init

Managing readers setAttributeReaders:
getAttributeReaders:

Managing text stream types understandsType:
addSourceType:
removeSourceType:

Managing parse options setMinimumWeight:
minimumWeight
setPercentPassed:
percentPassed
setWeightingDomain:
weightingDomain
setWeightingType:
weightingType

Parsing text parseFile:ofType:
parseStream:ofType:
analyzeFile:ofType:
analyzeStream:ofType:
reset

addSourceType:(const char *)aType

Records the Pasteboard type or file extension aType as one of the types for which the IXAttributeParser will respond YES when sent an understandsType: message, and which the IXAttributeParser will attempt to parse. If an IXAttributeParser has had no source types added, or has had all source types removed with removeSourceType:, it acts as though it understands any type, and will parse any file or stream. Returns self.

removeSourceType:, understandsType:, analyzeFile:ofType:, analyzeStream:ofType:, parseFile:ofType:, parseStream: ofType:, Pasteboard class of the Application Kit

(NXStream *)analyzeFile:(const char *)filename ofType:(const char *)aType

(NXStream *)analyzeStream:(NXStream *)stream ofType:(const char *)aType

Parses stream, and returns the contents of stream in Attribute Reader Format as read by the IXAttri IXAttributeReaders. If the IXAttributeParser doesn't understand the pasteboard type aType, this n Otherwise, aType is used to determine whether stream is in a parsable format (one of ARF, RTF, locate a filter service that can convert the contents of stream. Streams that can't be converted into parsed as though they contained ASCII text, unless a significant amount of the text is nonprintable stream isn't parsed.

analyzeFile:ofType:, parseStream:ofType:, parseFile:ofType:, understandsType:, addSourceTyp Format (Other Features section), Pasteboard class of the Application Kit

getAttributeReaders:(List *)aList

Empties aList, fills it with the IXAttributeReaders used by the IXAttributeParser, and returns it by of this message may free the List, but not its contents. Returns self.

setAttributeReaders:

init

Initializes a newly created IXAttributeParser, setting the percent passed to 100 and the weighting t IX_NoWeighting. Returns self.

setPercentPassed:, setWeightingType:

(unsigned int)minimumWeight

Returns the minimum weight required for a lexeme to be included in the attribute/value list.

setMinimumWeight:, percentPassed

parseFile:(const char *)filename ofType:(const char *)aType

Parses the contents of filename, and returns self. If the IXAttributeParser doesn't understand the ty returns nil. Otherwise, aType is used to determine whether the contents of filename are in a parsa ARF, RTF, or ASCII), or if not, to locate a filter service that can convert the contents of filename. converted into a parsable format are parsed as though they contained ASCII text, unless a significa nonprintable, in which case the stream isn't parsed.

parseStream:ofType:, analyzeFile:ofType:, analyzeStream:ofType:, understandsType:, addSour class of the Application Kit

parseStream:(NXStream *)stream ofType:(const char *)aType

(unsigned int)percentPassed

Returns the percentage of the lexemes for each attribute that will be included in the result of a pars
weight puts it at this percentile or higher will be included.

setPercentPassed:, minimumWeight

removeSourceType:(const char *)aType

Removes the pasteboard type or file extension aType from the IXAttributeParser's list of understoo
IXAttributeParser will respond NO to subsequent understandsType: messages with aType as the ar
parse files or streams of that type. Returns self.

addSourceType:, understandsType:, Pasteboard class of the Application Kit

reset

Clears the state built up by parsing a file or stream, preparing the IXAttributeParser to analyze a di
is possible to combine multiple streams or files by parsing them in sequence without resetting the I
which case the results accumulate in the attribute/value list. Returns self.

analyzeFile:ofType:, analyzeStream:ofType:, parseFile:ofType:, parseStream:ofType:

setAttributeReaders:(List *)aList

Establishes the objects in aList as the IXAttributeReaders used by the IXAttributeParser, and frees
of IXAttributeReaders that the IXAttributeParser will no longer use. The List must contain instanc
IXAttributeReader or a subclass. Readers will be used on a stream of text in the order they appear
self.

getAttributeReaders:

setMinimumWeight:(unsigned int)anInt

Sets the minimum weight required for inclusion in the parse result. For example, setting the minin
causes all lexemes with weight less than 10 to be dropped from the result of a parse. Returns self.

The IXAttributeParser uses only one of minimum weight or percent passed. If the minimum weigh
passed is reset to 100 if the percent passed is set, the minimum weight is reset to 0.

minimumWeight, setPercentPassed:

setPercentPassed:(unsigned int)anInt

Sets the percentage of lexemes for a given attribute that will be included in the result of a parse. A
weight puts it at this percentile or higher will be included. For example, setting this value to 25 wo

setWeightingDomain:(IXWeightingDomain *)aDomain

Sets the weighting domain used by the IXAttributeParser to aDomain, and returns self. The weigh
assign peculiarity weights to lexemes for a given attribute the frequency of the lexeme within the a
the frequency of the lexeme in the domain to give the lexeme's peculiarity, and the result is norma
square root. This is only done when the IXAttributeParser's weighting type is IX_PeculiarityWeig

weightingDomain, setWeightingType:

setWeightingType:(IXWeightingType)anInt

Sets the weighting type used by the IXAttributeParser to anInt and returns self. The weighting typ
how to calculate lexeme weights, and may be one of the following values:

IX_NoWeighting
IX_AbsoluteWeighting
IX_FrequencyWeighting
IX_PeculiarityWeighting

IX_NoWeighting means that all lexemes are assigned a weight of 0. With IX_AbsoluteWeighting
assigned a weight equal to the number of times it occurs within the attribute. IX_FrequencyWeigh
lexeme being weighted by relative frequency of occurrence: the number of times it occurs in the a
total number of lexemes in the attribute. IX_PeculiarityWeighting uses a weighting domain to calc
relative to some large body of text the final weight of a lexeme is calculated by taking the square r
the attribute divided by its frequency in the domain. IX_PeculiarityWeighting is useful for loweri
lexemes that are common in a particular set of texts.

weightingType, setWeightingDomain:

(BOOL)understandsType:(const char *)aType

Returns YES if the IXAttributeParser will parse files of the pasteboard type or file extension aTyp
types have been added with addSourceType:, or if all types added have been removed with remove
method always returns YES.

addSourceType:, removeSourceType:, Pasteboard class of the Application Kit

(IXWeightingDomain *)weightingDomain

Returns the weighting domain used by the IXAttributeParser, or nil if there is none.

setWeightingDomain:, setWeightingType:

(IXWeightingType)weightingType

Returns the weighting type used by the IXAttributeParser. See setWeightingType: for a list of the
their meanings.

setWeightingType:, setWeightingDomain: