntryCheckedOut
- **Boolean Entry::isEntryCheckedOut()**
  Returns true if the entry is checked out into a collaboration area otherwise it returns false.

isInheriting
- **Boolean Item::isInheriting(Category location, String sAttribPath)**
  Return true if the item inherits at a location for sAttribPath. The attribute will contain an unset value and will support inheritance. Note, no check is made that there is a value to inherit.

populateAllNonPersisted
- **Boolean Entry::populateAllNonPersisted()**
  Execute non-persisted script for all entrynodes in the entry. Return true if the script was completed succesfully, false otherwise

populateNonPersistedForEntryNode
- **Boolean EntryNode::populateNonPersistedForEntryNode()**
  Execute non-persisted script for this entrynode. Return true if the script was completed succesfully, false otherwise

previewEntryAttrib
- **String Entry::previewEntryAttrib(String sAttribPath)**
  Returns the preview string for displaying entry attribute specified by attribute path. Returns "" if sAttribPath refers to a non existing attribute or to a non-existing multi-occurrence instance

setEntryAttrib
- **void Entry::setEntryAttrib(String sAttribPath, Object sValue)**
  Sets the attribute sAttribPath (spec_name/attribute_name) of this entry to sValue. Perform optional checks before update if bDoChecks is true.

setEntryAttribValues

- **void Entry::setEntryAttribValues(String sAttribPath, Object[] sValues)**
  Sets the values of the multi-value attribute given sAttribPath (spec_name/attribute_name) of this entry.

setEntryNode

- **EntryNode EntryNode::setEntryNode(String sPath)**
  Return the entryNode with path sPath relative to EntryNode. If the path is not already built a NULL will be returned. Use the Entry::setEntryAttrib script operation to create a path that might not exist. Deprecated.

setEntryNodeRelationshipValue

- **Integer EntryNode::setEntryNodeRelationshipValue(Catalog relatedItemCtg, String sRelatedItemPrimaryKey)**
  Set the value of this EntryNode of type RELATIONSHIP to the related item represented by the given catalog and primary key. Return 1 if the value was set, 0 if nothing changed, and -1 if the item with the PK was not found.

setEntryNodeRelationshipValueUsingItem

- **Integer EntryNode::setEntryNodeRelationshipValueUsingItem(Item relatedItem)**
  Set the value of this EntryNode of type RELATIONSHIP to the related item given. Return 1 if the value was set, 0 if nothing changed, and -1 if the item with the PK was not found.

setEntryNodeValue

- **Integer EntryNode::setEntryNodeValue(Object value)**
  Set the value of this EntryNode and return 1 if the value was set, 0 if nothing changed, and -1 if there was a type conversion error.

setEntryRelationshipAttrib

- **void Entry::setEntryRelationshipAttrib(String sAttribPath, Catalog relatedItemCtg, String sRelatedItemPrimaryKey)**
  Sets the attribute sAttribPath (spec_name/attribute_path) of type RELATIONSHIP of this entry to the related item represented by the given catalog and primary key

setEntryRelationshipAttribUsingItem

- **void Entry::setEntryRelationshipAttribUsingItem(String sAttribPath, Item relatedItem)**
  Sets the attribute sAttribPath (spec_name/attribute_path) of type RELATIONSHIP of this entry to the related item given

setEntryStatus

- **void Entry::setEntryStatus(String status)**
  Sets the status of the entry

setInheriting

- **void Item::setInheriting(Category location, String sAttribPath, [Boolean flag])**
  By default or if flag is true, then set an item location's attribute to an unset value or a grouping to inheriting, which means that the attribute will inherit at this location. If flag is false, then the attribute will not inherit, meaning that the attribute holds a null override.

setPrimaryKey

- **void Entry::setPrimaryKey(String value)**
  Sets the primary key value of this entry.

throwValidationError

- **void EntryNode::throwValidationError(String errorType[, String errorString])**
  Sets up a validation error that shows up in validation errors in the gui and the list of

errors returned when an entry is saved in a script. ERRORTYPE should be one of "UNIQUENESS", "VALIDATION_RULE", "PATTERN", "MIN_OCCURENCE", "LENGTH".

operations_export

addAllObjectsToExport

- **void EnvObjectList::addAllObjectsToExport([String sObjectType, [String sActionMode]])**
  Notifies that all the entities of specific object type be exported. sObjectType is optional. sActionMode is used to specify the action mode in which the object has to be exported. It is optional. In case it is not specified, the value set using setActionModeForExport() is used. If no action mode has been set, CREATE_OR_UPDATE is used. List of acceptable values for sObjectType are: "ACG", "ALERT", "ATTRIBUTE_COLS", "CATALOG", "CATALOG_CONTENT", "CATALOG_VIEW", "COLLABORATION_AREA", "COLLABORATION_AREA_CONTENT", "COMPANY_ATTRIBUTES", "CONTAINER_ACCESSPRV", "DATASOURCE", "DESTINATION_SPEC", "DISTRIBUTION", "DISTRIBUTION_GROUP", "DOC_STORE", "EXPORTS", "FEEDS", "FILE_SPEC", "HIERARCHY", "HIERARCHY_CONTENT", "HIERARCHY_MAPS", "HIERARCHY_VIEW", "JOBS", "LOOKUP_TABLE", "LOOKUP_TABLE_CONTENT", "LOOKUP_TABLE_SPEC", "MAPS", "MY_SETTINGS", "PRIMARY_SPEC", "QUEUE", "REPORTS", "ROLES", "ROLE_LOCALE_ACCESS" "SELECTION", "SCRIPT_INPUT_SPEC", "SECONDARY_SPEC", "SPEC", "SUB_SPEC", "UDL", "UDL_CONTENT", "USERS", "WEBSERVICE", "WORKFLOW" List of acceptable values for sObjectType are: "CREATE_OR_UPDATE", "CREATE", "UPDATE", "DELETE"

addObjectByNameToExport

- **void EnvObjectList::addObjectByNameToExport(String sEntityName[, String sObjectType, [String sActionMode]])**
  Sets the entity to be exported by specifying the entity name as an argument. sObjectType is optional. sActionMode is used to specify the action mode in which the object has to be exported. It is optional. In case it is not specified, the value set using setActionModeForExport() is used. If no action mode has been set, CREATE_OR_UPDATE is used. In case of Catalog and Hierarchy Content export, this operation is used to specify the attribute collection associated with the object. In case of DocStore partial export, this operation is used to specify the DocStore path. List of acceptable values for sObjectType are: "ACG", "ALERT", "ATTRIBUTE_COLS", "CATALOG", "CATALOG_CONTENT", "CATALOG_VIEW", "COLLABORATION_AREA", "COLLABORATION_AREA_CONTENT", "COMPANY_ATTRIBUTES", "CONTAINER_ACCESSPRV", "DATASOURCE", "DESTINATION_SPEC", "DISTRIBUTION", "DISTRIBUTION_GROUP", "DOC_STORE", "EXPORTS", "FEEDS", "FILE_SPEC", "HIERARCHY", "HIERARCHY_CONTENT", "HIERARCHY_MAPS", "HIERARCHY_VIEW", "ITEM_CATEGORY_MAPS", "JOBS", "LOOKUP_TABLE", "LOOKUP_TABLE_CONTENT", "LOOKUP_TABLE_SPEC", "MAPS", "MY_SETTINGS", "PRIMARY_SPEC", "QUEUE", "REPORTS", "ROLES", "ROLE_LOCALE_ACCESS" "SELECTION", "SCRIPT_INPUT_SPEC", "SECONDARY_SPEC", "SPEC", "SUB_SPEC", "UDL", "UDL_CONTENT", "USERS", "WEBSERVICE", "WORKFLOW" List of acceptable values for sObjectType are: "CREATE_OR_UPDATE", "CREATE", "UPDATE", "DELETE"

exportEnv

- **String exportEnv(EnvObjectList envObjList, String sDocFilePath)**
  Exports the WebSphere Product Center objects specified in envObjList at the specified DocStore path. sDocFilePath is the filepath of the zip file that will be exported into the document store - returns the log as a string.

getCatalogNameToExport
- **String EnvObjectList::getCatalogNameToExport()**
  Returns the last value set with setCatalogByNameToExport

getHierarchyNameToExport
- **String EnvObjectList::getHierarchyNameToExport()**
  Returns the last value set with setHierarchyByNameToExport

getTypesToExport
- **String[] EnvObjectList::getTypesToExport()**
  Returns all the object types, set with setTypeToExport, for exporting

getTypeToExport
- **String EnvObjectList::getTypeToExport()**
  Returns the last object type set with setTypeToExport

new$EnvObjectList
- **new EnvObjectList()**
  Returns a container for the WebSphere Product Center objects to be exported. This class is used to add and retrieve the objects to be exported.

setActionModeToExport
- **void EnvObjectList::setActionModeToExport(String sActionMode)**
  Sets the default action mode for objects to be exported. The value specified in this method can be overridden by specifying the action mode in addAllObjectsToExport() or addObjectByNameToExport(). List of acceptable values for sObjectType are: "CREATE_OR_UPDATE", "CREATE", "UPDATE", "DELETE"

setCatalogByNameToExport
- **void EnvObjectList::setCatalogByNameToExport(String sCatalog)**
  Sets the Catalog whose contents are to be exported

setHierarchyByNameToExport
- **void EnvObjectList::setHierarchyByNameToExport(String sHierarchy)**
  Sets the Hierarchy whose contents are to be exported

setHierarchyMapToExport
- **void EnvObjectList::setHierarchyMapToExport(String sourceHierarchy, String destHierarchy [,String sActionMode])**
  Sets the source and destination Hierarchies whose mappings need to be exported. sActionMode is used to specify the action mode in which the object has to be exported. It is optional. In case it is not specified, the value set using setActionModeForExport() is used. If no action mode has been set, CREATE_OR_UPDATE is used. List of acceptable values for sObjectType are: "CREATE_OR_UPDATE", "CREATE", "UPDATE", "DELETE"

setItemCategoryMapToExport
- **void EnvObjectList::setItemCategoryMapToExport(String sCatalog, String sHierarchy [,String sActionMode)**
  Sets the Catalog and Hierarchy whose Item-Category mappings need to be exported

setTypeToExport
- **void EnvObjectList::setTypeToExport(String sObjectType)**
  Sets the object type to be exported. List of acceptable values for sObjectType are: "ACG", "ALERT", "ATTRIBUTE_COLS", "CATALOG", "CATALOG_CONTENT", "CATALOG_VIEW", "COLLABORATION_AREA", "COLLABORATION_AREA_CONTENT", "COMPANY_ATTRIBUTES", "CONTAINER_ACCESSPRV", "DATASOURCE", "DESTINATION_SPEC", "DISTRIBUTION", "DISTRIBUTION_GROUP", "DOC_STORE", "EXPORTS",

"FEEDS", "FILE_SPEC", "HIERARCHY", "HIERARCHY_CONTENT", "HIERARCHY_MAPS", "HIERARCHY_VIEW", "ITEM_CATEGORY_MAPS", "JOBS", "LOOKUP_TABLE", "LOOKUP_TABLE_CONTENT", "LOOKUP_TABLE_SPEC", "MAPS", "MY_SETTINGS", "PRIMARY_SPEC", "QUEUE", "REPORTS", "ROLES", "ROLE_LOCALE_ACCESS" "SELECTION", "SCRIPT_INPUT_SPEC", "SECONDARY_SPEC", "SPEC", "SUB_SPEC", "UDL", "UDL_CONTENT", "USERS", "WEBSERVICE", "WORKFLOW"

operations_import

createExport

- **String createExport(String marketSpecName, String catalogName, String specMapName, String exportScriptName, String syndicationName, [HashMap optionalArgs])**
  Creates the Export with given params. An optional parameter "charsetName", which may be set in the "optionalArgs" parameter, describes the file encoding of the export. Otherwise, the Cp1252 is chosen as the default file encoding. Returns Done if successful, Error if not. Here is a complete list of the optional arguments which may be set in the "optionalArgs" parameter: String approverUserName, String charsetName, String distributionName, String distributionGroupName, String selectionName, String synType, String diffType,String sParamsDocPath. The distributionName and distributionGroupName can be a list of distribution names delimited by the string returned by getStringValueForClassMember("com.ibm.ccd.common.util.Const", "CATEGORY_PATH_DELIMITER")

createImport

- **String createImport(String sImportName, String sImportType, String sSourceName, String sFileSpecName, String sCatalogName, String sSpecMapName, String sCategoryTreeName, String sScriptName, String sACGName, [HashMap optionalArgs])**
  Creates the Feed with given params. An optional argument "sCharsetName", which may be defined in the optionalArgs HashMap, describes the file encoding of the feed. Otherwise, Cp1252 is chosen as the default file encoding. Also, optional parameters to describe if the current container is a collaboration area, and the step path of the workflow step in to which the feed is to be done, could be specified. Returns Done if successful, Error if not. The complete list of optional arguments, which may be set in the optionalArgs parameter, is as follows: String sCharsetName, Boolean bIsCollaborationArea, String sWflStepPath, String sParamsDocPath, String sImportSemantic, and String sApproverUserName.

disableBatchProcessingForItems

- **void disableBatchProcessingForItems()**
  Sets up the import to not process items in bulk. This used to be achieved in earlier releases by setting up an import on a catalog different than the one the user wanted to import into.

loadImport

- **String loadImport(String sImportName, String sPath)**
  Loads file from the given DocStore path into the given Import. Returns Done if successful, Error if not.

startAggregationByName

- **void startAggregationByName(String sName, String sDocPath)**
  Run the feed called sName on the file sDocPath

startExportByName

- **Boolean startExportByName(String sName)**

Run the export called sName. Returns TRUE if success.

operations_import

importEnv

- **String importEnv(String sDocFilePath, [bFromFileSystem])**
  Imports the content of the archive in the docstore at sDocFilePath into this company. Returns the log as a String. WARNING: Transactional disruption will occur: This script operation will disrupt the current transaction - resulting in possible commit or rollback of prior data. The transactional state on completion is not guaranteed. This script operation should not be used inside a useTransaction or catchError block. This operation is very disruptive to a WPC company and should not be used on a live system.

operations_item

buildCtgItem

- **(deprecated) see new$CtgItem**

cloneItem

- **Item Item::cloneItem()**
  Create and return a clone of this item.

deleteCtgItem

- **void deleteCtgItem(Item itm)**
  Delete the catalog item itm

displayCtgItemAttrib

- **String Item::displayCtgItemAttrib(String sAttribPath)**
  Returns the html string for displaying item attribute specified by attribute path

getAvailableLocations

- **CategorySet Item::getAvailableLocations(Object locationOrCategoryTree)**
  Returns CategorySet of available locations.

getCatalog

- **Catalog Item::getCatalog() | Catalog SearchResultSet::getCatalog([int columnIndex])**
  Return the catalog object this item belongs to, or return the value of the designated column in the current row of this SearchResultSet object as a Catalog.

getCtgItemAllCategories

- **Category[] Item::getCtgItemCategories()**
  (Deprecated) See getCtgItemCategories. Return the all categories this item is mapped to,

getCtgItemAtOldVersion

- **Item Item::getCtgItemAtOldVersion()**
  Returns the old version of the item in the differences syndication.

getCtgItemAttrib

- **Object Item::getCtgItemAttrib(String sAttribPath)**
  Returns the value of the attribute sAttribPath (spec_name/attribute_name) of this item

getCtgItemAttribByPk

- **Object Catalog::getCtgItemAttribByPk(String pk, String sAttribPath)**

Returns the value of the attribute sAttribPath (spec_name/attribute_name) of this item

getCtgItemAttribNamesList

- **String[] Item::getCtgItemAttribNamesList([Boolean bAllAttributes])**
  Returns an array of String containing the attribute name of all the attributes of this item (optional parameter allows option exclude categorySpecificAttribute - true by default)

getCtgItemAttribsForKeys

- **String Item::getCtgItemAttribsForKeys (Object[] aAttribs [, String sDelimiter])**
  Gets the attributes for an item based upon the passed Object[] (declared: var aAttribs = [];) of attribute keys (paths). The resultant values are loaded into the value pair of the aAttribs mapping. By specifying the delimiter parameter, in addition to populating aAttribs mapping the operation returns a CSV string representation of the retrieved values separated by the delimiter character.

getCtgItemAttribsList

- **String[] Item::getCtgItemAttribsList()**
  Returns an array of String containing the paths (spec_name/attribute_name) of all the attributes of this item

getCtgItemAttributeNewValue

- **(deprecated) use Item::getCtgItemAttrib()**

getCtgItemAttributeOldValue

- **(deprecated) use Item::getCtgItemAtOldVersion()**

getCtgItemCategories

- **Category[] Item::getCtgItemCategories([String catTreeName] [, CategoryCache catCache])**
  Return the categories this item is mapped to. If catTreeName is given, returns the categories within that ctr only (use the default category tree if no category tree is passed). Also, can use an optional CategoryCache passed in catCache

getCtgItemCategoryPaths

- **String[] Item::getCtgItemCategoryPaths(String sPathDelimiter, [Boolean bWithRoot], [CategoryTree ctr])**
  Returns an array of delimited strings of the category paths this item belongs to. If ctr is given, returns the paths of the categories within that ctr only.

getCtgItemCategoryPathsForPrimaryKey

- **String[] Catalog::getCtgItemCategoryPathsForPrimaryKey(String sPrimaryKey, String sPathDelimiter, [Boolean bWithRoot], [CategoryTree ctr])**
  Returns an array of delimited strings of the category paths for the item with sPrimaryKey in Catalog. If ctr is given, returns the paths of the categories within that ctr only

getCtgItemCatSpecificAttribsList

- **String[] Item::getCtgItemCatSpecificAttribsList()**
  Returns an array of String containing the paths (spec_name/attribute_name) of all the category specific attributes of this item

getCtgItemDiffStatus

- **String Item::getCtgItemDiffStatus()**
  For content difference syndications, returns this item''s difference status (A, M, D, U)

getCtgItemId

- **Integer Item::getCtgItemId()**

Returns this item's Id

getCtgItemMappedAttrib
- **String Item::getCtgItemMappedAttrib(String sAttribMappedPath)**
  Returns the value of the attribute mapped to/from sAttribMappedPath (mapped_spec_name/attribute_name) of this item

getCtgItemMappedAttribs
- **HashMap Item::getCtgItemMappedAttribs()**
  Returns a HashMap with the mapped attributes values indexed by their path (mapped_spec_name/attribute_name) of this item

getCtgItemMappedAttribsList
- **String[] Item::getCtgItemMappedAttribsList()**
  Returns an array of String containing the paths (mapped_spec_name/attribute_name) of all the mapped attributes of this item

getCtgItemOrganizations
- **Organization[] Item::getCtgItemOrganizations()**
  Return the all organizations this item is mapped to,

getCtgItemOrganizations
- **Organization[] Item::getCtgItemOrganizations()**
  Return the all organizations this item is mapped to,

getCtgItemPrimaryKey
- **String Item::getCtgItemPrimaryKey()**
  Returns this item's primary key value

getErrorsForLocation
- **[ValidationError[] LocationErrors::getErrorsForLocation ()**
  [Returns the validation errors for the current location errors. There will be at least one validation error.

getExitValue
- **String Entry::getExitValue()**
  Returns the exit value, if set, of an entry in a workflow step. Assumed to be called from an IN(), OUT(), or TIMEOUT() step script function.

getGlobalErrors
- **ValidationError[] EntryValidationErrors::getGlobalErrors ()**
  Returns the validation errors for the global attributes. Will return an empty array if no such errors exists.

getItemRootEntryNodeForLocation
- **EntryNode Item::getItemRootEntryNodeForLocation(Category location)**
  Returns the root EntryNode for this item at the given location.

getItemRootEntryNodesHavingLocationData
- **EntryNode[] Item::getItemRootEntryNodesHavingLocationData()**
  Returns a list of EntryNodes, each is a root entryNode per location that has data defined.

getItemStatus
- **String Item::getItemStatus()**
  Return UNKNOWN, ADDED, MODIFIED, DELETED

getItemXMLRepresentation
- **String Item::getItemXMLRepresentation(Spec spec, boolean**

includePrimaryKeyValue[, String dateFormat])
Returns the XML representation of this item which is specific to the given spec. This representation can be consumed by the XML parser in the WPS portion of the WPC and WPS integration

getLinkedItemForNode
- **Item Item::getLinkedItemForNode(String node_path)**
  Returns the linked item associated with the specified node.

getLinkedItems
- **Hashmap[] Item::getLinkedItems()**
  Returns a list containing a Hashmap for each item linked to this item's primary key. Keys in the HashMap include "item_key","item_id","catalog_id", and "catalog_name".

getLocationErrors
- **LocationErrors[] EntryValidationErrors::getLocationErrors ()**
  Returns the locations errors for locations having validation errors. Will return an empty array if no such errors exists.

getLocationForErrors
- **ICategory LocationErrors::getLocationForErrors ()**
  Returns the category associted with the current location errors.

getLocationsHavingData
- **CategorySet Item::getLocationsHavingData(Object locationOrCategoryTree)**
  Returns the category set of locations for which this entry has location specific attributes defined under the specified location or category tree..

getOriginalItem
- **Item Item::getOriginalItem()**
  Returns the original picture of the item before modification. Deprecated. Please use Entry::getOriginalEntry

getValidationErrorEntryNode
- **EntryNode ValidationError::getValidationErrorEntryNode()**
  Return the EntryNode associated with this ValidationError

getValidationErrorMsg
- **String ValidationError::getValidationErrorMsg()**
  Return the error message associated with this ValidationError

getValidationErrorType
- **String ValidationError::getValidationErrorType()**
  Return the type associated with this ValidationError

initializeKeyValueMapping
- **Object[] initializeKeyValueMapping(Object[] aKeyset)**
  Create a linked hash map.

isCtgItemMappedToCategories
- **Boolean Item::isCtgItemMappedToCategories([CategoryTree ctr])**
  Returns true if the item is mapped to categories. If the optional argument ctr is given, returns true if the item is mapped to a category in ctr.

isItemAvailableInLocation
- **Boolean Item::isItemAvailableInLocation(Category location)**
  Returns true if item is mapped to the given location in the specified category tree.

locationHasData

- **Boolean Item::locationHasData(Category location)**
  Returns true if the location has data.

makeItemAvailableInLocation

- **void Item::makeItemAvailableInLocation(Category location, [Boolean bRecursive])**
  Makes this item available in a given location. Available means that an item can have location data for the given location. If bRecursive is true than make item available in all descendent locations.

makeItemAvailableInLocations

- **void Item::makeItemAvailableInLocations(Category[] locations, [Boolean bRecursive])**
  Makes this item available in the given locations. Available means that an item can have location data for the given location. If bRecursive is true than make item available in all descendent locations.

makeItemUnavailableInLocation

- **void Item::makeItemUnavailableInLocation(Category location, [Boolean bRecursive])**
  Makes this item unavailable in a given location. If bRecursive is true than make item unavailable in all descendent locations.

makeItemUnavailableInLocations

- **void Item::makeItemUnavailableInLocations(Category[] locations, [Boolean bRecursive])**
  Makes this item unavailable in the given locations. If bRecursive is true then make item unavailable in all descendent locations.

mapCtgItemToCategory

- **void Item::mapCtgItemToCategory(Category category, [Boolean addToPicture], [Boolean validateCategory])**
  Map this item to this category. If optional boolean addToPicture is false, the secondary specs will not be associated and cannot be set; useful for performance. If optional boolean validateCategory is true and the category's hierarchy does not have the VALIDATION_RULES option disabled, the mapping will only occur if the category passes validation. Validation is false by default.

mapCtgItemToOrganizations

- **void Item::mapCtgItemToOrganizations(Category[] organizations [, boolean bAdd])**
  Maps the item to all the organizations provided. If bAdd is true, the old mappings are added to otherwise they are overwritten to be the new set of organizations. Deprecated--Call moveCtgItemToCategories

moveCtgItemToCategories

- **void Item::moveCtgItemToCategories(Category[] categories), [, boolean bAdd])**
  Move item from existing categories to new set of categories, if bAdd is true, then category mappings will be added.

new$CtgItem

- **new CtgItem(String sCtgName/Catalog ctg], [Boolean bRunEntryBuildScript], [Boolean bBuildNonPersisted], [Boolean bBuildEmptyEntryPicture])**
  Returns a new item object. The argument can be a catalog name or a catalog object. The argument being a catalog object allows the propagation of attribute collections to process settings etc. to new items being built with this operation. If no catalog name/object is provided, then the default catalog from the current script context is used. bRunEntryBuildScript or bBuildNonPersisted should be set to false to disable the default behavior of this script operation to run the entry build script or build the

non-persisted attributes respectively for this new item.

removeCtgItemFromCategory

- **void Item::removeCtgItemFromCategory(Category category)**
  Remove mapping from this item to this category, if the mapping exists.

saveCtgItem

- **EntryValidationErrors Item::saveCtgItem()**
  Saves the item and returns the EntryValidationErrors object. Use operations EntryValidationErrors::getGlobalErrors() and EntryValidationErrors::getLocationErrors() to get the validation errors that may have prevented the save. WARNING: Transactional disruption will occur: When used in an import script, this script operation will commit any existing transaction, and will open a new transaction if a transaction did exist. This script operation should be used carefully; for example, it should not be called in a catchError block.

setCtgItemAttrib

- **boolean Item::setCtgItemAttrib(String sAttribPath, Object sValue)**
  Sets the attribute sAttribPath (spec_name/attribute_name) of this item to sValue. Returns true if it was set succesfully. Returns false if operation failed to set, or if old and new values are same

setCtgItemMappedAttrib

- **void Item::setCtgItemMappedAttrib(String sAttribPath, Object oValue)**
  Sets the attribute mapped to/from sAttribMappedPath (mapped_spec_name/attribute_name) of this item to sValue

setCtgItemMappedAttribs

- **void Item::setCtgItemMappedAttribs(HashMap hmPathValue, [SpecMap specmap])**
  Set the attributes of this item: hmPathValue should contain (path_y, value_x)'s; the item attribute path_x receives value_x if path_y is mapped to path_x in specmap - if no spec map is specified, the specmap of the import is being used.

setCtgItemPrimaryKey

- **void Item::setCtgItemPrimaryKey(String pk)**
  Sets this item's primary key value

setCtgItemRelationshipAttrib

- **void Item::setCtgItemRelationshipAttrib(String sAttribPath, Catalog relatedItemCtg, String sRelatedItemPrimaryKey)**
  Sets the attribute sAttribPath (spec_name/attribute_path) of type RELATIONSHIP of this item to the related item represented by the given catalog and primary key

setCtgItemRelationshipAttribUsingItem

- **void Item::setCtgItemRelationshipAttribUsingItem(String sAttribPath, Item relatedItem)**
  Sets the attribute sAttribPath (spec_name/attribute_path) of type RELATIONSHIP of this item to the related item given

setExitValue

- **Entry::setExitValue(String exitValue)**
  Set the exit value of an entry in a workflow step. Assumed to be called from an IN(), OUT(), or TIMEOUT() step script function.

setIgnoreCategorySpecificAttributes

- **void Item::setIgnoreCategorySpecificAttributes(Boolean bIgnore)**
  Set whether or not category specific attributes should be processed for the item

setItemAttributesFromXMLRepresentation

- **void Item::setItemAttributesFromXMLRepresentation(String xmlStr [,String dateFormat)**
  Updates this item based upon an XML representation which is created by the XML parser in the WPS portion of the WPC and WPS integration

setItemLocationAttrib

- **void Item::setItemLocationAttrib(Category location, String sAttribPath, Object sValue)**
  Sets the attribute sAttribPath (spec_name/attribute_name) of this item for the given location to sValue.

validateMappedAttribs

- **HashMap validateMappedAttribs(HashMap hmPathValue, [SpecMap specmap])**
  Validate a set of attribute values indexed by their mapped path against the destination spec

operations_jms

jmsCreateTextMsg

- **JMSMessage QueueSession::jmsCreateTextMsg(String msgText)**
  Creates a new JMS TextMessage using QueueSession information with the text provided.

jmsDisconnect

- **void QueueSession::jmsDisconnect(QueueConnection qcon)**
  Disconnects from the given queue manager.

jmsGetConnectionFactory

- **QueueConnectionFactory Context::jmsGetConnectionFactory(String jmsFactory)**
  Creates and returns a jms connection factory with the specified context.

jmsGetContext

- **Context jmsGetContext(String url, String jndiFactory)**
  Creates a JMS context.

jmsGetMessageCorrelationID

- **String JMSMessage::jmsGetMessageCorrelationID()**
  Returns a string containing Correlation Id for the JMS message.

jmsGetMessageID

- **String JMSMessage::jmsGetMessageID()**
  Returns a string containing the JMS message id.

jmsGetMessageProperties

- **HashMap JMSMessage::jmsGetMessageProperties()**
  Returns a hashmap from string property names to string values for those priorities.

jmsGetMQConnectionFactory

- **QueueConnectionFactory jmsGetMQConnectionFactory(String mqQueueManager, String mqHostname, String mqChannel, Integer mqPort)**
  Creates and returns a jms connection factory for communicating with MQ queues. Note that you do not need a Context to get an MQ connection factory whereas you need a Context for connecting to other JMS queues.

jmsGetQueue

- **javax.jms.Queue QueueSession::jmsGetQueue(String name)**

(DEPRECATED: use jmsGetQueueByName() instead). Returns a javax.jms.Queue object from the given QueueSession. NAME identifies the desired queue in a vendor-specific format.

jmsGetQueueByName

- **javax.jms.Queue jmsGetQueueByName(Context ctx, String name)**
  Returns a javax.jms.Queue object from the given JNDI Name and Context.

jmsGetQueueConnection

- **QueueConnection QueueConnectionFactory::jmsGetQueueConnection([String username], [String password])**
  Returns a JMS queue connection from the given connection factory. Uses the username and password supplied, or if no username or password is supplied, uses the defaults in common.properties if they exist, otherwise attempts to connect as the user running WebSphere Product Center.

jmsGetQueueSession

- **QueueSession QueueConnection::jmsGetQueueSession()**
  Returns a JMS queue session from the given connection factory.

jmsGetTextFromMsg

- **String JMSMessage::jmsGetTextFromMsg()**
  Returns a string containing the entire content of a JMS message, including headers.

jmsReceiveMsg

- **JMSMessage QueueSession::jmsReceiveMsg(String queueName, Integer timeout[, String messageSelector, JMSMessage messageToReceiveReplyFor, Context ctx])**
  (METHOD DEPRECATED. Use jmsReceiveMsgFromQueue() instead). Receives a JMS Message. Times out after TIMEOUT milliseconds. If INBOUNDQUEUE is not null, looks on that queue. If ctx is provided, INBOUNDQUEUE is assumed to be a JNDI name: otherise INBOUNDQUEUE is assumed to be a queue name in vendor-specific format. If INBOUNDQUEUE is null, and MESSAGETORECEIVEREPLYFOR is not null, looks on the queue defined in the "Reply-To" field of MESSAGETORECEIVEREPLYFOR. If INBOUNDQUEUE is null and MESSAGETORECEIVEREPLYFOR is null, throws an AustinException. We now know which queue will be used. If MESSAGESELECTOR and MESSAGETORECEIVEREPLYFOR are both null, selects the first message from that queue. Otherwise selects the first message from the queue (if any) fulfilling all of the conditions defined by MESSAGESELECTOR and MESSAGETORECEIVEREPLYFOR. If MESSAGETORECEIVEREPLYFOR is not null, rejects any message not having a correlation ID equal to MESSAGETORECEIVEREPLYFOR's message ID. If MESSAGESELECTOR is not null, rejects any message not fulfilling the condition defined in messageSelector. If no appropriate message is found, returns null.

jmsReceiveMsgFromQueue

- **JMSMessage QueueSession::jmsReceiveMsgFromQueue(javax.jms.Queue queue, Integer timeout[, String messageSelector, JMSMessage messageToReceiveReplyFor])**
  Receives a JMS Message. Times out after TIMEOUT milliseconds. If INBOUNDQUEUE is not null, looks on that queue. If INBOUNDQUEUE is null, and MESSAGETORECEIVEREPLYFOR is not null, looks on the queue defined in the "Reply-To" field of MESSAGETORECEIVEREPLYFOR. If INBOUNDQUEUE is null and MESSAGETORECEIVEREPLYFOR is null, throws an AustinException. We now know which queue will be used. If MESSAGESELECTOR and MESSAGETORECEIVEREPLYFOR are both null, selects the first message from that queue. Otherwise selects the first message from the queue (if any) fulfilling all of the conditions defined by MESSAGESELECTOR and MESSAGETORECEIVEREPLYFOR. If MESSAGETORECEIVEREPLYFOR is not

null, rejects any message not having a correlation ID equal to MESSAGETORECEIVEREPLYFOR's message ID. If MESSAGESELECTOR is not null, rejects any message not fulfilling the condition defined in messageSelector. If no appropriate message is found, returns null.

jmsSendMsg

- **JMSMessage QueueSession::jmsSendMsg(JMSMessage msg, String queueName[, HashMap properties, JMSMessage messageToReplyTo, Context ctx])**
  (METHOD DEPRECATED. Use jmsSendMsgToQueue() instead). Sends message MSG and returns MSG or null. The message is sent to the queue specified by OUTBOUNDQUEUE, unless OUTBOUNDQUEUE is null. If ctx is provided, OUTBOUNDQUEUE is assumed to be a JNDI name. If ctx is not provided, OUTBOUNDQUEUE is assumed to be a queue name in vendor-specific format. If OUTBOUNDQUEUE is null, MSG is sent to the reply-to queue of MESSAGETOREPLYTO, if MESSAGETOREPLYTO is provided. If OUTBOUNDQUEUE is null and MESSAGETOREPLYTO is not provided, throws an AustinException. If MESSAGETOREPLYTO is provided, the message id is read from it. PROPERTIES is a map from string keys to string values. There is one special (non-JMS) key: "TRIGO_INCOMING_REPLY_QUEUE". "TRIGO_INCOMING_REPLY_QUEUE" indicates the queue name to which an external application should send replies to this message. If ctx is provided, the value of "TRIGO_INCOMING_REPLY_QUEUE" is assumed to be a JNDI name: otherwise it is assumed to be a queue name in vendor-specific format.

jmsSendMsgToQueue

- **JMSMessage QueueSession::jmsSendMsgToQueue(JMSMessage msg, javax.jms.Queue outboundQueue [, HashMap properties, JMSMessage messageToReplyTo,])**
  Sends message MSG and returns MSG or null. The message is sent to the queue specified by OUTBOUNDQUEUE, unless OUTBOUNDQUEUE is null. If OUTBOUNDQUEUE is null, MSG is sent to the reply-to queue of MESSAGETOREPLYTO, if MESSAGETOREPLYTO is provided. If OUTBOUNDQUEUE is null and MESSAGETOREPLYTO is not provided, throws an AustinException. If MESSAGETOREPLYTO is provided, the message id is read from it. PROPERTIES is a map of string keys to string values with a single key value that is acted on. This special (non-JMS) key is "TRIGO_INCOMING_REPLY_QUEUE" whose value is the javax.jms.Queue object to which an external application should send the replies to this message.

jmsSetMessageText

- **void JMSMessage::jmsSetMessageText(String msgText)**
  Sets the provided text for the JMS TextMessage. Only JMS TextMessage type is supported.

operations_lkp

addRow

- **Boolean LookupTable::addRow(String sKey, String sValue), Boolean LookupTable::addRow(String sKey, String[] asValues)**
  Add a new row to this lookup table - with value(s) sValue/asValues for the key sKey. Returns TRUE if and only if the add was successful.

addRowByOrder

- **Boolean LookupTable::addRowByOrder(String sKey, String sValue), void LookupTable::addRow(String sKey, String[] asValues)**
  Add a new row to this lookup table - with value(s) sValue/asValues for the key sKey

containsUsingLookupTable

- **Boolean String::containsUsingLookupTable(LookupTable lkp)**
  Return true if and only if the string contains at least one of the keys from the lookup table

deleteLookupTable

- **deleteLookuptable(LookupTable lkp)**
  Delete the lookup table lkp. WARNING: Transactional disruption will occur: This script operation will roll back any existing transaction, and will leave the database connection in auto-commit. This script operation should be used carefully; for example, it should not be called in a catchError block.

getKeysFromValues

- **String[] LookupTable::getKeysFromValues(String[] values)**
  Reverse lookup of keys using values from the lookup table. The values can either be Paths in the Spec or the column number of the lookup table starting from 0 and not including the Key column.

getLkpByName

- **LookupTable getLkpByName(String name, [Boolean isReadOnly])**
  Returns the lookup table object with the corresponding name. By default the lookup table is read-only, but can be made mutable by setting the isReadOnly parameter to false.

getLkpId

- **Integer LookupTable::getLkpId()**
  Return the id of this lookup table.

getLkpKeys

- **String[] LookupTable::getLkpKeys()**
  Return the keys of this lookup table

lookup

- **String lookup(String sLookupTableName, String sKey [, String sSecKey]), String lookup(LookupTable lkp, String sKey [, String sSecKey])**
  Returns the sSecKey-th value for sKey in the lookup table sLookupTableName or lkp

lookupValues

- **String[] lookupValues(LookupTable lkp, String sKey)**
  Returns values for sKey in the lookup table lkp

put

- **void put(String sLkpTableName, String sStartKey, String sValue), void put(String sLkpTableName, String sStartKey, String[] asValues)**
  Put a new row in the lookup table sLkpTableName

replaceUsingLookupTable

- **String String::replaceUsingLookupTable(LookupTable lkp)**
  Return a string in which any substring matching a key in the lookup table is replace by the corresponding value

operations_locale

addToCompanyLocales

- **void addToCompanyLocales(Locale []companyLocales)**
  Adds the given locales to the list of locales that are defined for the company.

getCompanyLocales

- **Locale[] getCompanyLocales()**
  Returns the locales that are part of the current company

getCustomMessage
- **String getCustomMessage(String id, [Locale loc])**
  Given message id (and locale), returns description of the message.

getDefaultACGName
- **String getDefaultACGName()**
  Returns the name of default ACG in the current company

getDefaultAttrCollectionName
- **String getDefaultAttrCollectionName(String specName)**
  Returns the name of default lookup table hierarchy in the current company

getDefaultCharset
- **Locale getDefaultCharset()**
  Returns default charset of the current company

getDefaultLktHierarchyName
- **String getDefaultLktHierarchyName()**
  Returns the name of default lookup table hierarchy in the current company

getDefaultLocale
- **Locale getDefaultLocale()**
  Returns default locale of the current company

getDefaultOrgHierarchyName
- **String getDefaultOrgHierarchyName()**
  Returns the name of default organization hierarchy in the current company

getDefaultSpecDispNameAttribute
- **String getDefaultSpecDispNameAttribute()**
  Returns the display name attribute of default spec in the current company

getDefaultSpecName
- **String getDefaultSpecName()**
  Returns the name of default spec in the current company

getDefaultSpecNameAttribute
- **String getDefaultSpecNameAttribute()**
  Returns the name attribute of default spec in the current company

getDefaultSpecPathAttribute
- **String getDefaultSpecPathAttribute()**
  Returns the path attribute of default spec in the current company

getLocaleCode
- **String Locale::getLocaleCode()**
  Returns the 5 letter code (2 letter language code + underscore + 2 letter country code) for the given locale.

getLocaleDisplayName
- **String Locale::getLocaleDisplayName()**
  Returns a description of the locale suitable for display.

getLocalizedSpecNames
- **Spec[] getLocalizedSpecNames()**
  Returns all the specs that are localized.

getSystemMessageById
- **String getSystemMessageById(int id, [Locale loc])**
  Given message id (and locale), returns description of the message.

getSystemMessageByName
- **String getSystemMessageByName(String msg_name, [Locale loc])**
  Given message name (and locale), returns description of the message.

getUserLocale
- **Locale getUserLocale()**
  Returns the locale that is selected by the user for browsing content

new$Locale
- **new Locale(String language_code, String country_code)**
  Returns a locale with the language and country (two letter codes) combination specified. Throws exception if it is not supported.

removeFromCompanyLocales
- **void removeFromCompanyLocales(Locale []companyLocales)**
  Removes the given locales from the list of locales that are defined for the company. This will also remove the given locales from any specs that are localized using them.

replaceCompanyLocales
- **void replaceCompanyLocales(Locale []companyLocales)**
  Sets the given locales for the company. Removes any existing locales. This will also remove any locales removed as a result of this operation, from any specs that are localized using them. For example: Current locales are en_US and fr_FR. Calling replaceCompanyLocales({en_US,de_DE}) will result in (1) en_US and de_DE are removed from company. (2) company locales are set to en_US and de_DE (3) any specs localized with fr_FR will have fr_FR removed from them.

operations_mq

mqDisconnect
- **void MQQueueManager::mqDisconnect()**
  Disconnects from the given queue manager.

mqGetMessageDiagnostics
- **String mqGetMessageDiagnostics(MQMessage message)**
  Returns a string containing diagnostic information about the given message.

mqGetMessageId
- **String MQMessage::mqGetMessageId()**
  Returns the ID of the given message as a String containing a hexadecimal number.

mqGetQueueMgr
- **MQQueueManager mqGetQueueMgr(String hostname, String port, String channel, String queueMgrName)**
  Creates and returns a new MQ queue manager with the given properties.

mqGetReceivedMsg
- **MQMessage MQQueueManager::mqGetReceivedMsg(String queueName, String queueOpenOptions, String messageGetOptions)**
  Receives a message from queueName. Returns the message, as a MQMessage, or null.

mqGetReceivedMsgByMessageID

- **MQMessage MQQueueManager::mqGetReceivedMsgByMessageID(String queueName, String messageId, String passedInQueueOpenOptions, String passedInMessageGetOptions)**
  Finds the message in the given queue with given message ID. The ID is passed in a a String containing a hexadecimal number. Returns null if there is no such message in the given queue.

mqGetResponseToMsg

- **MQMessage MQQueueManager::mqGetResponseToMsg(MQMessage outgoingMessage, String queueOptions, String messageOptions)**
  Gets the response to the given message from the given queue.

mqGetTextFromMsg

- **String mqGetTextFromMsg(MQMessage mqMessage)**
  Returns a string containing the entire content of a MQMessage, including headers.

mqGetXMLMessageContent

- **String mqGetXMLMessageContent(String orgXmlMsg)**
  Discards any garbage at the beginning of the input string to get a XML document. More precisely, behaves as follows: If the input string is of the form A + B, where B is a valid XML document and A is any (possibly empty) string, this operation returns B. Otherwise, returns null.

mqSendReply

- **MQMessage MQQueueManager::mqSendReply(MQMessage receivedMsg, String msgText, String passedInQueueOpenOptions, String passedInMessagePutOptions)**
  Sends a reply to the given message, without indicating success or failure.

mqSendReplyWithStatus

- **MQMessage MQQueueManager::mqSendReplyWithStatus(MQMessage receivedMsg, String msgText, String status, String passedInQueueOpenOptions, String passedInMessagePutOptions)**
  Sends a reply to the given message, setting the feedback field to indicate the given status. Status must be one of the following (in upper or lower case): "SUCCESS", "FAIL", "VALCHANGE", "VALDUPES", "MULTIPLE_HITS", "FAIL_RETRIEVE_BY_CONTENT", "BO_DOES_NOT_EXIST", "UNABLE_TO_LOGIN", "APP_RESPONSE_TIMEOUT", "NONE".

mqSendTextMsg

- **MQMessage MQQueueManager::mqSendTextMsg(String msgText, String queueName, String queueOpenOptions, String messagePutOptions)**
  Sends a message provided in the String msgText over queueName. Returns the MQMessage

mqSendTextMsgWithReply

- **MQMessage MQQueueManager::mqSendTextMsgWithReply(String msgText, String queueName, String replyQueueName, String queueOpenOptions, String messagePutOptions)**
  Sends a message provided in the String msgText over queueName. The reply queue is specified. Returns the MQMessage object.

operations_mutablespec

addSubSpec

- **Boolean Spec::addSubSpec(Spec subSpec)**
  Adds an entire SubSpec using a SubSpec.

buildSpec

- **Spec buildSpec(String specName, String specType, [Spec specFileType])**
  Returns a new spec object with the given name and type. Valid types are PRIMARY_SPEC, SECONDARY_SPEC, FILE_SPEC, MKT_SPEC, SUB_SPEC, LKP_SPEC, SCRIPT_INPUT_SPEC. The optional parameter specFileType is actually mandatory for the spec of type FILE_SPEC but not applicable to any other type. specFileType specifies the data file type of the file spec. Valid data file types are "D", "T", "C", "F", "X", and "G" which stand for DELIMITED, TAB_DELIMITED, CSV, FIXEDWIDTH, XML, GENERATED_DURING_FEED. If the specFileType field is omitted when specifying a FILE_SPEC an AustinException is thrown.

buildSpecNode

- **Node buildSpecNode(Spec spec, String path, Integer order)**
  Returns a new node object of a spec with the given path and node order. Please make sure to use a spec that has been obtained using the new Spec() or buildSpec operation

buildTestSpec

- **Spec buildTestSpec(String name, String type, Integer fields)**
  Returns a new spec object with the specified name, type and number of fields in the spec

deleteSpec

- **void Spec::deleteSpec()**
  Delete this spec

exportXML

- **String IMutableSpec::exportXML()**
  Exports a WebSphere Product Center Spec to a String representing a XML file.

exportXSD

- **String IMutableSpec::exportXSD()**
  Exports a WebSphere Product Center Spec to a String representing the contents of XML Schema Definition.

importXML

- **IMutableSpec importXML(String filename)**
  Imports a XML file to a WebSphere Product Center Spec.

importXSD

- **IMutableSpec importXSD(String filename, String specName, String specType, String primaryKeyPath, String maxAncestors, String topLevelNamespace, String topLevelName, String archivedFilename)**
  Imports a XML Schema Definition file (.xsd) to a WebSphere Product Center Spec, using the given parameters.

new$Spec

- **new Spec(String specName, String specType, [String specFileType])**
  Returns a new spec object with the given name and type. Valid types are PRIMARY_SPEC, SECONDARY_SPEC, FILE_SPEC, MKT_SPEC, SUB_SPEC, LKP_SPEC, SCRIPT_INPUT_SPEC. The optional parameter specFileType is actually mandatory for the spec of type FILE_SPEC but not applicable to any other type. specFileType specifies the data file type of the file spec. Valid data file types are "D", "T", "C", "F", "X", and "G" which stand for DELIMITED, TAB_DELIMITED, CSV, FIXEDWIDTH, XML, GENERATED_DURING_FEED. If the specFileType field is omitted when specifying a FILE_SPEC an AustinException is thrown.

new$SpecLookupTableNode

- **new SpecLookupTableNode(Spec spec, String path, String lookupTableName,**

Integer order)
Returns a new node created in the spec according to the path and order with the specified Look up table attached.

new$SpecNode

- **new SpecNode(Spec spec, String path, Integer order)**
  Returns a new node created in the spec according to the path and order.

removeNode

- **Boolean Spec::removeNode(String path)**
  Removes a node from a spec.

saveSpec

- **void Spec::saveSpec()**
  Save this spec to the database

setAttribute

- **void Node::setAttribute(String sAttributeName, String sValue, [Boolean dontReplace])**
  Set an attribute of a node or a spec. Please consult the documentation for allowable values of sAttributeName. Common values are MAX_OCCURRENCE, MIN_OCCURRENCE, TYPE, DEFAULT_VALUE. If the optional third parameter "dontReplace" is supplied, and is true, or we are dealing with a node rather than a spec, sValue is added to any existing values for this attribute rather than replacing them.

setAttributes

- **void Node::setAttributes(String sAttributeName, HashMap sValues)**
  Set an attribute of a node or a spec to a set of values contained in the sValues HashMep. Any existing values are deleted before the new values are added. Please consult the documentation for allowable values of sAttributeName.

setNodeName

- **void IMutableSpec::setNodeName(String path, String newNodeName)**
  Renames the node in the given spec with the given path to newNodeName. Throws an exception if any of the following conditions is true: (i) there is no node at the given path in the given spec. (ii) there is a node at the given path in the given spec, but it is not a leaf node. (iii) there is a node at the given path in the given spec, but it is a primary key. (iv) there is already a node at the given path with newNodeName. (v) the name is an invalid name.

setPrimaryKeyPath

- **void Spec::setPrimaryKeyPath(String primaryKeyPath)**
  Sets the primaryKeyPath of this spec to the given path. Throws an AustinException under any of the following conditions: 1. The spec is not a primary spec or lookup spec. 2. The path does not exist in the spec. 3. The path refers to a node included from a SubSpec, and the node does not have minimum occurrence and maximum occurrence both set to 1.

operations_perf

beginPerf

- **beginPerf(String name)**
  Starts timing current block for perf. logging

endPerf

- **endPerf(String name)**
  Ends timing current block for perf. logging

getTimerElapsedTime
- **Integer Timer::getTimerElapsedTime()**
  Return the time elapsed between start and stop.

new$Timer
- **new Timer()**
  Create (and start) a timer.

startTimer
- **Timer::startTimer()**
  Start the timer.

stopTimer
- **Timer::stopTimer()**
  Stop the timer.

operations_queuemgr

createQueue
- **IMsgQueue createQueue (String queueName, String queueDesc, MsgQueueProtocolEnum protocol, String syncScriptPath))**
  Creates a new queue with the given parameters.

getMessageFromQueue
- **Message getMessageFromQueue (String queueName, Integer index)**
  Gets the indexth oldest message from the given queue, index starting with 1. For example, getMessageFromQueue("Queue1", 2) would return the 2nd oldest message from the queue with name "Queue1". If there is no such message or queue, returns null.

getMsgAppResponse
- **Void Message::getMsgAppResponse()**
  Initiates the request for response for a message.

getMsgAppResponseDoc
- **Doc Message::getMsgAppResponseDoc()**
  Returns the Doc object for the message.

getMsgAttachments
- **HashMap Message::getMsgAttachments ()**
  Returns a HashMap of attachment names to attachments for the given message..

getMsgByMsgId
- **Message getMsgByMsgId(String msgId)**
  Returns the message object with the message id msgId null otherwise.

getMsgDoc
- **Doc Message::getMsgDoc()**
  Returns the Doc object for the message.

getMsgId
- **String Message::getMsgId()**
  Returns the generated unique id for the message.

getMsgProtocolResponseDoc
- **Doc Message::getMsgProtocolResponseDoc()**
  Returns the Doc object for the message.

getMsgQueue
- **MsgQueue Message::getMsgQueue()**
  Returns the MsgQueue object for the message.

getMsgQueueName
- **String MsgQueue::getMsgQueueName()**
  Returns the name of this message queue.

qmgrGetMsgQueueByName
- **MsgQueue qmgrGetMsgQueueByName(String queueName)**
  Returns the queue if present in the system.

sendMsg
- **Message MsgQueue::sendMsg(Doc doc)**
  Sends the message. If successful, will return a message object. If it fails it will return null.

setMsgDoc
- **void Message::setMsgDoc(IDoc doc)**
  Sets the Doc object for the message.

operations_report

getReportByName
- **Report getReportByName(String reportName)**
  Return a report if one exists with the specified name and null otherwise

new$Report
- **new Report(String reportName, String reportScriptName, Distribution dist)**
  Return a new report object

renderHorizontalBars
- **String renderHorizontalBars(Integer barWidth, Integer barHeight, Integer[] anLengths, String[] asLabels)**
  Return an HTML table to display horizontal bars - anHeights[i] should have the length of the i-th bar and asLabels[i] the tooltip for the i-th bar

renderVerticalBars
- **String renderVerticalBars(Integer barWidth, Integer barHeight, Integer[] anLengths, String[] asLabels)**
  Return an HTML table to display vertical bars - anHeights[i] should have the length of the i-th column and asLabels[i] the tooltip for the i-th column

operations_scheduler

queryJobCompletionPercentage
- **Integer queryJobCompletionPercentage(Integer scheduleID)**
  Queries the completion percentage of the specified job. Method will return percent complete as Integer if the job is currently running, null otherwise.

queryJobStatus
- **String queryJobStatus(Integer scheduleID)**
  Queries the specified job if it is currently running. Method will return one of "Completed Running", "Running", "System Error", "Error Completing", "Not Started".

runJob

- **Integer runJob(String jobName, String jobType)**
  Runs the specified job immediately. Returns the scheduleID for the job. Job type will be one of "CTGTODB", "DBTOMKT", "REPORTEXE", or "CATALOGTOCATALOGEXPORT". * CTGTODB should be used for imports * DBTOMKT should be used for exports * REPORTEXE should be used for reports * CATALOGTOCATALOGEXPORT should be used for catalog exports

stopJob

- **void stopJob(Integer scheduleID)**
  Stops the specified job if it is currently running.

operations_search

copySearchItemData

- **void Selection::copySearchItemData(Item searchItem[, Boolean append])**
  Copy item search data to search selection where the item was retrieved from a search result set. Use the optional append argument if you want to add data to existing data.

copySearchItemLocationTreeData

- **void Selection::copySearchItemLocationTreeData(Item searchItem, CategoryTree locationTree[, Boolean append])**
  Copy item search data to search selection where the item was retrieved from a search result set. Data is added for locations for given location tree. Use the optional append argument if you want to add data to existing data.

deleteSearchTemplate

- **void SearchTemplate::deleteSearchTemplate()**
  Delete this search template

execute

- **SearchResultSet SearchQuery::execute()**
  Execute the search query.

executeInBackground

- **Schedule SearchQuery::executeInBackground(String selectionName)**
  Execute the search query in background and save result as a selection.

getBoolean

- **boolean SearchResultSet::getBoolean(int columnIndex)**
  Get the value of the designated column in the current row of this SearchResultSet object as a boolean.

getCategory

- **Category SearchResultSet::getCategory(int columnIndex)**
  Get the value of the designated column in the current row of this SearchResultSet object as a Category.

getDate

- **Date SearchResultSet::getDate(int columnIndex)**
  Get the value of the designated column in the current row of this SearchResultSet object as a Date.

getDouble

- **double SearchResultSet::getDouble(int columnIndex)**
  Get the value of the designated column in the current row of this SearchResultSet object as a double.

getFloat
- **float SearchResultSet::getFloat(int columnIndex)**
  Get the value of the designated column in the current row of this SearchResultSet object as a float.

getHierarchy
- **CategoryTree SearchResultSet::getHierarchy(int columnIndex)**
  Get the value of the designated column in the current row of this SearchResultSet object as a CategoryTree.

getInt
- **int SearchResultSet::getInt(int columnIndex)**
  Get the value of the designated column in the current row of this SearchResultSet object as an int.

getItem
- **Item SearchResultSet::getItem(int columnIndex)**
  Get the value of the designated column in the current row of this SearchResultSet object as an Item.

getLong
- **long SearchResultSet::getLong(int columnIndex)**
  Get the value of the designated column in the current row of this SearchResultSet object as a long.

getSearchTemplateByName
- **SearchTemplate getSearchTemplateByName(String name)**
  Return the search template with the given name. Otherwise it becomes null.

getSearchTemplateName
- **String SearchTemplate::getSearchTemplateName()**
  Return the name of this search template

getSpec
- **Spec SearchResultSet::getSpec(int columnIndex)**
  Get the value of the designated column in the current row of this SearchResultSet object as a Spec.

getString
- **String SearchResultSet::getString(int columnIndex)**
  Get the value of the designated column in the current row of this SearchResultSet object as a String.

isDefined
- **boolean SearchResultSet::isDefined(int columnIndex)**
  Return true if the value of the designated column in the current row of this SearchResultSet object is defined; otherwise, return false.

moveCursor
- **boolean SearchResultSet::moveCursor(int position)**
  Change cursor position, where 0 <= position < size(). So if size() = 100, you can set the position to 0, 1, ..., 98, 99. The return value is true if the cursor was moved (note that you will have to call next() to fetch the row), or false if the cursor could not be moved due to an incorrect position .

new$SearchQuery
- **new SearchQuery(String queryString)**
  Create a search query.

new$SearchSelection
- **new SearchSelection(Catalog catalog, String name)**
  Return an empty search selection.

new$SearchTemplate
- **new SearchTemplate(String name, Array attrGroupNames, Container container, String desc, [String colAreaName, String stepPath])**
  Return a new search template with the given name, container, and the set of attribute group names. Also, search templates in a collaboration area step can be defined by providing optional parameters colAreaName and stepPath.

reset
- **SearchResultSet::reset()**
  Reset cursor position to first position. Similar to calling moveCursor(0)

setItemLocationData
- **void Selection::setItemLocationData(Item item, CategoryTree locationTree, String[] locFullPaths, String delimiter, Boolean rootIncluded[, Boolean append])**
  Add item search data to search selection. Data is added for locations for given location tree as an array of full category paths. Use the given delimiter to delimit the path elements and set rootIncluded to true if path includes category tree root name. Use the optional append argument if you want to add to existing data.

operations_soap

invokeSoapServer
- **Object invokeSoapServer(String sURL, String sMethodName, Object[] aParamValues [,String[] aParamNames[, String userName, String password]])**
  Invoke a soap server. SURL is the URL of the service. SMETHODNAME is the name of the operation called. APARAMVALUES is an array containing the request parameters. APARAMNAMES is an optional array containing the names of the paramters. USERNAME is a username and company code separated by @, for example user@company to be passed in and used if authentication is required for this web service. PASSWORD is the password for the corresponding user in the company, and is only used if authentication is required for this web service. Returns the return value of the SOAP operation call.

invokeSoapServerForDocLit
- **Object invokeSoapServerForDocLit(String sURL, String xmlRequestMsg)**
  Invoke a soap server for Document-Literal based web services. SURL is the URL of the service. XMLREQUESTMSG is a string containing the request message in XML format.

operations_spec

addToSpecLocales
- **void Spec::addToSpecLocales(Locale []newLocales)**
  Adds the given locales to the list of locales that are defined for the spec.

buildSpecNodeName
- **String buildSpecNodeName(String name)**
  Returns the parsed name that was passed in so that it can be used as a spec node name (spec node name only accept letters and characters, others are converted to an underscore _)

getLocaleNode
- **Node Node::getLocaleNode(Locale locale)**

Returns the localized node for the supplied locale.

getLocales
- **Object Spec::getLocales()**
  returns the locales associated with the spec

getNodeAttributeValue
- **String Node::getNodeAttributeValue(String attributeName)**
  Returns the value of this node's attribute, i.e. MAXLENGTH, MAX_OCCURRENCE, MIN_OCCURRENCE, HELP_URL, TYPE, etc.

getNodeAttributeValues
- **HashMap Node::getNodeAttributeValues(String attributeName)**
  Returns the values of this node's attributes in a Hash Map, i.e. STRING_ENUMERATION.

getNodeByPath
- **Node Spec::getNodeByPath(String path)**
  Returns the node object for path in this spec.

getNodeChildren
- **INode[] Node::getNodeChildren()**
  Returns the children for the node

getNodeDisplayName
- **String Node::getNodeDisplayName([Locale locale])**
  Returns the display name of a locale node. Optionally, if the node is the parent of the locale nodes, pass in the locale for a particular locale node display name. If it is not valid for the node to have a display name, will return null.

getNodeLocale
- **Locale Node::getNodeLocale()**
  Returns the locale object for this node if it is a locale specific node.

getNodeLookupTableName
- **String Node::getNodeLookupTableName()**
  Returns the name of the Lookup Table associated with this node, if one exists.

getNodeName
- **String Node::getNodeName()**
  Returns the name of this node.

getNodePath
- **String Node::getNodePath()**
  Returns the path of this node.

getNodeSpec
- **Spec Node::getNodeSpec()**
  Returns the spec object for this node.

getPrimaryKeyNode
- **Node Spec::getPrimaryKeyNode()**
  Returns the primary-key node of this primary spec. If this is not a primary spec, returns null.

getSpecAttribNames
- **String[] Spec::getSpecAttribNames()**
  Returns the names of each attribute(node) specified in the spec

getSpecAttribPaths
- **String[] Spec::getSpecAttribPaths()**
  Returns the paths of each attribute(node) specified in the spec

getSpecByName
- **Spec getSpecByName(String name, [Boolean bImmutable])**
  Returns the spec object with the corresponding name. By default, a mutable spec is returned. If an immutable spec is needed, then an optional boolean parameter bImmutable is specified to be true. Please note that only mutable specs can be modified.

getSpecMultiOccurAttributePaths
- **HashMap Spec::getSpecMultiOccurAttributePaths()**
  Returns the multi occurence attribute paths for this spec.

getSpecName
- **String Spec::getSpecName()**
  Returns the name of this spec

getSpecNameList
- **String[] getSpecNameList(HashMap filters)**
  Returns the names of the Specs that match the given filters. VALID Filters: ("PATTERN", String) ("CONTAINER", Container Object) Will return only specs attached to container ("SPECTYPE", String {"PRIMARY_SPEC", "SECONDARY_SPEC", "LOOKUPTABLE_SPEC", "FILE_SPEC"} comma separated list) ("LOCALIZED", String {YES, NO}) Will return only localized or only non-localized specs

getSpecNodes
- **HashMap Spec::getSpecNodes()**
  Returns map of node paths to node objects for this spec.

getSpecPrimaryKeyAttributePath
- **String Spec::getSpecPrimaryKeyAttributePath()**
  Returns the primary key attribute path for this spec. Returns null if the path is not valid for this spec.

getSpecSequenceAttributePaths
- **HashMap Spec::getSpecSequenceAttributePaths()**
  Returns the sequence attribute paths for this spec.

getSpecType
- **String Spec::getSpecType()**
  Returns the type of this spec

getSpecUniqueAttributePaths
- **HashMap Spec::getSpecUniqueAttributePaths()**
  Returns the unique attribute paths for this spec.

isLocalized
- **Boolean Spec::isLocalized()**
  Returns a boolean to indicate whether or not a spec is localized

isNodeEditable
- **Boolean Node::isNodeEditable()**
  Returns true if the node is editable. Returns false otherwise

isNodeGrouping
- **Boolean Node::isNodeGrouping()**

Returns true if the node is a grouping node, false otherwise

isNodeIndexed
- **Boolean Node::isNodeIndexed()**
  Returns true if this node is indexed

isNodeNonPersisted
- **Boolean Node::isNodeNonPersisted()**
  Returns true if the node is a non-persisted node, false otherwise

isNodeSpecRoot
- **Boolean Node::isNodeSpecRoot()**
  Returns true if the node is a spec root node, false otherwise

loadSpecFromXML
- **void loadSpecFromXML(String specXml)**
  Creates spec defined in XMLSTRING. This spec can be loaded into different companies, like you would do from the command line

removeFromSpecLocales
- **void Spec::removeFromSpecLocales(Locale []newLocales)**
  Removes the given locales from the list of locales that are defined for the spec.

replaceSpecLocales
- **void Spec::replaceSpecLocales(Locale []newLocales)**
  Sets the given locales for the spec. Removes any existing locales.

setLocalized
- **void Spec::setLocalized(Boolean localized)**
  Sets the localized property of a spec

setNodeEditable
- **void Node::setNodeEditable(Boolean)**
  Sets the node to be editable or non-editable

setNodeIndexed
- **void Node::setNodeIndexed(Boolean)**
  Sets the node to be indexed or not

operations_specmap

buildTestSpecMap
- **SpecMap buildTestSpecMap(String mapName, String mapType, Object source, Object destination)**
  Returns a new spec map on the specified map type between the source and the destination - first delete existing map if there is one. The mapType can be FILE_CAT_MAP or CAT_MKT_MAP or FILE_CATALOG_MAP or CATALOG_MKT_MAP or CATALOG_CATALOG_MAP. If source or destination is catalog, user should pass Catalog object, else pass Spec object.

getDefaultSpecMapName
- **(deprecated) String getDefaultSpecMapName()**
  See getSpecMapByName. Returns the name of the spec map being used for an aggregation/syndication.

getSpecMapByName
- **SpecMap getSpecMapByName([String name])**
  Returns the specmap object with the corresponding name

getSpecMapDstObject
- **Object SpecMap::getSpecMapDstObject()**
  Returns the destination object of this spec map

getSpecMapSrcObject
- **Object SpecMap::getSpecMapSrcObject()**
  Returns the source object of this specmap

map
- **void SpecMap::map(String sSrcPath, String sDstPath)**
  Add a mapping from sSrcPath to sDstPath to this spec map

new$SpecMap
- **new SpecMap(String mapName, String mapType, Object source, Object destination)**
  Creates a new spec map of the given type between the source and destination objects. The mapType can be FILE_CAT_MAP or CAT_MKT_MAP or FILE_CATALOG_MAP or CATALOG_MKT_MAP or CATALOG_CATALOG_MAP.

saveSpecMap
- **void SpecMap::saveSpecMap()**
  Save this spec map to the database

operations_userdefinedlog

dumpUserDefinedLog
- **void UserDefinedLog::dumpUserDefinedLog(Writer out, String delim, String outputType, String docTag, HashMap hmNodeTags)**
  Dump all log entries from the user defined log to the Writer provided in no specific order. out - this is the output writer you want to dump the UDL to delim - the delimiter used for the current UDL entries outputType - one of COPY_UDE_OUTPUT, CSV_OUTPUT, XML_OUTPUT COPY_UDE_OUTPUT: dump each UDL entry exactly how it is currently stored CSV_OUTPUT: dump each UDL entry as comma seperated values XML_OUTPUT: dump each UDL entry within XML tags; docTag and hmNodeTags must also be specified docTag - this will comprise the XML tag surrounding the UDL dump hmNodeTags - this is the array of labels for each subtag to surround each delimited value

forEachUserDefinedLogEntry
- **forEachUserDefinedLogEntry(UserDefinedLog UDL, [Entry e,] String[] logEntries [, Boolean bReturnMultipleLogEntries = true]) { statements }**
  Executes the statements for each group of log entries in the given UserDefinedLog or, if Entry e is defined, each log entry for that specific Entry. If bReturnMultipleLogEntries is false, the array of log entries will contain only the first (oldest) log in chronological order. This is only a valid option if Entry e is not defined. If bReturnMultipleLogEntries is true, all logs are populated in the array in ascending chronological order for a given Entry (oldest first). By default, bReturnMultipleLogEntries is true.

getUserDefinedLog
- **UserDefinedLog Container::getUserDefinedLog(String name)**
  Returns the user defined log object having the given name, for this container

insertUserDefinedLog
- **void UserDefinedLog::insertUserDefinedLog()**
  Persist the new user defined log object to the database.

isUserDefinedLogNew
- **Boolean UserDefinedLog::isUserDefinedLogNew()**
  Check if the user defined log has been saved in the database.

newUserDefinedLog
- **UserDefinedLog Container::newUserDefinedLog(String name, String description, Boolean isRunningLog)**
  Returns a new user defined log object for this container with the given name and description. Will throw an exception if a log with the same name already exists for the container.

newUserDefinedLogEntry
- **newUserDefinedLogEntry(Date date, Container container, Entry entry, String log [, Entry category])**
  Returns a new user defined log entry object with for the specified entry which is either an item or category (with date/timestamp and log). If the category is also provided the logs will only be associated to that category.

saveUserDefinedLog
- **void UserDefinedLog::saveUserDefinedLog()**
  Update the persisted user defined log object in the database.

startBatchProcessingForUserDefinedLog
- **void UserDefinedLog::startBatchProcessingForUserDefinedLog()**
  Setup batch processing for the given User Defined Log. This operation is to be used mainly during import/mass update jobs.

stopBatchProcessingForUserDefinedLog
- **void UserDefinedLog::stopBatchProcessingForUserDefinedLog()**
  Stop batch processing for the given User Defined Log. This operation is to be used mainly during import/mass update jobs.

userDefinedLogAddEntry
- **void UserDefinedLog::userDefinedLogAddEntry(Entry entry, [String log_message], [Entry category])**
  Add an entry to the user defined log. If a message is specified, set that for the UserDefinedLogEntry. If the category is provided then the logs are only restricted for that category.

userDefinedLogDelete
- **void UserDefinedLog::userDefinedLogDelete()**
  Remove the user defined log object from the database. This action will also drop all entries to the log.

userDefinedLogDeleteEntriesFor
- **void UserDefinedLog::userDefinedLogDeleteEntriesFor(Entry entry [, Entry category])**
  Delete all log entries for an entry from the user defined log.

userDefinedLogDeleteEntry
- **void UserDefinedLog::userDefinedLogDeleteEntry(UserDefinedLogEntry entry)**
  Delete a particular entry from the user defined log.

userDefinedLogEntryGetDate
- **Date UserDefinedLogEntry::userDefinedLogEntryGetDate()**
  Get the date of the user defined log entry.

userDefinedLogEntryGetTarget
- **Entry UserDefinedLogEntry::userDefinedLogEntryGetTarget([Boolean**

containerIsCatalog)]
Get the entry object of the user defined log entry. If CONTAINERISCATALOG is true or is left unspecified, the entry must be in a catalog. If CONTAINERISCATALOG is false, the entry must be in a hierarchy.

userDefinedLogEntryGetValue
- **String UserDefinedLogEntry::userDefinedLogEntryGetValue()**
  Get the value of the user defined log entry.

userDefinedLogEntrySetDate
- **void UserDefinedLogEntry::userDefinedLogEntrySetDate(Date date)**
  Set the date of the user defined log entry.

userDefinedLogEntrySetValue
- **void UserDefinedLogEntry::userDefinedLogEntrySetValue(String log_message)**
  Set the log of the user defined log entry.

userDefinedLogGetContainer
- **Container UserDefinedLog::userDefinedLogGetContainer()**
  Get the container that is logged by the user defined log.

userDefinedLogGetDescription
- **String UserDefinedLog::userDefinedLogGetDescription()**
  Get the description of the user defined log.

userDefinedLogGetEntriesFor
- **UserDefinedLogEntry[] UserDefinedLog::userDefinedLogGetEntriesFor(Entry entry[, Entry category])**
  Get all log entries for an entry from the user defined log. The category can be provided in order to get the logs associated for that category only.

userDefinedLogGetName
- **String UserDefinedLog::userDefinedLogGetName()**
  Get the name of the user defined log.

userDefinedLogIsRunningLog
- **Boolean UserDefinedLog::userDefinedLogIsRunningLog()**
  Returns whether this user defined log is a running-log.

userDefinedLogSetDescription
- **void UserDefinedLog::userDefinedLogSetDescription(String desc)**
  Set the description of the user defined log. NOTE: You need to call insertUserDefinedLog/saveUserDefinedLog to persist this change.

userDefinedLogSetName
- **void UserDefinedLog::userDefinedLogSetName(String name)**
  Set the name of the user defined log. NOTE: You need to call insertUserDefinedLog/saveUserDefinedLog to persist this change.

operations_webservices

createWebService
- **WebService createWebService(String name, String implclass, String desc, String wsdlDocPath, String wsddDocPath, String protocol, String style, String implScriptPath, Boolean storeIncoming, Boolean storeOutgoing, Boolean deployed [, Boolean authRequired, Boolean skipRequestValidation, Boolean skipResponseValidation])**
  Creates a new web service with the given parameters. To save and deploy the web

service(if DEPLOYED is true), call saveWebService(). NAME is the name of the service. IMPLCLASS is the java class for java based web services or "" for script based ones, DESC is the description of the service. WSDLDOCPATH is the doc path at which the WSDL is stored. WSDDDOCPATH is the doc path at which the WSDD is stored. PROTOCOL is the protocol. Currently, "SOAP_HTTP" is the only supported protocol. STYLE is the message style. Currently, RPC_ENCODED and DOCUMENT_LITERAL are supported. IMPLSCRIPTPATH is the doc path of the service implementation script. It is the callers responsibility to ensure that WSDLDOCPATH, WSDDDOCPATH and IMPLSCRIPTPATH do not cause the documents for any other web service to be overwritten. STOREINCOMING determines whether incoming requests are stored. STOREOUTGOING determines whether outgoing request are stored. DEPLOYED determines whether the service will be deployed. AUTH_REQUIRED determines whether a username, company name, and password are required to invoke this web service. SKIPREQUESTVALIDATION determines whether the inbound SOAP message is validated against WSDL schema. SKIPRESPONSEVALIDATION determines whether the outbound SOAP message is validated against WSDL schema. If a web service with the name of NAME already exists, throws an AustinException.

deleteWebService
- **void WebService::deleteWebService()**
  Deletes the web service in the DB and undeploys it.

getDesc
- **String WebService::getDesc()**
  Returns the description of this web service

getImplclass
- **String WebService::getImplclass()**
  Returns the fully qualified name of the implementation class of this web service

getImplScriptPath
- **String WebService::getImplScriptPath()**
  Returns the docstore path where the implementation script for this web service is stored.

getName
- **String WebService::getName()**
  Returns the name of this web service

getProtocol
- **String WebService::getProtocol()**
  Returns the protocol for this web service.

getStoreIncoming
- **Boolean WebService::getStoreIncoming()**
  Returns whether incoming messages for this web service are stored.

getStoreOutgoing
- **Boolean WebService::getStoreOutgoing()**
  Returns whether outgoing messages for this web service are stored.

getStyle
- **String WebService::getStyle()**
  Returns the style for this web service.

getUrl
- **String WebService::getUrl()**
  Returns the URL for this web service

getWebServiceByName
- **WebService getWebServiceByName (String name)**
  Returns the web service with the given name. If there is no such web service, returns null.

getWsddDocPath
- **String WebService::getWsddDocPath()**
  Returns the docstore path where the WSDD for this web service is stored.

getWsdlDocPath
- **String WebService::getWsdlDocPath()**
  Returns the docstore path where the WSDL for this web service is stored.

getWsdlUrl
- **String WebService::getWsdlUrl()**
  Returns the WSDL URL for this web service

isAuthRequired
- **Boolean WebService::isAuthRequired()**
  Returns whether this web service requires authentication

isDeployed
- **Boolean WebService::isDeployed()**
  Returns whether this web service is deployed.

listTransactions
- **void WebService::listTransactions()**
  List the recorded transactions in order of date (undocumented, for internal use only).

saveWebService
- **void WebService::saveWebService()**
  Saves the web service in the DB. If deployment setting have changed, they take effect upon saving.

setAuthRequired
- **void WebService::setAuthRequired(Boolean authRequired)**
  Sets whether this WebService requires authentication. The setting will take effect upon saving.

setDeployed
- **void WebService::setDeployed(Boolean deployed)**
  Sets whether this WebService is deployed. The setting will take effect upon saving..

setDesc
- **void WebService::setDesc(String desc)**
  Sets the description of the given WebService.

setImplclass
- **void WebService::setImplclass(String implclass)**
  Sets the fully qualified name of the implementation class of the given WebService.

setImplScriptPath
- **void WebService::setImplScriptPath(String implScriptPath)**
  Sets the docstore path of the implementation script for this webservice. The caller must ensure that this does not overwrite the implementation script for any other service.

setName
- **void WebService::setName(String name)**

Sets the name of the given WebService.

setProtocol
- **void WebService::setProtocol(String protocol)**
  Sets the protocol of the given WebService.

setStoreIncoming
- **void WebService::setStoreIncoming(Boolean storeIncoming)**
  Sets the storeIncoming of the given WebService.

setStoreOutgoing
- **void WebService::setStoreOutgoing(Boolean storeOutgoing)**
  Sets whether this WebService should store outgoing messages.

setStyle
- **void WebService::setStyle(String style)**
  Sets the style of the given WebService.

setWsddDocPath
- **void WebService::setWsddDocPath(String wsddDocPath)**
  Sets the docstore path of the WSDD document. The caller must ensure that this does not overwrite the WSDD for any other service.

setWsdlDocPath
- **void WebService::setWsdlDocPath(String wsdlDocPath)**
  Sets the docstore path of the WSDL document. The caller must ensure that this does not overwrite the WSDL for any other service.

operations_wfl

createNestedWflStep
- **WorkflowStep Workflow::createNestedWflStep(Workflow nestedWfl)**
  Adds a nested workflow step to the workflow. Returns the WorkflowStep object.

createWflStep
- **WorkflowStep Workflow::createWflStep(String stepType, String stepName)**
  Adds a new step to the workflow if the step with the given name does not exists. StepType can be one of the following: AND_APPROVAL, OR_APPROVAL, MODIFY, DISPATCH, MERGE, GENERAL, AUTOMATED, INTERIM_CHECKOUT, CONDENSER. Returns the WorkflowStep object.

deleteWfl
- **void Workflow::deleteWfl()**
  Delete a workflow. It throws an exception if the workflow can not be deleted, for example, if it is used by any collaboration area

getAllWflNames
- **String[] getAllWflNames()**
  Returns a list of all workflow names.

getWflAccessControlGroup
- **String Workflow::getWflAccessControlGroup()**
  Returns access control group name of the workflow.

getWflByName
- **Workflow getWflByName(String wflName)**
  Returns the workflow if found otherwise null.

getWflContainerType
- **String Workflow::getWflContainerType()**
  Returns the workflow container type. The type could be either 'CATALOG' or 'CATEOGRY_TREE'

getWflDesc
- **String Workflow::getWflDesc()**
  Returns the workflow name.

getWflFailureStep
- **WorkflowStep Workflow::getWflFailureStep()**
  Returns the failure step of the workflow.

getWflInitialStep
- **WorkflowStep Workflow::getWflInitialStep()**
  Returns the initial step of the workflow.

getWflName
- **String Workflow::getWflName()**
  Returns the workflow name.

getWflStepByName
- **WorkflowStep Workflow::getWflStepByName(String stepName)**
  Returns the step of the workflow otherwise null.

getWflStepPaths
- **String[] Workflow::getWflStepPaths()**
  Returns the paths for all the steps of the workflow.

getWflSteps
- **WorkflowStep[] Workflow::getWflSteps()**
  Returns the list of all the steps in the workflow.

getWflSuccessStep
- **WorkflowStep Workflow::getWflSuccessStep()**
  Returns the success step of the workflow.

new$Workflow
- **new Workflow(String wflName, String containerType)**
  Create a new workflow of the given container type and with the given name. Container type can be one of the following: CATALOG, CATEGORY_TREE

saveWfl
- **Boolean Workflow::saveWfl()**
  Saves the workflow. Returns true or false depending on whether the workflow was successfully saved or not.

setCategoryTreesForRecategorization
- **void Workflow::setCategoryTreesForRecategorization(String[] categoryTreeNames)**
  Sets the category trees which will be modified by this workflow. If no category trees are set that would mean that ALL of the category trees associated to the source container will be modified by this workflow.

setWflAccessControlGroup
- **void Workflow::setWflAccessControlGroup(String acg)**
  Sets access control group name of the workflow.

setWflDesc

- **void Workflow::setWflDesc(String wflDesc)**
  Sets the workflow description

setWflName

- **void Workflow::setWflName(String wflName)**
  Sets the workflow name

operations_wflstep

getEditableAttributeGroups

- **String[] WorkflowStep::getEditableAttributeGroups([String subViewType], [String locationHierarchyName])**
  Gets the editable attribute groups of a workflow step. The result is an array attribute collection names. The optional parameter subViewType can be 'ITEM_LOCATION', 'BULK_EDIT', 'ITEM_EDIT', 'CATEGORY_EDIT', or 'CATEGORY_BULK_EDIT'. The optional parameter locationHierarchyName is required when the subViewType is 'ITEM_LOCATION'.

getLocationHierarchyNames

- **String[] WorkflowStep::getLocationHierarchyNames([Boolean canModifyAvailability])**
  Returns the list of location hierarchy names defined in the given workflow step. The optional parameter canModifyAvailability filters the list of location hierarchy names based on the 'modify location hierarchy availability' flag. If not specified, no filtering takes place.

getModifyLocationHierarchyAvailability

- **Boolean WorkflowStep::getModifyLocationHierarchyAvailability(String locationHierarchyName)**
  Returns the 'modify location hierarchy availability' flag for a given location hierarchy in the given workflow step.The optional parameter locationHierarchyName is required when the subViewType is 'ITEM_LOCATION'.

getNextWflStepsForExitValue

- **String[] WorkflowStep::getNextWflStepsForExitValue(String exitValue)**
  Returns the names of the next steps for a particular exitValue of a WorkflowStep.

getRequiredAttributeGroups

- **String[] WorkflowStep::getRequiredAttributeGroups([String subViewType], [String locationHierarchyName])**
  Gets the required attribute groups of a workflow step. The result is an array attribute collection names. The optional parameter subViewType can be 'ITEM_LOCATION', 'BULK_EDIT', 'ITEM_EDIT', 'CATEGORY_EDIT', or 'CATEGORY_BULK_EDIT'. The optional parameter locationHierarchyName is required when the subViewType is 'ITEM_LOCATION'.

getViewableAttributeGroups

- **String[] WorkflowStep::getViewableAttributeGroups([String subViewType], [String locationHierarchyName])**
  Gets the viewable attribute groups of a workflow step. The result is an array attribute collection names. The optional parameter subViewType can be 'ITEM_LOCATION', 'BULK_EDIT', 'ITEM_EDIT', 'CATEGORY_EDIT', or 'CATEGORY_BULK_EDIT'. The optional parameter locationHierarchyName is required when the subViewType is 'ITEM_LOCATION'.

getWflStepAddEntries

- **Boolean WorkflowStep::getWflStepAddEntries()**
  Returns value of 'allow import into step' flag.

getWflStepAttributeGroups
- **String[] WorkflowStep::getWflStepAttributeGroups()**
  Returns an array of all the attribute group names for the workflow step.

getWflStepCategorizeEntries
- **Boolean WorkflowStep::getWflStepCategorizeEntries()**
  Returns value of 'allow recategorization' flag.

getWflStepDefaultScriptPath
- **String WorkflowStep::getWflStepDefaultScriptPath()**
  Gets the default path of the workflow script for the step: scripts/workflow/<workflow name>/<step name>.

getWflStepDesc
- **String WorkflowStep::getWflStepDesc()**
  Returns the workflow step name.

getWflStepEntryNotification
- **String WorkflowStep::getWflStepEntryNotification()**
  Gets the notification emails which will get sent when the item gets into the step.

getWflStepExitValues
- **String[] WorkflowStep::getWflStepExitValues()**
  Retrieve the exit values of the WorkflowStep.

getWflStepName
- **String WorkflowStep::getWflStepName()**
  Returns the workflow step name.

getWflStepPerformerRoles
- **String[] WorkflowStep::getWflStepPerformerRoles()**
  Returns the list of user roles for the workflow step.

getWflStepPerformerUsers
- **String[] WorkflowStep::getWflStepPerformerUsers()**
  Returns the list of user names for the workflow step.

getWflStepReserveToEdit
- **Boolean WorkflowStep::getWflStepReserveToEdit()**
  Returns the reserve for edit flag for a workflow step.

getWflStepScriptPath
- **String WorkflowStep::getWflStepScriptPath()**
  Gets the path of the workflow script for the step. If no script is defined, returns null.

getWflStepTimeoutDate
- **Date WorkflowStep::getWflStepTimeoutDate()**
  Gets the timeout date for the workflow step. If no timeout date was set, a null is returned.

getWflStepTimeoutDuration
- **String WorkflowStep::getWflStepTimeoutDuration()**
  Gets the timeout duration for the workflow step. Returns an integer in seconds. If no timeout duration was set, 0 is returned.

getWflStepTimeoutNotification
- **String WorkflowStep::getWflStepTimeoutNotification()**
  Gets the notification emails which will get sent when the step times out.

getWflStepType

- **String WorkflowStep::getWflStepType()**
  Returns the workflow step type.

getWflStepView

- **CtgView WorkflowStep::getWflStepView(String subViewType, [String locationHierarchyName])**
  Returns a ctg view with a give subViewType for the workflow step. The parameter subViewType can be 'ITEM_LOCATION', 'BULK_EDIT', 'ITEM_EDIT', 'CATEGORY_EDIT', or 'CATEGORY_BULK_EDIT'. The optional parameter locationHierarchyName is required when the subViewType is 'ITEM_LOCATION'.

getWflStepViews

- **CtgView[] WorkflowStep::getWflStepViews()**
  Returns an array of all the step views for the workflow step.

mapWflStepExitValueToNextStep

- **void WorkflowStep::mapWflStepExitValueToNextStep(String exitValue, String | WorkflowStep | String[] nextStep | WorkflowStep[] nextStep)**
  Maps the exit value of the WorkflowStep to the nextStep. The nextStep can either be the stepName or one WorkflowStep or an array of StepNames or an array of WorkflowSteps.

setEditableAttributeGroups

- **void WorkflowStep::setEditableAttributeGroups(String subViewType, String[]/AttrGroup[] attrGroups, [String locationHierarchyName])**
  Sets the editable attrinute groups for the workflow step for a given subViewType. The parameter subViewType can be 'ITEM_LOCATION', 'BULK_EDIT', 'ITEM_EDIT', 'CATEGORY_EDIT', or 'CATEGORY_BULK_EDIT'. The optional parameter locationHierarchyName is required when the subViewType is 'ITEM_LOCATION'. WorflowStep cannot be of type "SUCCESS", as it is hardwired that an Entry must validate against its Container Spec in order to leave the Success step.

setModifyLocationHierarchyAvailability

- **void WorkflowStep::setModifyLocationHierarchyAvailability(String locationHierarchyName, Boolean canModifyAvailability)**
  Sets the 'modify location hierarchy availability' flag for a given location hierarchy in the given workflow step.

setRequiredAttributeGroups

- **void WorkflowStep::setRequiredAttributeGroups(String subViewType, String[]/AttrGroup[] attrGroups, [String locationHierarchyName])**
  Sets the required attrinute groups for the workflow step for a given subViewType. The parameter subViewType can be 'ITEM_LOCATION', 'BULK_EDIT', 'ITEM_EDIT', 'CATEGORY_EDIT', or 'CATEGORY_BULK_EDIT'. The optional parameter locationHierarchyName is required when the subViewType is 'ITEM_LOCATION'. WorflowStep cannot be of type "SUCCESS", as it is hardwired that an Entry must validate against its Container Spec in order to leave the Success step.

setViewableAttributeGroups

- **void WorkflowStep::setViewableAttributeGroups(String subViewType, String[]/AttrGroup[] attrGroups, [String locationHierarchyName])**
  Sets the viewable attrinute groups for the workflow step for a given subViewType. The parameter subViewType can be 'ITEM_LOCATION', 'BULK_EDIT', 'ITEM_EDIT', 'CATEGORY_EDIT', or 'CATEGORY_BULK_EDIT'. The optional parameter locationHierarchyName is required when the subViewType is 'ITEM_LOCATION'. WorflowStep cannot be of type "SUCCESS", as it is hardwired that an Entry must validate against its Container Spec in order to leave the Success step.

setWflStepAddEntries
- **void WorkflowStep::setWflStepAddEntries(Boolean flag)**
  Sets value of 'allow import into step' flag.

setWflStepCategorizeEntries
- **void WorkflowStep::setWflStepCategorizeEntries(Boolean flag)**
  Sets value of 'allow recategorization' flag.

setWflStepDesc
- **void WorkflowStep::setWflStepDesc(String desc)**
  Sets the desc for the workflow step.

setWflStepEntryNotification
- **void WorkflowStep::setWflStepEntryNotification(String emailAdresses)**
  Sets up the notification emails which will get sent when the item gets into the step. Email addresses must be seperated by semi-colons.

setWflStepExitValues
- **void WorkflowStep::setWflStepExitValues(String[] exitValues)**
  Sets the exit values for the workflow step.

setWflStepPerformerRoles
- **void WorkflowStep::setWflStepPerformerRoles(String[] roles)**
  Sets the user roles for the workflow step.

setWflStepPerformerUsers
- **void WorkflowStep::setWflStepPerformerUsers(String[] users)**
  Sets the users for the workflow step.

setWflStepReserveToEdit
- **void WorkflowStep::setWflStepReserveToEdit(Boolean flag)**
  Sets the reserve for edit flag for a workflow step.

setWflStepScriptPath
- **void WorkflowStep::setWflStepScriptPath([String scriptPath])**
  Sets up the workflow script path for this step. If no argument is passed, the defaut location is used (script/<workflow name>/<step name>). Note that this operation does not check that the script is already loaded (it allows you to load the script later if needed).

setWflStepTimeoutDate
- **void WorkflowStep::setWflStepTimeoutDate(Date date)**
  Sets up the timeout date for the workflow step.

setWflStepTimeoutDuration
- **void WorkflowStep::setWflStepTimeoutDuration(Integer seconds)**
  Sets up the timeout duration for the workflow step. The duration is in seconds.

setWflStepTimeoutNotification
- **void WorkflowStep::setWflStepTimeoutNotification(String emailAdresses)**
  Sets up the notification emails which will get sent when the step times out. Email addresses must be seperated by semi-colons.

operations_widget

buildWidget
- **Widget buildWidget(String sType, String sName)**

Creates a widget of type sType and name sName

getWidgetProperty

- **Object Widget::getWidgetProperty(String sPropertyName)**
  Return the property sPropertyName of this widget

invalidate

- **void invalidate()**
  Invalidates this widget

pullPropertyFromWidget

- **pullPropertyFromWidget(String sDestProperty, Object oSrcWidget, String sSrcWidgetProperty**
  The value of sDestProperty on this widget will always reflect the value of sSrcWidgetProperty on oSrcWidget - oSrcWidget is either a widget or a property of this widget that holds a widget

pushPropertyToWidget

- **pullPropertyFromWidget(String sSrcProperty, Object oDestWidget, String sDestWidgetProperty**
  The value of sDestWidgetProperty on oDestWidget will always reflect the value of sSrcProperty on this widget - oDestWidget is either a widget or a property of this widget that holds a widget

renderWidget

- **Widget::renderWidget(Writer out)**
  Renders the widget w

setWidgetProperty

- **void Widget::setWidgetProperty(String sPropertyName, Object oValue)**
  Set the property sPropertyName of this widget to the value oValue

operations_worklist

addWorkEntry

- **void WorkEntryList::addWorkEntry(int index, WorkEntry workEntry)**
  Insert a WorkEntry into the WorkEntryList at the specified index

getEntryFromWorkEntry

- **Entry WorkEntry::getEntryFromWorkEntry()**
  Get the Entry held by this WorkEntry

getIndexesOfEntriesHavingState

- **Map WorkEntryList::getIndexesOfEntriesHavingState(String state)**
  Get the current indexes of the worklist entries having a particular state

getMarkedEntries

- **EntrySet WorkEntryList::getMarkedEntries([start, end])**
  Return an entry set containing the marked entries in this work entry list - with indexes between start and end -

getWorkEntryAt

- **WorkEntry WorkEntryList::getWorkEntryAt(int i)**
  Get the WorkEntry for the specified index in the WorkEntryList

getWorkEntryListSize

- **Integer WorkEntryList::getWorkEntryListSize()**
  Gets the size of this work entry list

getWorkEntryState
- **String WorkEntry::getWorkEntryState()**
  Get the current state of this WorkEntry

isWorkEntryMarked
- **Boolean WorkEntry::isWorkEntryMarked()**
  Is the current WorkEntry marked

isWorkEntryMarkedNew
- **Boolean WorkEntry::isWorkEntryMarkedNew()**
  Is the current WorkEntry marked new

markWorkEntryDirty
- **void WorkEntry::markWorkEntryDirty()**
  Mark this WorkEntry as being dirty

new$WorkEntry
- **new WorkEntry(Entry entry, [Boolean markAsNew])**
  Creates a workentry for a given entry

new$WorkEntryList
- **new WorkEntryList(ctgOrSelection, [sortingNodeId], [sortingOrder])**
  Create a new work entry list from a catalog or a selection

removeWorkEntry
- **void WorkEntryList::removeWorkEntry(int index)**
  Removes the WorkEntry at the specified index from the WorkEntryList

saveMarkedEntries
- **WorkEntryList::saveMarkedEntries(workList, [start, end, [colArea, path, comment]])**
  Save the set of marked entries for this work entry list - with indexes between start and end - - for entries in the step specified by path in the collaboration area colArea with given comment.

setWorkEntryMarked
- **void WorkEntry::setWorkEntryMarked(Boolean mark)**
  Marks/unmarks this WorkEntry

syncWorkEntryAt
- **void WorkEntryList::syncWorkEntryAt(int i)**
  Sync the work entry at the specified index with it's database picture

other

getAribaAttribute
- **String getAribaAttribute(String attribName)**
  Gets Ariba's constant attribute names. Valid attribute names are PAYLOADID, TIMESTAMP, SHAREDSECRET, AUSTINDUNS

setBypassApproval
- **void setBypassApproval(Boolean bypassApproval)**
  If an approval workflow is setup, use this to bypass the approval process

sleep
- **void sleep(String)**
  Sleeps for the given number of milliseconds.

page_layout

getPageLayoutByName

- **PageLayout getPageLayoutByName(String sPageLayoutName)**
  Returns the page layout object with the corresponding name

new$PageLayout

- **new PageLayout(String sPageLayoutName)**
  Returns a new page layout with the given name

savePageLayout

- **void PageLayout::savePageLayout()**
  Saves the current page layout

re

buildRE

- **new RE(String pattern, Integer matchFlags)**
  Returns a regular expression corresponding to the given pattern. Match flags are 0=caseSensitive, 1=ignoreCase, 2=matchMultiline (new lines match as ^ and $, 4=matchSingleLine (treat multiple lines as one line). Flags are additive.

match

- **String[] RE::match(String str)**
  Return the contents of the parenthesized subexpressions after a successful match

new$RE

- **new RE(String pattern, Integer matchFlags)**
  Returns a regular expression corresponding to the given pattern. Optional match flags are 0=caseSensitive, 1=ignoreCase, 2=matchMultiline (new lines match as ^ and $, 4=matchSingleLine (treat multiple lines as one line). Flags are additive.

substitute

- **String RE::substitute(String substituteIn, String substitution)**
  Substitutes a string for this regular expression in another string. This method works like the Perl function of the same name. Given a regular expression of "a*b", a String to substituteIn of "aaaabfooaaabgarplyaaabwackyb" and the substitution String "-", the resulting String returned by subst would be "-foo-garply-wacky-". Returns: The string substituteIn with zero or more occurrences of the current regular expression replaced with the substitution String (if this regular expression object doesn't match at any position, the original String is returned unchanged).

reader

forEachLine

- **forEachLine(BufferedReader in, String line) { statements }**
  Executes the statements for each line read from in

forEachXMLNode

- **forEachXMLNode([XMLNode rootNode], String xPath [, XMLNode node]) { statements }**
  Executes the statements for each XML node having the relative path xPath - paths in the block are relative to xPath. If the node variable is passed in as an argument, it is populated with the XMLNode that is being operated on in each iteration of forEachXMLNode. If the rootNode is specified, the path is relative to the path of

rootNode.

getCurrentLine
- **String getCurrentLine()**
  Returns the current line

new$CSVParser
- **new CSVParser(BufferedReader reader)**
  Returns a comma separated parser given the buffered reader

new$DelimParser
- **new DelimParser(BufferedReader reader, String delimiter)**
  Returns a delimiter parser which parses, based on the given delimiter

new$FixedWidthParser
- **new FixedWidthParser(BufferedReader reader)**
  Returns a new fixed width parser given the buffered reader

new$Reader
- **new Reader(String documentPath [, String charsetName])**
  Returns the buffered reader for the document specified by the path. If the document path starts with "file://", then the reader will read the file system file given by the specified path. You may optionally specify a charset that differs from the one stored with the document in the doc store.

newCSVParser
- **CSVParser newCSVParser(BufferedReader input)**
  Returns a Comma Separated Parser using the given buffered reader input

newDelimParser
- **DelimiterParser newDelimParser(BufferedReader input, String delim)**
  Returns a parser which parses based on the delimiter provided

newFixedWidthParser
- **FixedWidthParser newFixedWidthParser(BufferedReader input, [Integer fieldPos1, Integer fieldPos2, ..., Integer fieldPosN])**
  Returns a fixed width parser given the buffered reader input. fieldPos are optional parameters which indicate the positions of the fields.

nextLine
- **String nextLine (BufferedReader in)**
  Returns the next line from the reader

parseXMLNode
- **String parseXMLNode (String sXMLSubPath)**
  Deprecated: Returns the value given by the sXMLSubPath XPath in the current XML document

parseXMLNodeWithNameSpace
- **String parseXMLNodeWithNameSpace (String sXMLSubPath)**
  Returns the value given by the sXMLSubPath XPath in the current XML document. When specify the XPath value, the user has the choice to specifying a namespace uri qualified path or using literal path matching specified when using the parseXMLNode script operation

splitLine
- **String[] Parser::splitLine()**
  Returns an array of tokens obtained by breaking the line using this parser (e.g. CSV parser, fixed width parser)

reflect

createJavaArray

- **Object createJavaArray(String typeName, Integer dim0 [,Integer dim1.....Integer dim9])**
  Create an array of type typeName. The number of dims specified indicates the number of dimensions that the array will be created with. The value of these numbers indicates the number of elements in that dimension. e.g supplying 1 and 4 as the dims would indicate that a 2 dimensional array will be created; the first dimension containing 1 element, the second containing 4 elements.If an array of primitives is to be created, supply the type as the java primitive keyword such as "int" or "boolean". If the type is a class name, it should be fully qualified and should not be an interface.

createJavaConstructor

- **Constructor createJavaConstructor(String className [,String type0,String type1.....String type9])**
  Create a java.reflect.Constructor Object by reflection using a className and optional types. If the constuctor you wish to target contains primitive arguments, supply those arguments with type the java primitive keyword such as "int" or "boolean". The className should be fully qualified and should not be an interface.. className should not be a primitive class (i.e. Class literal names such as int.class or int.TYPE are not accepted). In order to pass an array type use [] for one dimentional arrays and multiple []s for multiple dimention arrays. e.g. to target a 2 dimension array of ints pass "int[][]" to target a 1 dimensional array of Strings pass "java.lang.String[]".

createJavaMethod

- **Method createJavaMethod(String className,String methodName [,String type0,String type1.....String type9])**
  Create a java.reflect.Method Object by reflection using a className, methodName and optional types. className and methodName should not be null. The className should be fully qualified. The className may be a fully qualified interface name. If the method you wish to target contains primitive arguments, those arguments should be supplied with the java primitive keyword such as "int","boolean".The className should not be primitive classes (i.e. Class literal names such as int.class or int.TYPE). In order to pass an array type use [] for one dimentional arrays and multiple []s for multiple dimension arrays. e.g. to target a 2 dimension array of ints pass "int[][]" to target a 1 dimensional array of Strings pass "java.lang.String[]".

javaArrayFromScriptArray

- **Object javaArrayFromScriptArray(Array scriptArray, String type)**
  Transforms the provided scriptArray into a java array holding the same elements in the same order. If scriptArray is null, returns null. The user must provide the type (or subtype) of the array's elements. The types can be primitive java data types (int, char, byte, float, boolean, long, double, short) or any valid java class (eg. java.lang.String , java.lang.Integer). Type can also be a multidimentional array of elements(primitive/non primitive) with the brackets intact (eg int[], java.lang.Integer[][]). A fully qualified name is to be provided whenever using a class as type

runJavaConstructor

- **Object runJavaConstructor(Constructor constructor [,Object arg0,Object arg1.....Object arg9])**
  Run the supplied Constructor (which can be created using a createConstructor call). Run the constructor using the supplied objects as parameters. The supplied constructor should not be null. The supplied parameters should match the number and types associated with the Constructor. If the Constructor parameters contain primitives, then these parameters should not be supplied as null Objects and should be supplied as the appropriate wrapper primitive object (such as instances of

java.lang.Integer).

runJavaMethod

- **Object runJavaMethod(Object obj, Method method [,Object arg0,Object arg1.....Object arg9])**
  Run the supplied Method (which was created using a previous createJavaMethod call). If the Method is not static, then it is invoked on the supplied object using the supplied parameters. For static methods the supplied obj is ignored. The supplied Method should not be null. For instance methods the supplied object should not be null. The number of supplied parameters should match the number and types associated with the Method. If the Method parameters contain primitives, then these parameters should not be supplied as null Objects and should be supplied as the appropriate wrapper primitive object (such as instances of java.lang.Integer).

scriptArrayFromJavaArray

- **Object scriptArrayFromJavaArray(OneDimensionalJavaArray)**
  Transforms a 1 dimensional java array into a script array holding the same elements in the same order. If OneDimensionalJavaArray is not a 1 dimensional array or is null then null is returned. 1 dimension arrays of primitives can be supplied as the parameter.

scripting

setScriptProgress

- **void ::setScriptProgress(number percent)**
  Sets the percentage completed value in the context of a running script. This script is applicable in scripts running as part of jobs like import, export, report etc.

setScriptStatsDeletedCnt

- **void ::setScriptStatsDeletedCnt(number count)**
  Sets the count of items deleted in the context of a running script

security

authenticateWPCUser

- **Boolean authenticateWPCUser(String sUserName, String sPassword, String sCmpCode [, Boolean bEncodedPassword)**
  Provides authentication for a WPC user. Optional parameter bEncodedPassword indicates if the password is being passed already encoded. Default is false.

cloneUser

- **User cloneUser(String original_username, String username, String firstname, String lastname, String email, Boolean enabled, String password[, Category organization[, HashMap roles]])**
  Clones an existing user info into a new user. Password field is required. The optional roles and organization fields, when specified, override the roles and/or organization of the existing user.

createAccessControlGroup

- **ACG createAccessControlGroup(String sACGName, [String sACGDesc])**
  Creates an access control group object with the specified acg name and an optional acg description.

createRole

- **Role createRole(String sRoleName, [String sRoleDesc])**
  Creates a role object with the specified rolename and an optional role description.

createUser

- **User ::createUser(String username, String firstname, String lastname, String email, Boolean enabled, String password, HashMap roles, Category organization [, Boolean encryptPassword, Boolean enableLdap, String nameAttr, String serverUrl])**
  Creates an user with the specified parameters. Enabled, Password, Roles, and organization parameters are required. encryptPassword exists for the purpose of migrating environments so that encrypted passwords exported from one environment can be loaded into another environment without encrypting them again and that there is no possibility of knowing what the password was. EnableLdap marks the user as LDAP enabled and allows the provision of extra LDAP parameters, the LDAP name attribute and the LDAP Server URL

getAccessControlGroupByName

- **ACG getAccessControlGroupByName(String sACGName)**
  Returns a access control group object for the specified acg name

getAccessControlGroupName

- **String ACG::getAccessControlGroupName()**
  Return the name of the access control group

getAccessControlGroupPrivsForRole

- **String[] Role::getAccessControlGroupPrivsForRole(String acgName)**
  Gets the access control group privileges for the gvien access control group and the given role. The return parameter is an array of privileges (which are defined in the format: Catalog__list, Selection__list, SelectionMembers__view_items etc.).

getAccessControlGroupsForRole

- **String[] Role::getAccessControlGroupsForRole()**
  Gets the access control groups for the given role.

getAllUsers

- **User[] getAllUsers()**
  Returns all users

getCompanyCode

- **String getCompanyCode()**
  Returns the company code of this company.

getCompanyName

- **String getCompanyName()**
  Returns the name of this company.

getCurrentUserName

- **String getCurrentUserName()**
  Returns the name of the current user

getLdapEntryDn

- **String User::getLdapEntryDn()**
  Returns the distinguished name field associated with an LDAP authenticated User.

getLdapServerUrl

- **String User::getLdapServerUrl()**
  Return the URL of the server providing this users' LDAP authentication.

getLocalesForRole

- **String Role::getLocalesForRole()**
  Gets the locales that this role has access to for all containers

getLoginString
- **String getLoginString(String sUrl, Date dExpirationDate, [String sUserName])**
  Returns the url string needed for login automatically to the given url as the current user. If you are an admin, you can generate a login string for another user by passing the username as an extra parameter. Note that the url should not include the server name/port and should start with '/'. If an error occur, a null string is returned.

getRoleByName
- **Role getRoleByName(String sRoleName)**
  Returns a role object for the specified role

getRoleDescription
- **String Role::getRoleDescription()**
  Return the description of the role

getRoleName
- **String Role::getRoleName()**
  Return the name of the role

getRoles
- **Role[] getRoles()**
  Returns all roles for the current company

getRolesForCompany
- **Role[] getRolesForCompany(String sCmpCode)**
  Returns all roles of the given company

getUserAddress
- **String User::getUserAddress()**
  Return the User's Address

getUserByUsername
- **User getUserByUsername(String sUserName, [String sCmpCode])**
  Returns the User object for the given User Name and sCmpCode. If sCmpCode is not given, company code is taken from the current context of script execution

getUserCompanyCode
- **String User::getUserCompanyCode()**
  Return the User's Company Code

getUserCompanyName
- **String User::getUserCompanyName()**
  Return the User's Company Name

getUserEmail
- **String User::getUserEmail()**
  Return the User's Email Address

getUserEnabled
- **boolean User::getUserEnabled()**
  Returns if the User is enabled or not.

getUserFax
- **String User::getUserFax()**
  Return the User's Fax Number

getUserFirstName
- **String User::getUserFirstName()**
  Return the User's First Name

getUserLastName
- **String User::getUserLastName()**
  Return the User's Last Name

getUserLdapEnabled
- **boolean User::getUserEnabled()**
  Returns if the User is a LDAP user or not.

getUserName
- **String User::getUserName()**
  Return the User Name

getUserOrganizations
- **Category[] User::getUserOrganizations()**
  Return the User's Organizations

getUserPhone
- **String User::getUserPhone()**
  Return the User's Phone Number

getUserRoles
- **String[] User::getUserRoles()**
  Return the User's Roles

getUsers
- **User[] getUsers()**
  Returns all Users for the current company

getUsersFromRole
- **User[] Role::getUsersFromRole()**
  Returns all users within the Role

getUserTitle
- **String User::getUserTitle()**
  Return the User's Title

populateSecurityContext
- **AustinContext ::populateSecurityContext(User user[, String[] roleNames, InitialLdapContext ldapContext, ICategory organization])**
  Returns the context for the given user by assigning the access privileges for the roles passed in roleNames. It has no effect on the current users context.If ldapContext is present then a handle of the context will be set in the returned context. If the user is not already present in WPC a new user will be created in the organization specified otherwise in the default organization of default organization hierarchy.

saveUser
- **ValidationError[] User::saveUser()**
  Save the User's Profile. Returns null if the save was successful, otherwise returns an array of ValidationErrors.

setAccessControlGroupForRole
- **Boolean Role::setAccessControlGroupForRole(String acgName, String[] privs)**
  Sets an access control group with the given set of privileges for the role. The parameter privs is an array of privileges (which are picked from the strings in the format: Catalog__list, Selection__list, SelectionMembers__view_items etc.). Please note the the page privileges like PAGE_OBJ_CTG_CONSOLE__view, PAGE_OBJ_CAT_CREATE__view are stored only in the "Default" ACG.

setAccessControlGroupForRoleMigration
- **Boolean Role::setAccessControlGroupForRoleMigration(String acgName, String[] privs)**
  Script operation for migrating the old priv names to the new ones. Its exactly the same as setAccessControlGroupForRole operations except it has a mapping of old priv name to new ones.

setAllAccessControlGroupForRole
- **void Role::setAllAccessControlGroupForRole(String acgName, [String[] privExclusions])**
  Sets access control group acgName with all privileges except for the ones in privExclusions.

setLdapEntryDn
- **void User::setLdapEntryDn(String sEntryDN)**
  Sets the distinguished name field associated with an LDAP authenticated User

setLdapServerUrl
- **void User::setLdapServerUrl(String sServerUrl)**
  Sets the URL of the server providing this users' LDAP authentication.

setLocalesForRole
- **void Role::setLocalesForRole(String localesCSVString)**
  Sets the locales that this role has access to for all containers

setUserAddress
- **void User::setUserAddress(String str)**
  Set the User's Address

setUserEmail
- **void User::setUserEmail(String str)**
  Set the User's Email Address

setUserFax
- **void User::setUserFax(String str)**
  Set the User's Fax Number

setUserFirstName
- **void User::setUserFirstName(String str)**
  Set the User's First Name

setUserLastName
- **void User::setUserLastName(String str)**
  Set the User's Last Name

setUserLdapEnabled
- **void User::setUserLdapEnabled(boolean)**
  Sets the user as a LDAP user.

setUsername
- **void User::setUsername(String sUsername)**
  Sets the name of the current user

setUserPhone
- **void User::setUserPhone(String str)**
  Set the User's Phone Number

setUserRoles
- **Boolean User::setUserRoles(Role[] roles)**

Sets the roles for a user

setUserTitle
- **void User::setUserTitle(String str)**
  Set the User's Title

validateUser
- **boolean validateUser(String sUserName, String sPassword, String sCmpCode)**
  Confirms if the combination of User name, password and company id represent a valid and enabled WPC user. Returns true if the user could logon to WPC otherwise false.

set

containsKey
- **Boolean HashMap::containsKey(Object key)**
  Returns true if key exists.

containsValue
- **Boolean HashMap::containsValue(Object val)**
  Returns true if value exists.

forEachHmElement
- **forEachHmElement(HashMap hm, Object oKey, Object oValue) { statements }**
  Executes the statements for each (oKey, oValue) map in hm

intersectValues
- **HashMap intersectValues(HashMap hm1, HashMap hm2, ...)**
  Return the set-intersection of hm1, hm2, ... (only values are considered)

keyForValue
- **Object HashMap::keyForValue(Object valueToSearch)**
  Returns a key mapped to valueToSearch in hm or null

mergeValues
- **HashMap mergeValues(HashMap hm1, HashMap hm2, ...)**
  Return the set-union of hm1, hm2, ... (only values are considered)

size
- **Integer Object::size()**
  Returns the size of an object of type array, HashMap, or SearchResultSet.

sort
- **Array Array::sort()**
  Return the array sorted

string

buildCSV
- **String buildCSV (String str1, String str2, ..., String strN)**
  Takes a variable number of arguments, and returns a string with the arguments concatenated in csv format

buildDelim
- **String buildDelim (String delimiter, String qualifier, String str1, String str2, ..., String strN)**
  Takes a variable number of arguments, and returns a string with the arguments

concatenated in delim format, using the qualifier to enclose strings that contain the delimiter.

buildFixedWidth

- **String buildFixedWidth (String str1, Integer len1, String strN, Integer lenN)**
  Takes a variable number of arguments, and returns a string with the arguments concatenated in fixed width format.

checkDouble

- **Double checkDouble(String str, Double defaultValue)**
  If the input string is null or empty, the default value is returned. Otherwise the original value parsed as an Double is returned.

checkInt

- **Integer checkInt(String str, Integer defaultValue)**
  If the input string is null or empty, the default value is returned. Otherwise the original value parsed as an Integer is returned.

checkString

- **String checkString (String str, String defaultValue [, Boolean trim])**
  If the input string is null or empty, the default value is returned, otherwise the original value is returned. The input string will be trimmed of all leading and trailing spaces, unless a value of false for the optional TRIM parameter.

concat

- **String concat (String str1, String str2, ..., String strN)**
  Takes a variable number of arguments, and returns a string with the arguments concatenated in the order given

contains

- **Boolean String::contains (String match)**
  Tests if this string contains an occurence of the match substring

decodeUsingCharset

- **String String::decodeUsingCharset(String charset)**
  Returns a string by decoding the string using the named charset

encodeUsingCharset

- **String String::encodeUsingCharset(String charset)**
  Encodes the string using the named charset

endsWith

- **Boolean String::endsWith (String match)**
  Tests if this string ends with an occurence of the match substring

escapeForCSV

- **String escapeForCSV(String s)**
  Escape for CSV

escapeForHTML

- **String escapeForHTML(String s[,boolean isAscii])**
  Escape for HTML.By default isAscii is true. When isAscii is false the characters will be escaped with html entities

escapeForJS

- **String escapeForJS(String s)**
  Escape for JavaScript

escapeWithHTMLEntities

- **String escapeWithHTMLEntities(String str, Integer beg, Integer end)**
  Translates all character with HTML character codes less than beg or greater than end to HTML character codes

getNameFromPath

- **String getNameFromPath(String str[, String delimiter])**
  if str contains / returns the substring of str after the last / char exclusively, otherwise returns the original string

getParentPath

- **String getParentPath(String str)**
  if str contains / returns the substring of str up to the last / char exclusively, otherwise returns the empty string

getRidOfRootName

- **String getRidOfRootName(String str)**
  if str contains '/', gets rid of all preceding first '/' inclusive

indexOf

- **Integer String::indexOf (String match)**
  Returns the index within this string of the first occurrence of the specified match substring

isLowerCase

- **Boolean String::isLowerCase ()**
  Checks if all the characters in this string are lower case using the rules of the default locale

isStringSingleByte

- **Boolean isStringSingleByte(String s)**
  For SHIFT_JIS encoding, this returns true if the string is made of single byte characters only. False is returned otherwise

isUpperCase

- **Boolean String::isUpperCase ()**
  Checks if all the characters in this string are upper case using the rules of the default locale

lastIndexOf

- **Integer String::lastIndexOf (String match)**
  Returns the index within this string of the rightmost occurrence of the specified match substring

length

- **Integer String::length ()**
  Returns the length of this string

parseCSV

- **String[] String::parseCSV () | String String::parserCSV(Integer field)**
  Returns an array of each token, as parsed by the CSV parser. If a field number is provided, just the corresponding token substring is returned. A nullpointer exception is thrown if the string to be parsed is null.

parseDelim

- **String[] String::parseDelim (String delimeter) | String String::parseDelim (String delimeter, Integer iField)**
  Returns an array of each token, as parsed by the Delim parser. If a field number is provided, just the corresponding token substring is returned.

parseFixedWidth
- **String String::parseFixedWidth (Integer beginIndex, Integer endIndex)**
  Returns the corresponding token substring between the two indexes

removeHTML
- **String removeHTML (String str)**
  Returns a new string resulting from removing all html tags from the original string

replace
- **String replace (String str, String match, String replacement)**
  Returns a new string resulting from replacing all occurrences of the match substring in this string with the replacement substring

replaceCharsNotInDecRangeWithHex
- **String replaceCharsNotInDecRangeWithHex (String str, Integer iStartDecRange, Integer iEndDecRange, String sEncoding, String sQualifier)**
  Does the replace where iStartDecRange and iEndDecRange are inclusive

replaceString
- **String replaceString (String str, String match, String replacement)**
  Returns a new string resulting from replacing all occurrences of the match substring in this string with the replacement substring

resizeString
- **String resizeString (String str, Integer finalLength, Character padChar, Boolean padToTheRight)**
  Use to increase the size of a string to the finalLength by applying the appropriate padding to the left or right of the string with the given padChar.

startsWith
- **Boolean String::startsWith (String match)**
  Tests if this string begins with an occurence of the match substring

stripOutNonASCII
- **String stripOutNonASCII (String str)**
  Returns a new string resulting from removing all non-ASCII characters in this string

substring
- **String substring (String str, Integer beginIndex [, Integer endIndex])**
  Returns a new string that is a substring of this string. The beginIndex is inclusive but endIndex is not.

toLowerCase
- **String toLowerCase (String str)**
  Converts all of the characters in this string to lower case using the rules of the default locale

toTitleCase
- **String toTitleCase (String str)**
  Converts the first alphabet of all the words in a string to upper case

toUpperCase
- **String toUpperCase (String str)**
  Converts all of the characters in this string to upper case using the rules of the default locale

trim
- **String trim (String str)**
  Removes white space from both ends of this string

unescapeHTMLEntities

- **String urlEncode(String str)**
  Translates all character escaped with HTML character codes to corresponding characters

urlEncode

- **String urlEncode(String str)**
  Translates a string into x-www-form-urlencoded format

system

dumpContext

- **String dumpContext([Logger l])**
  Return the script context in a string (and dumps it to the logger l if specified)

dumpSystemLog

- **String dumpSystemLog(String sName, int nbLines)**
  Return the last nLines of the system log sName

getFunctionByName

- **FunctionObject ScriptObject::getFunctionByName(String sFunctionName)**
  Build the function object for the function sFunctionName in this script object

getLogger

- **Logger getLogger(String s)**
  Returns a logger (loggers are in the system log directory with the given name). "s" is a category name defined in log.xml ("com.ibm.ccd.wpc_user_scripting." is prepended automatically to "s".. Logger descriptions are stored in corresponding log files as specified in appender-ref in logs.xml for respective category name. If the name is not present in log.xml then by default logger descriptions are stored in default.log file under respective service directory.

getMemorySummary

- **String getMemorySummary()**
  Invokes the garbage collector, sleeps for 5 seconds and then returns a string summarizing memory usage.

getPageURL

- **String getPageURL(String pageName, Object[] requiredObject)**
  Return the URL for the page requested. The required objects are defined by the page itself which is limited to the following choices (Including their requirements): ITEM_LIST (Catalog, Category, CategoryTree) : displays all items in category. ITEM (Catalog, ItemId/PrimaryKey) : displays item. SEARCH (Catalog) : display rich search for the catalog. COLAREA_STEP (CollaborationArea, StepPath): displays all items/categories in step. COLAREA_ENTRY (CollaborationArea, StepPath, Item/Category PrimaryKey): displays item/category in step.

getProductCenterURL

- **Returns the property trigo_web_url defined in common.properties (which holds the fully-qualified URL, including port number, of the web site where users should point their browsers to access this instance of Product Center**
  Return the current script execution mode

getScriptByPath

- **ScriptObject getScriptByPath(String sScriptPath)**
  Build the script object for the script stored at sScriptPath in the DocStore. If the string starts with "file://" then the script will be loaded from the file system according to the

specified path

getScriptContextValue
- **Object getScriptContextValue(String sVariableName)**
  Return the value of the variable named sVariableName

getScriptExecutionMode
- **String getScriptExecutionMode()**
  Return the current script execution mode

getSystemDefaultEncoding
- **String getSystemDefaultEncoding()**
  Return the value of the system's default encoding

invoke
- **Object FunctionObject::invoke(Object arg1, Object arg2, etc)**
  Invoke this function object with the arguments arg1, arg2, etc

loggerDebug
- **void Logger::loggerDebug(String s)**
  Write s to this logger

loggerError
- **void Logger::loggerError(String s)**
  Write s to this logger

loggerFatal
- **void Logger::loggerFatal(String s)**
  Write s to this logger

loggerInfo
- **void Logger::loggerInfo(String s)**
  Write s to this logger

loggerWarn
- **void Logger::loggerWarn(String s)**
  Write s to this logger

runScript
- **void ScriptObject::runScript(HashMap hmContext)**
  Run this script

setScriptContextValue
- **void setScriptContextValue(String sVariableName, Object oVariableValue)**
  Set the value of the variable named sVariableName. This is a way of defining a variable within a script but this must be used with caution. There are already a number of implicit system-defined variables and this script op should not be used to redefine any of these implicit variables. If an implicit variable is redefined the results may be unpredictable. Note that there are a number of implict variables which whose names start with a $ sign. This script op must not be used to define any variables whose name starts with a $ sign. Again, the results may be unpredictable if a variable whose name starts with a dollar sign is defined. The following is a list of the implicit variables (not including those whose name starts with a $ sign): all_itemset_fetch_linked_item, all_itemset_readonly, attribute_group, bypass_approval_workflow, catalog, category, category_tree, colArea, collaboration_area, container, destination_attribute, entry, entrynode, entrySet, err, err_lines, http_request, in, inputs, invoking_user, item, job, lkpTable, location, location_tree, locationRootEntryNode, logger, lookup_table, message, msg_attachments, multi_request, node, organization, organization_type,

original_doc_folder,out, outs, page, page_layout, queueid, request, res run_rule_per_occurence, save_event, sequence, soapFaultCode, soapFaultMsg, soapIncomingAttachments, soapMessage, SoapOperationName, soapOutgoingAttachments, soapParams, spec, spec_map, special_outs, specmap_script_dest_attrib, step, stepPath, this, top, val, workflow, workIndex, workList, wrn.

startTransaction

- **startTransaction { statements }**
  Executes the statements in a transaction, rollback takes place if an error occurs. Does not do anything if a transaction is already open

useTransaction

- **useTransaction { statements }**
  Executes the statements in a transaction, rollback takes place if an error occurs

timezone

getTimeZoneDesc

- **String getTimeZoneDesc(int offsetInMinutes,Locale locale)**
  Get the time zone's description with the offset value in minutes.

getTimeZoneOffsetFromDBValue

- **Number getTimeZoneOffsetFromDBValue(String dbValue)**
  Get time zone from the db value and return the offset from GMT in minutes.

getUserTimeZoneDesc

- **String getUserTimeZoneDesc()**
  Get the user setting time zone's description in native language.

getUserTimeZoneOffset

- **Number getUserTimeZoneOffset()**
  Get user setting time zone's offset from GMT in minutes.

parseTimeZoneToDBValue

- **String parseTimeZoneToDBValue(String srcStr)**
  Parse the string to time zone then return the db value.

setUserTimeZone

- **void setUserTimeZone(int offset)**
  Change user setting's time zone with the offset value in minutes.

writer

close

- **void Writer::close([String path])**
  Close this writer, and stores its content in the doc store location specified by path, or if the string starts with "file://", on the file system according to the specified path

createOtherOut

- **Writer createOtherOut(String name, [String charset])**
  Returns a new writer with the given name and an optional charset value. If the string starts with "file://" then the writer will write into the file system file given by the specified path

print

- **void Writer::print(Object o)**

Writes o as a string into this writer

println

- **void Writer::println(Object o)**
  Writes o as a string and appends a new line to it into this writer

printXML

- **void Writer::printXML(String sTagName, String sValue [, String sAttributes] [, boolean escape])**
  Writes an XML tag with the text value sValue, the tag name sTagName and the attributes sAttributes. The value of escape if given as true, will print the tag with angle brackets surrounding it, converted to escape characters.

save

- **Doc Writer::save(String documentPath)**
  Creates an Doc object with the content in the Writer and saves it in the specified documentPath

setOutputAttribute

- **void Writer::setOutputAttribute(String sAttributeName, String sAttribueValue)**
  Set an attribute of this writer - which becomes an attribute of the document this writer is flushed into, if any

setOutputName

- **void Writer::setOutputName(String sName)**
  Set the name of this writer - which becomes the name of the document this writer is flushed into, if any

write

- **void Writer::write(Object o)**
  Writes o as a string into this writer

writeBinaryFile

- **void writeBinaryFile(String sDestFileName, String sOrigFilePath)**
  Pipes the docstore file represented by sOrigFilePath into a new Doc of name sDestFileName in the directory of the current transaction instance

writeDoc

- **void Writer::writeDoc(Doc doc)**
  Appends doc as a string into this writer

writeFile

- **void Writer::writeFile(String sFilePath)**
  Pipes the dostore file represented sFilePath into this writer

writeFileUsingOut

- **void Writer::writeFileUsingOut(Writer w)**
  Pipes w into this writer

writeFileUsingReader

- **void Writer::writeFileUsingReader(Reader r)**
  Pipes r into this writer

writeln

- **void Writer::writeln(Object o)**
  Writes o as a string and appends a new line to it into this writer

zip

unzip

- **Boolean unzip(String srcPath, String dstPath)**
  Unzip zip file given by srcPath into directory given by dstPath

zip

- **Boolean zip(String srcPath, String dstPath[,Stiring[] filesList)**
  Zips files under directory given by srcPath and creates zip file given by dstPath

zip_archive

addCtgFile

- **Boolean ZipArchive::addCtgFile(String sFileName [, Boolean bUpperCaseName])**
  Use to add a supplier ctg file (including images) to a zip archive

closeZipArchive

- **void ZipArchive::closeZipArchive([Boolean deleteAfterDistribution])**
  Use to close a zip archive and upload to the docstore for future distributions. By default, the archive is deleted after the distribution, unless 'deleteAfterDistribution' is false.

getCtgFileDiffStatus

- **Boolean getCtgFileDiffStatus(String sFileName)**
  Returns true or false to indicate whether or not the file was modified between the two versions selected for differences syndication

getCtgFileExists

- **Boolean getCtgFileExists(String sFileName)**
  Returns true or false to indicate whether the physical file really exists

new$ZipArchive

- **new ZipArchive(String sFileName)**
  Returns a new zip archive with the given file name