

# **Integrated Modeling for Road Conditions Prediction: System Design Description**

October 31, 2022

Final



U.S. Department of Transportation  
**Federal Highway Administration**

## **Notice**

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The U.S. Government assumes no liability for the use of the information contained in this document.

The U.S. Government does not endorse products or manufacturers. Trademarks or manufacturers' names appear in this report only because they are considered essential to the objective of the document.

## **Quality Assurance Statement**

The Federal Highway Administration (FHWA) provides high-quality information to serve government, industry, and the public in a manner that promotes public understanding. Standards and policies are used to ensure and maximize the quality, objectivity, utility, and integrity of its information. The FHWA periodically reviews quality issues and adjusts its programs and processes to ensure continuous quality improvement.

SI* (MODERN METRIC) CONVERSION FACTORS				
APPROXIMATE CONVERSIONS TO SI UNITS				
Symbol	When You Know	Multiply By	To Find	Symbol
		<b>LENGTH</b>		
in	inches	25.4	millimeters	mm
ft	feet	0.305	meters	m
yd	yards	0.914	meters	m
mi	miles	1.61	kilometers	km
		<b>AREA</b>		
in <sup>2</sup>	square inches	645.2	square millimeters	mm <sup>2</sup>
ft <sup>2</sup>	square feet	0.093	square meters	m <sup>2</sup>
yd <sup>2</sup>	square yard	0.836	square meters	m <sup>2</sup>
ac	acres	0.405	hectares	ha
mi <sup>2</sup>	square miles	2.59	square kilometers	km <sup>2</sup>
		<b>VOLUME</b>		
fl oz	fluid ounces	29.57	milliliters	mL
gal	gallons	3.785	liters	L
ft <sup>3</sup>	cubic feet	0.028	cubic meters	m <sup>3</sup>
yd <sup>3</sup>	cubic yards	0.765	cubic meters	m <sup>3</sup>
NOTE: volumes greater than 1,000 L shall be shown in m <sup>3</sup>				
		<b>MASS</b>		
oz	ounces	28.35	grams	g
lb	pounds	0.454	kilograms	kg
T	short tons (2,000 lb)	0.907	megagrams (or "metric ton")	Mg (or "t")
<b>TEMPERATURE (exact degrees)</b>				
°F	Fahrenheit	5 (F-32)/9 or (F-32)/1.8	Celsius	°C
<b>ILLUMINATION</b>				
fc	foot-candles	10.76	lux	lx
fl	foot-Lamberts	3.426	candela/m <sup>2</sup>	cd/m <sup>2</sup>
<b>FORCE and PRESSURE or STRESS</b>				
lbf	poundforce	4.45	newtons	N
lbf/in <sup>2</sup>	poundforce per square inch	6.89	kilopascals	kPa
APPROXIMATE CONVERSIONS FROM SI UNITS				
Symbol	When You Know	Multiply By	To Find	Symbol
		<b>LENGTH</b>		
mm	millimeters	0.039	inches	in
m	meters	3.28	feet	ft
m	meters	1.09	yards	yd
km	kilometers	0.621	miles	mi
		<b>AREA</b>		
mm <sup>2</sup>	square millimeters	0.0016	square inches	in <sup>2</sup>
m <sup>2</sup>	square meters	10.764	square feet	ft <sup>2</sup>
m <sup>2</sup>	square meters	1.195	square yards	yd <sup>2</sup>
ha	hectares	2.47	acres	ac
km <sup>2</sup>	square kilometers	0.386	square miles	mi <sup>2</sup>
		<b>VOLUME</b>		
mL	milliliters	0.034	fluid ounces	fl oz
L	liters	0.264	gallons	gal
m <sup>3</sup>	cubic meters	35.314	cubic feet	ft <sup>3</sup>
m <sup>3</sup>	cubic meters	1.307	cubic yards	yd <sup>3</sup>
		<b>MASS</b>		
g	grams	0.035	ounces	oz
kg	kilograms	2.202	pounds	lb
Mg (or "t")	megagrams (or "metric ton")	1.103	short tons (2,000 lb)	T
<b>TEMPERATURE (exact degrees)</b>				
°C	Celsius	1.8C+32	Fahrenheit	°F
<b>ILLUMINATION</b>				
lx	lux	0.0929	foot-candles	fc
cd/m <sup>2</sup>	candela/m <sup>2</sup>	0.2919	foot-Lamberts	fl
<b>FORCE and PRESSURE or STRESS</b>				
N	newtons	2.225	poundforce	lbf
kPa	kilopascals	0.145	poundforce per square inch	lbf/in <sup>2</sup>

\*SI is the symbol for International System of Units. Appropriate rounding should be made to comply with Section 4 of ASTM E380.  
(Revised March 2003)



## TABLE OF CONTENTS

<b>EXECUTIVE SUMMARY .....</b>	<b>1</b>
<b>CHAPTER 1. INTRODUCTION .....</b>	<b>3</b>
<b>BACKGROUND .....</b>	<b>3</b>
<b>PURPOSE .....</b>	<b>3</b>
<b>SCOPE .....</b>	<b>3</b>
<b>DOCUMENT OVERVIEW .....</b>	<b>4</b>
<b>CHAPTER 2. GENERAL DESCRIPTION .....</b>	<b>5</b>
<b>SYSTEM PERSPECTIVE .....</b>	<b>5</b>
<b>FUSER ROLES .....</b>	<b>6</b>
<b>CHAPTER 3. DESIGN DESCRIPTION .....</b>	<b>7</b>
<b>COLLECT .....</b>	<b>10</b>
Collector.....	10
<b>STORE .....</b>	<b>14</b>
FileCache .....	14
ObsView .....	19
ProjProfiles .....	19
<b>SYSTEM .....</b>	<b>20</b>
Directory .....	20
ObsType.....	21
Configuration [Config] .....	22
Scheduling.....	22
ExtMapping.....	22
<b>COMPUTATION .....</b>	<b>23</b>
Alerts .....	23
DataAssimilation.....	24
EventComp .....	24
InrixComp .....	25
Precipitation Category [PcCat] .....	25
LAc2cDetectors .....	26
<b>FORECAST .....</b>	<b>26</b>
Machine Learning-based Prediction .....	26
Machine Learning-based Prediction for Hurricane Traffic .....	31
MLPPredict .....	34
MLPUpdate .....	35
MLPHurricane .....	35
MLPProcess .....	36
METRo.....	36
MetroProcess .....	37
WayNetworks .....	37
DEM .....	38
Mercator .....	39
OsmBz2ToBin .....	39

<b>OsmBinParser .....</b>	<b>39</b>
<b>WEB .....</b>	<b>40</b>
SecureBaseBlock .....	40
NetworkGeneration .....	40
SessMgr .....	41
Scenarios .....	41
Subscriptions .....	42
LayerServlet .....	42
TileCache .....	43
TileServlet.....	43
<b>WEB PAGES.....</b>	<b>44</b>
LogOn.....	44
Map.....	44
Create Scenario .....	45
View Scenarios .....	46
Create Report .....	46
View Reports .....	46
Network.....	47
<b>SYSTEM COMPUTING INFRASTRUCTURE .....</b>	<b>47</b>
<b>CHAPTER 4. REFERENCES .....</b>	<b>49</b>
<b>APPENDIX A. INPUT DATA AVAILABILITY .....</b>	<b>51</b>
<b>APPENDIX B. LAYER DEFINITIONS.....</b>	<b>53</b>
<b>APPENDIX C. OBSERVATION TYPE DEFINITIONS .....</b>	<b>61</b>
<b>APPENDIX D. ALERT DEFINITIONS.....</b>	<b>76</b>
<b>APPENDIX E. CLASS DOCUMENTATION .....</b>	<b>78</b>
IMRCP.COLLECT.AHPS CLASS REFERENCE .....	78
Public Member Functions .....	78
Additional Inherited Members .....	78
Detailed Description.....	78
Constructor & Destructor Documentation.....	78
Member Function Documentation .....	78
IMRCP.STORE.AHPSSTORE CLASS REFERENCE.....	80
Public Member Functions .....	80
Protected Member Functions.....	80
Additional Inherited Members .....	80
Detailed Description.....	80
Member Function Documentation .....	80
IMRCP.STORE.AHPSWRAPPER CLASS REFERENCE .....	80
Public Member Functions .....	80
Public Attributes .....	81
Additional Inherited Members .....	81
Detailed Description.....	81
Member Function Documentation .....	81
Member Data Documentation .....	81

<b>IMRCP.COMP.ALERTCONDITION CLASS REFERENCE .....</b>	<b>81</b>
Public Member Functions .....	81
Public Attributes .....	82
Detailed Description.....	82
Constructor & Destructor Documentation.....	82
Member Function Documentation .....	82
Member Data Documentation .....	82
<b>IMRCP.COMP.ALERTS CLASS REFERENCE .....</b>	<b>83</b>
Public Member Functions .....	83
Additional Inherited Members .....	83
Detailed Description.....	83
Member Function Documentation .....	83
<b>IMRCP.STORE.ALERTSCSVWRAPPER CLASS REFERENCE.....</b>	<b>84</b>
Public Member Functions .....	84
Additional Inherited Members .....	84
Detailed Description.....	84
Constructor & Destructor Documentation.....	84
Member Function Documentation .....	85
<b>IMRCP.STORE.ALERTSSTORE CLASS REFERENCE .....</b>	<b>85</b>
Public Member Functions .....	85
Protected Member Functions.....	85
Additional Inherited Members .....	85
Detailed Description.....	85
Member Function Documentation .....	85
<b>IMRCP.WEB.LAYERS.AREALAYERSSERVLET CLASS REFERENCE .....</b>	<b>86</b>
Public Member Functions .....	86
Protected Member Functions.....	86
Additional Inherited Members .....	86
Detailed Description.....	86
Member Function Documentation .....	86
<b>IMRCP.SYSTEM.ARRAYS CLASS REFERENCE .....</b>	<b>87</b>
Static Public Member Functions .....	87
Detailed Description.....	87
Member Function Documentation .....	87
<b>IMRCP.SYSTEM.ASYNCQ&lt; T &gt; CLASS TEMPLATE REFERENCE .....</b>	<b>92</b>
Public Member Functions .....	92
Protected Member Functions.....	92
Detailed Description.....	93
Constructor & Destructor Documentation.....	93
Member Function Documentation .....	93
<b>IMRCP.SYSTEM.BASEBLOCK CLASS REFERENCE .....</b>	<b>94</b>
Public Member Functions .....	94
Public Attributes .....	94
Protected Member Functions.....	94
Protected Attributes.....	95
Additional Inherited Members .....	95

<b>Detailed Description.....</b>	<b>95</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>95</b>
<b>Member Function Documentation .....</b>	<b>95</b>
<b>Member Data Documentation .....</b>	<b>100</b>
<b>IMRCP.STORE.BINOBSSSTORE CLASS REFERENCE .....</b>	<b>101</b>
<b>Public Member Functions .....</b>	<b>101</b>
<b>Protected Member Functions.....</b>	<b>101</b>
<b>Additional Inherited Members .....</b>	<b>101</b>
<b>Detailed Description.....</b>	<b>101</b>
<b>Member Function Documentation .....</b>	<b>101</b>
<b>IMRCP.SYSTEM.BLOCKCONFIG CLASS REFERENCE .....</b>	<b>102</b>
<b>Public Member Functions .....</b>	<b>102</b>
<b>Detailed Description.....</b>	<b>102</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>102</b>
<b>Member Function Documentation .....</b>	<b>102</b>
<b>IMRCP.COLLECT.BLUETOAD CLASS REFERENCE .....</b>	<b>104</b>
<b>Public Member Functions .....</b>	<b>104</b>
<b>Additional Inherited Members .....</b>	<b>104</b>
<b>Detailed Description.....</b>	<b>104</b>
<b>Member Function Documentation .....</b>	<b>104</b>
<b>IMRCP.SYSTEM.BUFFEREDINSTREAM CLASS REFERENCE.....</b>	<b>105</b>
<b>Public Member Functions .....</b>	<b>105</b>
<b>Static Protected Attributes.....</b>	<b>105</b>
<b>Detailed Description.....</b>	<b>105</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>105</b>
<b>Member Data Documentation .....</b>	<b>105</b>
<b>IMRCP.STORE.BYTEOBSENTRYDATA CLASS REFERENCE.....</b>	<b>106</b>
<b>Public Member Functions .....</b>	<b>106</b>
<b>Additional Inherited Members .....</b>	<b>106</b>
<b>Detailed Description.....</b>	<b>106</b>
<b>Member Function Documentation .....</b>	<b>106</b>
<b>IMRCP.COLLECT.CAP CLASS REFERENCE.....</b>	<b>106</b>
<b>Public Member Functions .....</b>	<b>106</b>
<b>Additional Inherited Members .....</b>	<b>107</b>
<b>Detailed Description.....</b>	<b>107</b>
<b>Member Function Documentation .....</b>	<b>107</b>
<b>IMRCP.STORE.CAPOBS CLASS REFERENCE.....</b>	<b>107</b>
<b>Public Member Functions .....</b>	<b>107</b>
<b>Public Attributes .....</b>	<b>107</b>
<b>Additional Inherited Members .....</b>	<b>108</b>
<b>Detailed Description.....</b>	<b>108</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>108</b>
<b>Member Function Documentation .....</b>	<b>109</b>
<b>Member Data Documentation .....</b>	<b>109</b>
<b>IMRCP.STORE.CAPSTORE CLASS REFERENCE .....</b>	<b>109</b>
<b>Public Member Functions .....</b>	<b>109</b>

<b>Protected Member Functions.....</b>	<b>109</b>
<b>Additional Inherited Members .....</b>	<b>110</b>
<b>Detailed Description.....</b>	<b>110</b>
<b>Member Function Documentation .....</b>	<b>110</b>
<b>IMRCP.STORE.CAPWRAPPER CLASS REFERENCE .....</b>	<b>110</b>
<b>Public Member Functions .....</b>	<b>110</b>
<b>Additional Inherited Members .....</b>	<b>110</b>
<b>Detailed Description.....</b>	<b>111</b>
<b>Member Function Documentation .....</b>	<b>111</b>
<b>IMRCP.WEB.CENGROUP CLASS REFERENCE .....</b>	<b>111</b>
<b>Public Member Functions .....</b>	<b>111</b>
<b>Detailed Description.....</b>	<b>111</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>111</b>
<b>Member Function Documentation .....</b>	<b>112</b>
<b>IMRCP.WEB.CENPLACE CLASS REFERENCE .....</b>	<b>112</b>
<b>Public Member Functions .....</b>	<b>112</b>
<b>Public Attributes .....</b>	<b>112</b>
<b>Detailed Description.....</b>	<b>112</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>112</b>
<b>Member Function Documentation .....</b>	<b>113</b>
<b>Member Data Documentation .....</b>	<b>113</b>
<b>IMRCP.COLLECT.COLLECTOR CLASS REFERENCE .....</b>	<b>113</b>
<b>Public Member Functions .....</b>	<b>113</b>
<b>Protected Member Functions.....</b>	<b>113</b>
<b>Protected Attributes.....</b>	<b>113</b>
<b>Additional Inherited Members .....</b>	<b>114</b>
<b>Detailed Description.....</b>	<b>114</b>
<b>Member Function Documentation .....</b>	<b>114</b>
<b>Member Data Documentation .....</b>	<b>115</b>
<b>IMRCP.STORE.NHCWRAPPER.CONEPARSER CLASS REFERENCE .....</b>	<b>115</b>
<b>Protected Member Functions.....</b>	<b>115</b>
<b>Detailed Description.....</b>	<b>115</b>
<b>Member Function Documentation .....</b>	<b>116</b>
<b>IMRCP.SYSTEM.CONFIG CLASS REFERENCE .....</b>	<b>116</b>
<b>Public Member Functions .....</b>	<b>116</b>
<b>Static Public Member Functions .....</b>	<b>116</b>
<b>Detailed Description.....</b>	<b>116</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>116</b>
<b>Member Function Documentation .....</b>	<b>117</b>
<b>IMRCP.SYSTEM.CSVREADER CLASS REFERENCE .....</b>	<b>118</b>
<b>Public Member Functions .....</b>	<b>118</b>
<b>Protected Attributes.....</b>	<b>119</b>
<b>Static Protected Attributes.....</b>	<b>119</b>
<b>Detailed Description.....</b>	<b>119</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>119</b>
<b>Member Function Documentation .....</b>	<b>120</b>

<b>Member Data Documentation .....</b>	<b>122</b>
<b>IMRCP.STORE.CSVSTORE CLASS REFERENCE .....</b>	<b>123</b>
<b>Public Member Functions .....</b>	<b>123</b>
<b>Protected Member Functions.....</b>	<b>123</b>
<b>Additional Inherited Members .....</b>	<b>123</b>
<b>Detailed Description.....</b>	<b>123</b>
<b>Member Function Documentation .....</b>	<b>123</b>
<b>IMRCP.STORE.CSVWRAPPER CLASS REFERENCE.....</b>	<b>125</b>
<b>Public Member Functions .....</b>	<b>125</b>
<b>Additional Inherited Members .....</b>	<b>125</b>
<b>Detailed Description.....</b>	<b>125</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>125</b>
<b>Member Function Documentation .....</b>	<b>125</b>
<b>IMRCP.COMP.DATAASSIMILATION CLASS REFERENCE.....</b>	<b>126</b>
<b>Public Member Functions .....</b>	<b>126</b>
<b>Protected Member Functions.....</b>	<b>126</b>
<b>Protected Attributes.....</b>	<b>126</b>
<b>Additional Inherited Members .....</b>	<b>126</b>
<b>Detailed Description.....</b>	<b>126</b>
<b>Member Function Documentation .....</b>	<b>126</b>
<b>Member Data Documentation .....</b>	<b>128</b>
<b>IMRCP.WEB.TILES.DATAOBSTILECACHE CLASS REFERENCE .....</b>	<b>129</b>
<b>Protected Member Functions.....</b>	<b>129</b>
<b>Additional Inherited Members .....</b>	<b>129</b>
<b>Detailed Description.....</b>	<b>129</b>
<b>Member Function Documentation .....</b>	<b>130</b>
<b>IMRCP.STORE.DATAOBSWRAPPER CLASS REFERENCE .....</b>	<b>130</b>
<b>Public Member Functions .....</b>	<b>130</b>
<b>Additional Inherited Members .....</b>	<b>130</b>
<b>Detailed Description.....</b>	<b>130</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>130</b>
<b>Member Function Documentation .....</b>	<b>131</b>
<b>IMRCP.STORE.GRIB.DATAREP CLASS REFERENCE .....</b>	<b>132</b>
<b>Public Member Functions .....</b>	<b>132</b>
<b>Static Public Member Functions .....</b>	<b>132</b>
<b>Public Attributes .....</b>	<b>132</b>
<b>Detailed Description.....</b>	<b>132</b>
<b>Member Function Documentation .....</b>	<b>133</b>
<b>Member Data Documentation .....</b>	<b>134</b>
<b>IMRCP.STORE.GRIB.DATAREPPNG CLASS REFERENCE.....</b>	<b>134</b>
<b>Public Member Functions .....</b>	<b>134</b>
<b>Additional Inherited Members .....</b>	<b>134</b>
<b>Detailed Description.....</b>	<b>134</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>135</b>
<b>Member Function Documentation .....</b>	<b>135</b>
<b>IMRCP.SYSTEM.DBF.DBFDOUBLE CLASS REFERENCE .....</b>	<b>135</b>

Public Member Functions .....	135
Additional Inherited Members .....	135
Detailed Description.....	136
Constructor & Destructor Documentation.....	136
Member Function Documentation .....	136
<b>IMRCP.SYSTEM.DBF.DBFFIELD CLASS REFERENCE.....</b>	<b>137</b>
Public Member Functions .....	137
Static Public Member Functions .....	137
Static Public Attributes .....	137
Protected Member Functions.....	137
Protected Attributes.....	137
Detailed Description.....	138
Constructor & Destructor Documentation.....	138
Member Function Documentation .....	138
Member Data Documentation .....	140
<b>IMRCP.SYSTEM.DBF.DBFLONG CLASS REFERENCE .....</b>	<b>140</b>
Public Member Functions .....	140
Additional Inherited Members .....	141
Detailed Description.....	141
Constructor & Destructor Documentation.....	141
Member Function Documentation .....	141
<b>IMRCP.SYSTEM.DBF.DBFRESULTSET CLASS REFERENCE .....</b>	<b>142</b>
Public Member Functions .....	142
Detailed Description.....	146
Constructor & Destructor Documentation.....	146
Member Function Documentation .....	147
<b>IMRCP.SYSTEM.DBF.DBFSTRING CLASS REFERENCE.....</b>	<b>148</b>
Public Member Functions .....	148
Additional Inherited Members .....	148
Detailed Description.....	148
Constructor & Destructor Documentation.....	149
Member Function Documentation .....	149
<b>IMRCP.FORECAST.MLP.MLPHURRICANE.DELEGATE CLASS REFERENCE</b>	<b>150</b>
Public Member Functions .....	150
Detailed Description.....	150
Member Function Documentation .....	150
<b>IMRCP.GEOSRV.DEM CLASS REFERENCE .....</b>	<b>151</b>
Public Member Functions .....	151
Additional Inherited Members .....	151
Detailed Description.....	151
Member Function Documentation .....	151
<b>IMRCP.SYSTEM.DIRECTORY CLASS REFERENCE.....</b>	<b>152</b>
Classes .....	152
Public Member Functions .....	152
Static Public Member Functions .....	152
Static Public Attributes .....	152

<b>Detailed Description.....</b>	<b>192</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>193</b>
<b>Member Function Documentation .....</b>	<b>193</b>
<b>Member Data Documentation .....</b>	<b>195</b>
<b>IMRCP.FORECAST.MDSS.DOMETROWRAPPER CLASS REFERENCE .....</b>	<b>195</b>
<b>Public Member Functions .....</b>	<b>195</b>
<b>Public Attributes .....</b>	<b>196</b>
<b>Static Public Attributes .....</b>	<b>196</b>
<b>Detailed Description.....</b>	<b>196</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>196</b>
<b>Member Function Documentation .....</b>	<b>196</b>
<b>Member Data Documentation .....</b>	<b>198</b>
<b>IMRCP.SYSTEM.EMAIL CLASS REFERENCE .....</b>	<b>198</b>
<b>Public Member Functions .....</b>	<b>198</b>
<b>Public Attributes .....</b>	<b>199</b>
<b>Detailed Description.....</b>	<b>199</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>199</b>
<b>Member Data Documentation .....</b>	<b>199</b>
<b>IMRCP.SYSTEM.EMAILS CLASS REFERENCE .....</b>	<b>200</b>
<b>Static Public Member Functions .....</b>	<b>200</b>
<b>Detailed Description.....</b>	<b>200</b>
<b>Member Function Documentation .....</b>	<b>200</b>
<b>IMRCP.STORE.ENTRYDATA CLASS REFERENCE .....</b>	<b>200</b>
<b>Public Member Functions .....</b>	<b>200</b>
<b>Public Attributes .....</b>	<b>200</b>
<b>Protected Member Functions.....</b>	<b>201</b>
<b>Protected Attributes.....</b>	<b>201</b>
<b>Detailed Description.....</b>	<b>201</b>
<b>Member Function Documentation .....</b>	<b>201</b>
<b>Member Data Documentation .....</b>	<b>203</b>
<b>IMRCP.COMP.EVENTCOMP CLASS REFERENCE .....</b>	<b>203</b>
<b>Public Member Functions .....</b>	<b>203</b>
<b>Protected Member Functions.....</b>	<b>203</b>
<b>Protected Attributes.....</b>	<b>203</b>
<b>Static Protected Attributes.....</b>	<b>203</b>
<b>Additional Inherited Members .....</b>	<b>203</b>
<b>Detailed Description.....</b>	<b>204</b>
<b>Member Function Documentation .....</b>	<b>204</b>
<b>Member Data Documentation .....</b>	<b>205</b>
<b>IMRCP.STORE.EVENTOBS CLASS REFERENCE .....</b>	<b>205</b>
<b>Public Member Functions .....</b>	<b>205</b>
<b>Public Attributes .....</b>	<b>205</b>
<b>Static Public Attributes .....</b>	<b>206</b>
<b>Detailed Description.....</b>	<b>206</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>206</b>
<b>Member Function Documentation .....</b>	<b>207</b>

<b>Member Data Documentation .....</b>	<b>208</b>
<b>IMRCP.SYSTEMEXTMAPPING CLASS REFERENCE.....</b>	<b>209</b>
<b>Public Member Functions .....</b>	<b>209</b>
<b>Additional Inherited Members .....</b>	<b>209</b>
<b>Detailed Description.....</b>	<b>209</b>
<b>Member Function Documentation .....</b>	<b>209</b>
<b>IMRCP.STOREFILECACHE CLASS REFERENCE .....</b>	<b>210</b>
<b>Public Member Functions .....</b>	<b>210</b>
<b>Public Attributes .....</b>	<b>210</b>
<b>Static Public Attributes .....</b>	<b>210</b>
<b>Protected Member Functions.....</b>	<b>210</b>
<b>Protected Attributes.....</b>	<b>210</b>
<b>Static Protected Attributes.....</b>	<b>211</b>
<b>Detailed Description.....</b>	<b>211</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>211</b>
<b>Member Function Documentation .....</b>	<b>211</b>
<b>Member Data Documentation .....</b>	<b>214</b>
<b>IMRCP.SYSTEMFILENAMEFORMATTER CLASS REFERENCE .....</b>	<b>217</b>
<b>Public Member Functions .....</b>	<b>217</b>
<b>Protected Member Functions.....</b>	<b>217</b>
<b>Detailed Description.....</b>	<b>217</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>218</b>
<b>Member Function Documentation .....</b>	<b>218</b>
<b>IMRCP.SYSTEMFILEUTIL CLASS REFERENCE .....</b>	<b>222</b>
<b>Static Public Attributes .....</b>	<b>222</b>
<b>Detailed Description.....</b>	<b>222</b>
<b>Member Data Documentation .....</b>	<b>222</b>
<b>IMRCP.STOREFILEWRAPPER CLASS REFERENCE .....</b>	<b>223</b>
<b>Public Member Functions .....</b>	<b>223</b>
<b>Public Attributes .....</b>	<b>223</b>
<b>Static Public Attributes .....</b>	<b>223</b>
<b>Protected Attributes.....</b>	<b>223</b>
<b>Detailed Description.....</b>	<b>223</b>
<b>Member Function Documentation .....</b>	<b>223</b>
<b>Member Data Documentation .....</b>	<b>225</b>
<b>IMRCP.STOREFLOATOBSENTRYDATA CLASS REFERENCE.....</b>	<b>225</b>
<b>Public Member Functions .....</b>	<b>225</b>
<b>Additional Inherited Members .....</b>	<b>226</b>
<b>Detailed Description.....</b>	<b>226</b>
<b>Member Function Documentation .....</b>	<b>226</b>
<b>IMRCP.STOREFLOODMAPPING CLASS REFERENCE .....</b>	<b>226</b>
<b>Static Public Attributes .....</b>	<b>226</b>
<b>Detailed Description.....</b>	<b>226</b>
<b>Member Data Documentation .....</b>	<b>227</b>
<b>IMRCP.COLLECTFLOODSTAGEMETADATA CLASS REFERENCE.....</b>	<b>227</b>
<b>Public Member Functions .....</b>	<b>227</b>

<b>Public Attributes .....</b>	<b>227</b>
<b>Detailed Description.....</b>	<b>227</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>227</b>
<b>Member Function Documentation .....</b>	<b>228</b>
<b>    Member Data Documentation .....</b>	<b>228</b>
<b>IMRCP.WEB.REPORTSUBSCRIPTION FORMAT ENUM REFERENCE .....</b>	<b>228</b>
<b>    Public Attributes .....</b>	<b>228</b>
<b>    Detailed Description.....</b>	<b>228</b>
<b>    Member Data Documentation .....</b>	<b>229</b>
<b>IMRCP.COLLECT.GEOTAB CLASS REFERENCE.....</b>	<b>229</b>
<b>    Public Member Functions .....</b>	<b>229</b>
<b>    Static Public Attributes .....</b>	<b>229</b>
<b>    Additional Inherited Members .....</b>	<b>229</b>
<b>    Detailed Description.....</b>	<b>229</b>
<b>    Member Function Documentation .....</b>	<b>229</b>
<b>    Member Data Documentation .....</b>	<b>230</b>
<b>IMRCP.STORE.GEOTABSTORE CLASS REFERENCE .....</b>	<b>230</b>
<b>    Public Member Functions .....</b>	<b>230</b>
<b>    Protected Member Functions.....</b>	<b>230</b>
<b>    Additional Inherited Members .....</b>	<b>230</b>
<b>    Detailed Description.....</b>	<b>230</b>
<b>    Member Function Documentation .....</b>	<b>231</b>
<b>IMRCP.STORE.GEOTABWRAPPER CLASS REFERENCE .....</b>	<b>231</b>
<b>    Public Member Functions .....</b>	<b>231</b>
<b>    Additional Inherited Members .....</b>	<b>231</b>
<b>    Detailed Description.....</b>	<b>231</b>
<b>    Constructor &amp; Destructor Documentation.....</b>	<b>232</b>
<b>    Member Function Documentation .....</b>	<b>232</b>
<b>IMRCP.GEOSRV.GEOUTIL CLASS REFERENCE.....</b>	<b>232</b>
<b>    Static Public Member Functions .....</b>	<b>232</b>
<b>    Static Public Attributes .....</b>	<b>233</b>
<b>    Detailed Description.....</b>	<b>233</b>
<b>    Member Function Documentation .....</b>	<b>233</b>
<b>    Member Data Documentation .....</b>	<b>244</b>
<b>IMRCP.COLLECT.GFS CLASS REFERENCE .....</b>	<b>245</b>
<b>    Public Member Functions .....</b>	<b>245</b>
<b>    Additional Inherited Members .....</b>	<b>245</b>
<b>    Detailed Description.....</b>	<b>245</b>
<b>    Member Function Documentation .....</b>	<b>245</b>
<b>IMRCP.STORE.GRIBENTRYDATA CLASS REFERENCE .....</b>	<b>246</b>
<b>    Public Member Functions .....</b>	<b>246</b>
<b>    Public Attributes .....</b>	<b>246</b>
<b>    Additional Inherited Members .....</b>	<b>246</b>
<b>    Detailed Description.....</b>	<b>246</b>
<b>    Constructor &amp; Destructor Documentation.....</b>	<b>247</b>
<b>    Member Function Documentation .....</b>	<b>247</b>

<b>Member Data Documentation .....</b>	<b>247</b>
<b>IMRCP.STORE.GRIB.GRIBPARAMETER CLASS REFERENCE.....</b>	<b>248</b>
<b>Public Member Functions .....</b>	<b>248</b>
<b>Public Attributes .....</b>	<b>248</b>
<b>Detailed Description.....</b>	<b>248</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>248</b>
<b>Member Function Documentation .....</b>	<b>248</b>
<b>Member Data Documentation .....</b>	<b>248</b>
<b>IMRCP.STORE.GRIBSTORE CLASS REFERENCE .....</b>	<b>249</b>
<b>    Public Member Functions .....</b>	<b>249</b>
<b>    Protected Member Functions.....</b>	<b>249</b>
<b>    Additional Inherited Members .....</b>	<b>249</b>
<b>    Detailed Description.....</b>	<b>249</b>
<b>    Member Function Documentation .....</b>	<b>249</b>
<b>IMRCP.WEB.TILES.GRIBTILECACHE CLASS REFERENCE .....</b>	<b>250</b>
<b>    Public Member Functions .....</b>	<b>250</b>
<b>    Protected Member Functions.....</b>	<b>250</b>
<b>    Additional Inherited Members .....</b>	<b>250</b>
<b>    Detailed Description.....</b>	<b>250</b>
<b>    Member Function Documentation .....</b>	<b>250</b>
<b>IMRCP.STORE.GRIBWRAPPER CLASS REFERENCE .....</b>	<b>251</b>
<b>    Public Member Functions .....</b>	<b>251</b>
<b>    Static Public Member Functions .....</b>	<b>251</b>
<b>    Public Attributes .....</b>	<b>251</b>
<b>    Static Public Attributes .....</b>	<b>251</b>
<b>    Additional Inherited Members .....</b>	<b>251</b>
<b>    Detailed Description.....</b>	<b>251</b>
<b>    Constructor &amp; Destructor Documentation.....</b>	<b>251</b>
<b>    Member Function Documentation .....</b>	<b>252</b>
<b>    Member Data Documentation .....</b>	<b>255</b>
<b>IMRCP.STORE.GRIB.GRID CLASS REFERENCE .....</b>	<b>255</b>
<b>    Public Member Functions .....</b>	<b>255</b>
<b>    Public Attributes .....</b>	<b>256</b>
<b>    Detailed Description.....</b>	<b>256</b>
<b>    Constructor &amp; Destructor Documentation.....</b>	<b>256</b>
<b>    Member Data Documentation .....</b>	<b>256</b>
<b>IMRCP.STORE.GRIDDEDFILEWRAPPER CLASS REFERENCE .....</b>	<b>257</b>
<b>    Public Member Functions .....</b>	<b>257</b>
<b>    Static Protected Member Functions.....</b>	<b>257</b>
<b>    Protected Attributes.....</b>	<b>257</b>
<b>    Additional Inherited Members .....</b>	<b>257</b>
<b>    Detailed Description.....</b>	<b>257</b>
<b>    Constructor &amp; Destructor Documentation.....</b>	<b>257</b>
<b>    Member Function Documentation .....</b>	<b>257</b>
<b>    Member Data Documentation .....</b>	<b>259</b>
<b>IMRCP.GEOSRV.OSM.HASHBUCKET CLASS REFERENCE .....</b>	<b>259</b>

<b>Public Member Functions .....</b>	<b>259</b>
<b>Static Public Member Functions .....</b>	<b>260</b>
<b>Public Attributes .....</b>	<b>260</b>
<b>Detailed Description.....</b>	<b>260</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>260</b>
<b>Member Function Documentation .....</b>	<b>260</b>
<b>Member Data Documentation .....</b>	<b>261</b>
<b>IMRCP.SYSTEM.SHP.HEADER CLASS REFERENCE .....</b>	<b>262</b>
<b>Public Member Functions .....</b>	<b>262</b>
<b>Detailed Description.....</b>	<b>262</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>262</b>
<b>IMRCP.FORECAST.MLP.MLPHURRICANE.HURDATA CLASS REFERENCE .</b>	<b>262</b>
<b>Public Attributes .....</b>	<b>262</b>
<b>Detailed Description.....</b>	<b>262</b>
<b>Member Data Documentation .....</b>	<b>263</b>
<b>IMRCP.FORECAST.MLP.HURHISTDATARECORD CLASS REFERENCE.....</b>	<b>263</b>
<b>Public Member Functions .....</b>	<b>263</b>
<b>Public Attributes .....</b>	<b>263</b>
<b>Static Public Attributes .....</b>	<b>263</b>
<b>Detailed Description.....</b>	<b>263</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>264</b>
<b>Member Function Documentation .....</b>	<b>264</b>
<b>Member Data Documentation .....</b>	<b>264</b>
<b>IMRCP.STORE.HURRICANCENTER CLASS REFERENCE.....</b>	<b>265</b>
<b>Public Member Functions .....</b>	<b>265</b>
<b>Public Attributes .....</b>	<b>265</b>
<b>Detailed Description.....</b>	<b>265</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>266</b>
<b>Member Function Documentation .....</b>	<b>266</b>
<b>Member Data Documentation .....</b>	<b>266</b>
<b>IMRCP.FORECAST.MLP.MLPHURRICANE.HURWORK CLASS REFERENCE</b>	<b>267</b>
<b>Public Member Functions .....</b>	<b>267</b>
<b>Detailed Description.....</b>	<b>267</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>267</b>
<b>Member Function Documentation .....</b>	<b>267</b>
<b>IMRCP.SYSTEM.ID CLASS REFERENCE.....</b>	<b>267</b>
<b>Public Member Functions .....</b>	<b>267</b>
<b>Static Public Member Functions .....</b>	<b>268</b>
<b>Static Public Attributes .....</b>	<b>268</b>
<b>Protected Member Functions.....</b>	<b>268</b>
<b>Protected Attributes.....</b>	<b>268</b>
<b>Static Protected Attributes.....</b>	<b>268</b>
<b>Detailed Description.....</b>	<b>268</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>268</b>
<b>Member Function Documentation .....</b>	<b>269</b>
<b>Member Data Documentation .....</b>	<b>271</b>

<b>IMRCP.SYSTEM.IMRCPBLOCK INTERFACE REFERENCE .....</b>	<b>272</b>
Public Member Functions .....	272
Static Public Attributes .....	272
Detailed Description.....	272
Member Function Documentation .....	273
Member Data Documentation .....	274
<b>IMRCP.STORE.IMRCPCAPRESULTSET CLASS REFERENCE .....</b>	<b>275</b>
Classes .....	275
Public Member Functions .....	275
Protected Attributes.....	275
Additional Inherited Members .....	275
Detailed Description.....	275
Constructor & Destructor Documentation.....	275
Member Data Documentation .....	275
<b>IMRCP.STORE.IMRCPEVENTRESULTSET CLASS REFERENCE .....</b>	<b>275</b>
Classes .....	275
Public Member Functions .....	276
Protected Attributes.....	276
Additional Inherited Members .....	276
Detailed Description.....	276
Constructor & Destructor Documentation.....	276
Member Data Documentation .....	276
<b>IMRCP.STORE.IMRCPOBSRESULTSET CLASS REFERENCE .....</b>	<b>276</b>
Classes .....	276
Public Member Functions .....	276
Protected Attributes.....	276
Additional Inherited Members .....	276
Detailed Description.....	276
Constructor & Destructor Documentation.....	277
Member Data Documentation .....	277
<b>IMRCP.STORE.IMRCRESULTSET&lt; T &gt; CLASS TEMPLATE REFERENCE ...</b>	<b>277</b>
Public Member Functions .....	277
Protected Member Functions.....	281
Protected Attributes.....	281
Detailed Description.....	281
Member Function Documentation .....	281
Member Data Documentation .....	297
<b>IMRCP.COLLECT.INRIX CLASS REFERENCE .....</b>	<b>297</b>
Public Member Functions .....	297
Additional Inherited Members .....	297
Detailed Description.....	298
Member Function Documentation .....	298
<b>IMRCP.COMP.INRIXCOMP CLASS REFERENCE .....</b>	<b>298</b>
Public Member Functions .....	298
Protected Member Functions.....	299
Protected Attributes.....	299

<b>Additional Inherited Members .....</b>	<b>299</b>
<b>Detailed Description.....</b>	<b>299</b>
<b>Member Function Documentation .....</b>	<b>299</b>
<b>Member Data Documentation .....</b>	<b>299</b>
<b>IMRCP.STORE.IMRCPCAPRESULTSET.INTDELEGATE INTERFACE REFERENCE .....</b>	<b>300</b>
<b>Detailed Description.....</b>	<b>300</b>
<b>IMRCP.STORE.IMRCPEVENTRESULTSET.INTDELEGATE INTERFACE REFERENCE .....</b>	<b>300</b>
<b>Detailed Description.....</b>	<b>300</b>
<b>IMRCP.STORE.IMRCPOBSRESULTSET.INTDELEGATE INTERFACE REFERENCE .....</b>	<b>300</b>
<b>Detailed Description.....</b>	<b>300</b>
<b>IMRCP.SYSTEM.INTKEYVALUE&lt; T &gt; CLASS TEMPLATE REFERENCE .....</b>	<b>300</b>
<b>Public Member Functions .....</b>	<b>300</b>
<b>Detailed Description.....</b>	<b>301</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>301</b>
<b>Member Function Documentation .....</b>	<b>301</b>
<b>IMRCP.SYSTEM.INTROSORT CLASS REFERENCE .....</b>	<b>302</b>
<b>Static Public Member Functions .....</b>	<b>302</b>
<b>Detailed Description.....</b>	<b>302</b>
<b>Member Function Documentation .....</b>	<b>302</b>
<b>IMRCP.SYSTEM.IRUNTARGET&lt; E &gt; INTERFACE TEMPLATE REFERENCE .....</b>	<b>304</b>
<b>Public Member Functions .....</b>	<b>304</b>
<b>Detailed Description.....</b>	<b>304</b>
<b>Member Function Documentation .....</b>	<b>304</b>
<b>IMRCP.COLLECT.LAC2C CLASS REFERENCE.....</b>	<b>305</b>
<b>Public Member Functions .....</b>	<b>305</b>
<b>Additional Inherited Members .....</b>	<b>305</b>
<b>Detailed Description.....</b>	<b>305</b>
<b>Member Function Documentation .....</b>	<b>305</b>
<b>IMRCP.COMP.LAC2CDETECTORSCOMP CLASS REFERENCE .....</b>	<b>305</b>
<b>Classes .....</b>	<b>305</b>
<b>Public Member Functions .....</b>	<b>306</b>
<b>Protected Member Functions.....</b>	<b>306</b>
<b>Protected Attributes.....</b>	<b>306</b>
<b>Additional Inherited Members .....</b>	<b>306</b>
<b>Detailed Description.....</b>	<b>306</b>
<b>Member Function Documentation .....</b>	<b>306</b>
<b>Member Data Documentation .....</b>	<b>307</b>
<b>IMRCP.COMP.LAC2CEVENTSCOMP CLASS REFERENCE.....</b>	<b>307</b>
<b>Public Member Functions .....</b>	<b>307</b>
<b>Protected Member Functions.....</b>	<b>307</b>
<b>Additional Inherited Members .....</b>	<b>307</b>
<b>Detailed Description.....</b>	<b>307</b>
<b>Member Function Documentation .....</b>	<b>308</b>

<b>IMRCP.COLLECT.LADOTD511 CLASS REFERENCE .....</b>	<b>308</b>
Public Member Functions .....	308
Additional Inherited Members .....	308
Detailed Description.....	308
Member Function Documentation .....	308
<b>IMRCP.STORE.LADOTD511EVENT CLASS REFERENCE .....</b>	<b>309</b>
Public Member Functions .....	309
Additional Inherited Members .....	309
Detailed Description.....	309
Member Function Documentation .....	309
<b>IMRCP.COMP.LADOTD511LINE CLASS REFERENCE .....</b>	<b>310</b>
Public Member Functions .....	310
Protected Member Functions.....	310
Protected Attributes.....	310
Additional Inherited Members .....	310
Detailed Description.....	310
Member Function Documentation .....	310
Member Data Documentation .....	311
<b>IMRCP.COMP.LADOTD511POINT CLASS REFERENCE .....</b>	<b>311</b>
Public Member Functions .....	311
Protected Member Functions.....	311
Protected Attributes.....	311
Additional Inherited Members .....	311
Detailed Description.....	311
Member Function Documentation .....	312
Member Data Documentation .....	312
<b>IMRCP.STORE.LADOTDEVENT CLASS REFERENCE .....</b>	<b>313</b>
Public Member Functions .....	313
Public Attributes .....	313
Additional Inherited Members .....	313
Detailed Description.....	313
Constructor & Destructor Documentation.....	313
Member Function Documentation .....	313
Member Data Documentation .....	314
<b>IMRCP.STORE.GRIB.LAMBERTCONFORMALPROJ CLASS REFERENCE ....</b>	<b>315</b>
Public Member Functions .....	315
Public Attributes .....	315
Additional Inherited Members .....	315
Detailed Description.....	315
Constructor & Destructor Documentation.....	315
Member Data Documentation .....	316
<b>IMRCP.WEB.LANESERVLET CLASS REFERENCE .....</b>	<b>316</b>
Public Member Functions .....	316
Additional Inherited Members .....	316
Detailed Description.....	316
Member Function Documentation .....	316

<b>IMRCP.WEB.LATLNG CLASS REFERENCE .....</b>	<b>317</b>
Public Member Functions .....	317
Detailed Description.....	317
Constructor & Destructor Documentation.....	317
Member Function Documentation .....	318
<b>IMRCP.WEB.LATLNGBOUNDS CLASS REFERENCE .....</b>	<b>318</b>
Public Member Functions .....	318
Detailed Description.....	319
Constructor & Destructor Documentation.....	319
Member Function Documentation .....	319
<b>IMRCP.STORE.GRIB.LATLONPROJ CLASS REFERENCE .....</b>	<b>320</b>
Public Member Functions .....	320
Public Attributes .....	320
Additional Inherited Members .....	320
Detailed Description.....	321
Constructor & Destructor Documentation.....	321
Member Data Documentation .....	321
<b>IMRCP.WEB.LAYERS.LAYERSERVLET CLASS REFERENCE .....</b>	<b>321</b>
Public Member Functions .....	321
Protected Member Functions.....	322
Static Protected Attributes.....	322
Additional Inherited Members .....	322
Detailed Description.....	322
Member Function Documentation .....	322
Member Data Documentation .....	325
<b>IMRCP.SYSTEM.LOCKS CLASS REFERENCE .....</b>	<b>325</b>
Public Member Functions .....	325
Additional Inherited Members .....	326
Detailed Description.....	326
Member Function Documentation .....	326
<b>IMRCP.SYSTEM.MATHUTIL CLASS REFERENCE.....</b>	<b>326</b>
Static Public Member Functions .....	326
Detailed Description.....	327
Member Function Documentation .....	327
<b>IMRCP.GEOSRV.MERCATOR CLASS REFERENCE.....</b>	<b>327</b>
Public Member Functions .....	327
Static Public Member Functions .....	327
Static Public Attributes .....	328
Detailed Description.....	328
Constructor & Destructor Documentation.....	328
Member Function Documentation .....	328
Member Data Documentation .....	331
<b>IMRCP.FORECAST.MDSS.METRO CLASS REFERENCE .....</b>	<b>332</b>
Classes .....	332
Public Member Functions .....	332
Additional Inherited Members .....	332

<b>Detailed Description.....</b>	<b>332</b>
<b>Member Function Documentation .....</b>	<b>332</b>
<b>IMRCP.FORECAST.MDSS.METROFILESET CLASS REFERENCE .....</b>	<b>334</b>
<b>Public Member Functions .....</b>	<b>334</b>
<b>Static Public Member Functions .....</b>	<b>334</b>
<b>Public Attributes .....</b>	<b>334</b>
<b>Detailed Description.....</b>	<b>334</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>335</b>
<b>Member Function Documentation .....</b>	<b>335</b>
<b>Member Data Documentation .....</b>	<b>337</b>
<b>IMRCP.FORECAST.MDSS.METROPROCESS CLASS REFERENCE .....</b>	<b>338</b>
<b>Public Member Functions .....</b>	<b>338</b>
<b>Public Attributes .....</b>	<b>338</b>
<b>Detailed Description.....</b>	<b>338</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>338</b>
<b>Member Function Documentation .....</b>	<b>339</b>
<b>Member Data Documentation .....</b>	<b>339</b>
<b>IMRCP.STORE.METROSTORE CLASS REFERENCE .....</b>	<b>341</b>
<b>Public Member Functions .....</b>	<b>341</b>
<b>Additional Inherited Members .....</b>	<b>341</b>
<b>Detailed Description.....</b>	<b>341</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>341</b>
<b>Member Function Documentation .....</b>	<b>341</b>
<b>IMRCP.STORE.METROWRAPPER CLASS REFERENCE .....</b>	<b>342</b>
<b>Public Member Functions .....</b>	<b>342</b>
<b>Additional Inherited Members .....</b>	<b>342</b>
<b>Detailed Description.....</b>	<b>342</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>342</b>
<b>Member Function Documentation .....</b>	<b>342</b>
<b>IMRCP.FORECAST.MLP.MLPBLOCK CLASS REFERENCE .....</b>	<b>344</b>
<b>Classes .....</b>	<b>344</b>
<b>Public Member Functions .....</b>	<b>344</b>
<b>Static Public Member Functions .....</b>	<b>344</b>
<b>Static Public Attributes .....</b>	<b>344</b>
<b>Protected Member Functions.....</b>	<b>344</b>
<b>Protected Attributes.....</b>	<b>344</b>
<b>Static Protected Attributes.....</b>	<b>345</b>
<b>Additional Inherited Members .....</b>	<b>345</b>
<b>Detailed Description.....</b>	<b>345</b>
<b>Member Function Documentation .....</b>	<b>345</b>
<b>Member Data Documentation .....</b>	<b>349</b>
<b>IMRCP.STORE.MLPCSV CLASS REFERENCE .....</b>	<b>351</b>
<b>Public Member Functions .....</b>	<b>351</b>
<b>Additional Inherited Members .....</b>	<b>351</b>
<b>Detailed Description.....</b>	<b>351</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>352</b>

<b>Member Function Documentation .....</b>	<b>352</b>
<b>IMRCP.FORECAST.MLP.MLPHURRICANE CLASS REFERENCE .....</b>	<b>352</b>
<b>Classes .....</b>	<b>352</b>
<b>Public Member Functions .....</b>	<b>352</b>
<b>Additional Inherited Members .....</b>	<b>352</b>
<b>Detailed Description.....</b>	<b>352</b>
<b>Member Function Documentation .....</b>	<b>353</b>
<b>IMRCP.FORECAST.MLP.MLPMETADATA CLASS REFERENCE .....</b>	<b>354</b>
<b>Public Attributes .....</b>	<b>354</b>
<b>Detailed Description.....</b>	<b>354</b>
<b>Member Data Documentation .....</b>	<b>354</b>
<b>IMRCP.FORECAST.MLP.MLPPREDICT CLASS REFERENCE .....</b>	<b>355</b>
<b>Public Member Functions .....</b>	<b>355</b>
<b>Protected Member Functions.....</b>	<b>356</b>
<b>Protected Attributes.....</b>	<b>356</b>
<b>Additional Inherited Members .....</b>	<b>356</b>
<b>Detailed Description.....</b>	<b>356</b>
<b>Member Function Documentation .....</b>	<b>356</b>
<b>Member Data Documentation .....</b>	<b>358</b>
<b>IMRCP.FORECAST.MLP.MLPPROCESS CLASS REFERENCE .....</b>	<b>359</b>
<b>Public Member Functions .....</b>	<b>359</b>
<b>Public Attributes .....</b>	<b>359</b>
<b>Detailed Description.....</b>	<b>359</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>359</b>
<b>Member Function Documentation .....</b>	<b>359</b>
<b>Member Data Documentation .....</b>	<b>360</b>
<b>IMRCP.FORECAST.MLP.MLPRECORD CLASS REFERENCE.....</b>	<b>360</b>
<b>Public Member Functions .....</b>	<b>360</b>
<b>Public Attributes .....</b>	<b>360</b>
<b>Detailed Description.....</b>	<b>361</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>361</b>
<b>Member Function Documentation .....</b>	<b>361</b>
<b>Member Data Documentation .....</b>	<b>362</b>
<b>IMRCP.STORE.MLPSTORE CLASS REFERENCE .....</b>	<b>364</b>
<b>Protected Member Functions.....</b>	<b>364</b>
<b>Additional Inherited Members .....</b>	<b>364</b>
<b>Detailed Description.....</b>	<b>364</b>
<b>Member Function Documentation .....</b>	<b>364</b>
<b>IMRCP.FORECAST.MLP.MLPUPDATE CLASS REFERENCE .....</b>	<b>364</b>
<b>Public Member Functions .....</b>	<b>364</b>
<b>Static Public Member Functions .....</b>	<b>364</b>
<b>Public Attributes .....</b>	<b>364</b>
<b>Protected Member Functions.....</b>	<b>364</b>
<b>Additional Inherited Members .....</b>	<b>365</b>
<b>Detailed Description.....</b>	<b>365</b>
<b>Member Function Documentation .....</b>	<b>365</b>

<b>Member Data Documentation .....</b>	<b>367</b>
<b>IMRCP.WEB.MONITORSERVLET CLASS REFERENCE .....</b>	<b>367</b>
<b>Public Member Functions .....</b>	<b>367</b>
<b>Additional Inherited Members .....</b>	<b>367</b>
<b>Detailed Description.....</b>	<b>367</b>
<b>Member Function Documentation .....</b>	<b>368</b>
<b>IMRCP.STORE.NCFENTRYDATA CLASS REFERENCE .....</b>	<b>369</b>
<b>Public Member Functions .....</b>	<b>369</b>
<b>Public Attributes .....</b>	<b>369</b>
<b>Protected Attributes.....</b>	<b>369</b>
<b>Additional Inherited Members .....</b>	<b>369</b>
<b>Detailed Description.....</b>	<b>369</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>369</b>
<b>Member Function Documentation .....</b>	<b>370</b>
<b>Member Data Documentation .....</b>	<b>371</b>
<b>IMRCP.WEB.TILES.NCFTILECACHE CLASS REFERENCE .....</b>	<b>372</b>
<b>Public Member Functions .....</b>	<b>372</b>
<b>Protected Member Functions.....</b>	<b>372</b>
<b>Protected Attributes.....</b>	<b>372</b>
<b>Additional Inherited Members .....</b>	<b>372</b>
<b>Detailed Description.....</b>	<b>372</b>
<b>Member Function Documentation .....</b>	<b>372</b>
<b>Member Data Documentation .....</b>	<b>372</b>
<b>IMRCP.STORE.NCFWRAPPER CLASS REFERENCE .....</b>	<b>373</b>
<b>Public Member Functions .....</b>	<b>373</b>
<b>Public Attributes .....</b>	<b>373</b>
<b>Static Protected Member Functions.....</b>	<b>373</b>
<b>Protected Attributes.....</b>	<b>373</b>
<b>Additional Inherited Members .....</b>	<b>373</b>
<b>Detailed Description.....</b>	<b>374</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>374</b>
<b>Member Function Documentation .....</b>	<b>374</b>
<b>Member Data Documentation .....</b>	<b>376</b>
<b>IMRCP.STORE.NDFDQPFSTORE CLASS REFERENCE .....</b>	<b>377</b>
<b>Public Member Functions .....</b>	<b>377</b>
<b>Additional Inherited Members .....</b>	<b>377</b>
<b>Detailed Description.....</b>	<b>377</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>377</b>
<b>Member Function Documentation .....</b>	<b>377</b>
<b>IMRCP.STORE.NDFDQPFWRAPPER CLASS REFERENCE.....</b>	<b>378</b>
<b>Public Member Functions .....</b>	<b>378</b>
<b>Additional Inherited Members .....</b>	<b>378</b>
<b>Detailed Description.....</b>	<b>378</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>378</b>
<b>Member Function Documentation .....</b>	<b>378</b>
<b>IMRCP.GEOSRV.NETWORK CLASS REFERENCE .....</b>	<b>379</b>

<b>Public Member Functions .....</b>	<b>379</b>
<b>Public Attributes .....</b>	<b>379</b>
<b>Detailed Description.....</b>	<b>379</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>379</b>
<b>Member Function Documentation .....</b>	<b>379</b>
<b>Member Data Documentation .....</b>	<b>381</b>
<b>IMRCP.WEB.NETWORKGENERATION CLASS REFERENCE .....</b>	<b>382</b>
<b>Classes .....</b>	<b>382</b>
<b>Public Member Functions .....</b>	<b>382</b>
<b>Additional Inherited Members .....</b>	<b>382</b>
<b>Detailed Description.....</b>	<b>383</b>
<b>Member Function Documentation .....</b>	<b>383</b>
<b>IMRCP.COLLECT.NHCKMZ CLASS REFERENCE .....</b>	<b>390</b>
<b>Public Member Functions .....</b>	<b>390</b>
<b>Static Public Member Functions .....</b>	<b>390</b>
<b>Additional Inherited Members .....</b>	<b>390</b>
<b>Detailed Description.....</b>	<b>390</b>
<b>Member Function Documentation .....</b>	<b>390</b>
<b>IMRCP.STORE.NHCSTORE CLASS REFERENCE .....</b>	<b>391</b>
<b>Public Member Functions .....</b>	<b>391</b>
<b>Protected Member Functions.....</b>	<b>391</b>
<b>Additional Inherited Members .....</b>	<b>392</b>
<b>Detailed Description.....</b>	<b>392</b>
<b>Member Function Documentation .....</b>	<b>392</b>
<b>IMRCP.WEB.TILES.NHCTILES CLASS REFERENCE.....</b>	<b>393</b>
<b>Public Member Functions .....</b>	<b>393</b>
<b>Protected Member Functions.....</b>	<b>393</b>
<b>Additional Inherited Members .....</b>	<b>393</b>
<b>Detailed Description.....</b>	<b>393</b>
<b>Member Function Documentation .....</b>	<b>393</b>
<b>IMRCP.STORE.NHCWRAPPER CLASS REFERENCE .....</b>	<b>394</b>
<b>Classes .....</b>	<b>394</b>
<b>Public Member Functions .....</b>	<b>394</b>
<b>Static Public Member Functions .....</b>	<b>394</b>
<b>Public Attributes .....</b>	<b>394</b>
<b>Static Public Attributes .....</b>	<b>394</b>
<b>Additional Inherited Members .....</b>	<b>394</b>
<b>Detailed Description.....</b>	<b>394</b>
<b>Member Function Documentation .....</b>	<b>395</b>
<b>Member Data Documentation .....</b>	<b>396</b>
<b>IMRCP.COMP.NOTIFICATIONS CLASS REFERENCE.....</b>	<b>396</b>
<b>Public Member Functions .....</b>	<b>396</b>
<b>Additional Inherited Members .....</b>	<b>396</b>
<b>Detailed Description.....</b>	<b>397</b>
<b>Member Function Documentation .....</b>	<b>397</b>
<b>IMRCP.WEB.NOTIFICATIONSERVLET CLASS REFERENCE .....</b>	<b>397</b>

<b>Public Member Functions .....</b>	<b>397</b>
<b>Additional Inherited Members .....</b>	<b>397</b>
<b>Detailed Description.....</b>	<b>397</b>
<b>Member Function Documentation .....</b>	<b>398</b>
<b>IMRCP.COMP.NOTIFICATIONSET CLASS REFERENCE .....</b>	<b>398</b>
<b>Public Member Functions .....</b>	<b>398</b>
<b>Detailed Description.....</b>	<b>398</b>
<b>Member Function Documentation .....</b>	<b>399</b>
<b>IMRCP.STORE.OBS CLASS REFERENCE.....</b>	<b>399</b>
<b>Public Member Functions .....</b>	<b>399</b>
<b>Public Attributes .....</b>	<b>400</b>
<b>Static Public Attributes .....</b>	<b>400</b>
<b>Detailed Description.....</b>	<b>400</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>400</b>
<b>Member Function Documentation .....</b>	<b>403</b>
<b>Member Data Documentation .....</b>	<b>404</b>
<b>IMRCP.WEB.OBSCHARTREQUEST CLASS REFERENCE.....</b>	<b>407</b>
<b>Public Member Functions .....</b>	<b>407</b>
<b>Detailed Description.....</b>	<b>407</b>
<b>Member Function Documentation .....</b>	<b>407</b>
<b>IMRCP.WEB.OBSINFO CLASS REFERENCE .....</b>	<b>408</b>
<b>Public Member Functions .....</b>	<b>408</b>
<b>Public Attributes .....</b>	<b>408</b>
<b>Static Public Attributes .....</b>	<b>408</b>
<b>Detailed Description.....</b>	<b>408</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>408</b>
<b>Member Function Documentation .....</b>	<b>408</b>
<b>Member Data Documentation .....</b>	<b>408</b>
<b>IMRCP.WEB.OBSREQUEST CLASS REFERENCE .....</b>	<b>409</b>
<b>Public Member Functions .....</b>	<b>409</b>
<b>Detailed Description.....</b>	<b>409</b>
<b>Member Function Documentation .....</b>	<b>409</b>
<b>IMRCP.SYSTEM.OBSTYPE CLASS REFERENCE .....</b>	<b>411</b>
<b>Static Public Member Functions .....</b>	<b>411</b>
<b>Static Public Attributes .....</b>	<b>411</b>
<b>Detailed Description.....</b>	<b>412</b>
<b>Member Function Documentation .....</b>	<b>412</b>
<b>Member Data Documentation .....</b>	<b>415</b>
<b>IMRCP.STORE.OBSVIEW CLASS REFERENCE.....</b>	<b>420</b>
<b>Public Member Functions .....</b>	<b>420</b>
<b>Additional Inherited Members .....</b>	<b>420</b>
<b>Detailed Description.....</b>	<b>420</b>
<b>Member Function Documentation .....</b>	<b>420</b>
<b>IMRCP.COLLECT.OHGO CLASS REFERENCE .....</b>	<b>421</b>
<b>Public Member Functions .....</b>	<b>421</b>
<b>Additional Inherited Members .....</b>	<b>421</b>

Detailed Description.....	422
Member Function Documentation .....	422
<b>IMRCP.COMP.OHGOCONSTRUCTION CLASS REFERENCE .....</b>	<b>422</b>
Protected Member Functions.....	422
Additional Inherited Members .....	422
Detailed Description.....	422
Member Function Documentation .....	423
<b>IMRCP.COMP.OHGOINCIDENTS CLASS REFERENCE.....</b>	<b>423</b>
Protected Member Functions.....	423
Additional Inherited Members .....	423
Detailed Description.....	423
Member Function Documentation .....	423
<b>IMRCP.GEOSRV.OSM.OSMBINPARSER CLASS REFERENCE .....</b>	<b>424</b>
Public Member Functions .....	424
Detailed Description.....	424
Constructor & Destructor Documentation.....	424
Member Function Documentation .....	424
<b>IMRCP.GEOSRV.OSM.OSMBZ2TOBIN CLASS REFERENCE .....</b>	<b>426</b>
Public Member Functions .....	426
Static Public Member Functions .....	426
Detailed Description.....	426
Constructor & Destructor Documentation.....	427
Member Function Documentation .....	427
<b>IMRCP.GEOSRV.OSM.OSMNODE CLASS REFERENCE.....</b>	<b>428</b>
Public Member Functions .....	428
Public Attributes .....	428
Static Public Attributes .....	428
Detailed Description.....	428
Constructor & Destructor Documentation.....	428
Member Function Documentation .....	429
Member Data Documentation .....	430
<b>IMRCP.GEOSRV.OSM.OSMOBJECT CLASS REFERENCE .....</b>	<b>431</b>
Public Member Functions .....	431
Public Attributes .....	431
Static Public Attributes .....	431
Detailed Description.....	431
Constructor & Destructor Documentation.....	431
Member Data Documentation .....	432
<b>IMRCP.GEOSRV.OSM.OSMUTIL CLASS REFERENCE .....</b>	<b>432</b>
Static Public Member Functions .....	432
Static Public Attributes .....	433
Detailed Description.....	433
Member Function Documentation .....	433
Member Data Documentation .....	439
<b>IMRCP.GEOSRV.OSM.OSMWAY CLASS REFERENCE .....</b>	<b>439</b>
Public Member Functions .....	439

<b>Public Attributes .....</b>	<b>439</b>
<b>Static Public Attributes .....</b>	<b>440</b>
<b>Detailed Description.....</b>	<b>440</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>440</b>
<b>Member Function Documentation .....</b>	<b>441</b>
<b>Member Data Documentation .....</b>	<b>443</b>
<b>IMRCP.WEB.OUTPUTCSV CLASS REFERENCE .....</b>	<b>444</b>
<b>Protected Attributes.....</b>	<b>444</b>
<b>Static Protected Attributes.....</b>	<b>444</b>
<b>Additional Inherited Members .....</b>	<b>445</b>
<b>Detailed Description.....</b>	<b>445</b>
<b>Member Data Documentation .....</b>	<b>445</b>
<b>IMRCP.WEB.OUTPUTFORMAT CLASS REFERENCE.....</b>	<b>445</b>
<b>Public Member Functions .....</b>	<b>445</b>
<b>Protected Member Functions.....</b>	<b>445</b>
<b>Protected Attributes.....</b>	<b>445</b>
<b>Detailed Description.....</b>	<b>445</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>446</b>
<b>Member Function Documentation .....</b>	<b>446</b>
<b>Member Data Documentation .....</b>	<b>446</b>
<b>IMRCP.COMP.LAC2CDETECTORSCOMP.PARSER CLASS REFERENCE .....</b>	<b>446</b>
<b>Public Member Functions .....</b>	<b>446</b>
<b>Protected Member Functions.....</b>	<b>446</b>
<b>Protected Attributes.....</b>	<b>446</b>
<b>Detailed Description.....</b>	<b>446</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>446</b>
<b>Member Function Documentation .....</b>	<b>447</b>
<b>Member Data Documentation .....</b>	<b>447</b>
<b>IMRCP.COMP.PCCAT CLASS REFERENCE .....</b>	<b>447</b>
<b>Public Member Functions .....</b>	<b>447</b>
<b>Protected Attributes.....</b>	<b>447</b>
<b>Additional Inherited Members .....</b>	<b>448</b>
<b>Detailed Description.....</b>	<b>448</b>
<b>Member Function Documentation .....</b>	<b>448</b>
<b>Member Data Documentation .....</b>	<b>449</b>
<b>IMRCP.STORE.GRIB.PNGINSTREAM CLASS REFERENCE.....</b>	<b>450</b>
<b>Public Member Functions .....</b>	<b>450</b>
<b>Static Public Attributes .....</b>	<b>450</b>
<b>Detailed Description.....</b>	<b>450</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>450</b>
<b>Member Function Documentation .....</b>	<b>451</b>
<b>Member Data Documentation .....</b>	<b>451</b>
<b>IMRCP.SYSTEM.SHP.POINT CLASS REFERENCE .....</b>	<b>452</b>
<b>Public Member Functions .....</b>	<b>452</b>
<b>Public Attributes .....</b>	<b>452</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>452</b>

<b>IMRCP.WEB.LAYERS.POINTSLAYERSERVLET CLASS REFERENCE .....</b>	<b>452</b>
Protected Member Functions.....	452
Additional Inherited Members .....	453
Detailed Description.....	453
Member Function Documentation .....	453
<b>IMRCP.SYSTEM.SHP.POLYLINE CLASS REFERENCE .....</b>	<b>453</b>
Public Member Functions .....	453
Protected Member Functions.....	453
Additional Inherited Members .....	453
Detailed Description.....	453
Constructor & Destructor Documentation.....	454
Member Function Documentation .....	454
<b>IMRCP.SYSTEM.SHP.POLYSHAPE CLASS REFERENCE.....</b>	<b>455</b>
Public Member Functions .....	455
Static Public Member Functions .....	455
Public Attributes .....	455
Protected Member Functions.....	455
Static Protected Member Functions .....	455
Protected Attributes.....	455
Detailed Description.....	455
Constructor & Destructor Documentation.....	455
Member Function Documentation .....	456
Member Data Documentation .....	458
<b>IMRCP.SYSTEM.SHP.POLYSHAPEITERATOR CLASS REFERENCE .....</b>	<b>458</b>
Public Member Functions .....	458
Detailed Description.....	458
Member Function Documentation .....	459
<b>IMRCP.FORECAST.MLP.PREDICTION CLASS REFERENCE .....</b>	<b>459</b>
Public Member Functions .....	459
Detailed Description.....	459
Constructor & Destructor Documentation.....	460
Member Function Documentation .....	460
<b>IMRCP.STORE.GRIB.PROJECTION CLASS REFERENCE .....</b>	<b>460</b>
Static Public Member Functions .....	460
Public Attributes .....	460
Detailed Description.....	460
Member Function Documentation .....	461
Member Data Documentation .....	461
<b>IMRCP.STORE.PROJPROFILE CLASS REFERENCE.....</b>	<b>462</b>
Public Member Functions .....	462
Public Attributes .....	462
Static Public Attributes .....	462
Detailed Description.....	462
Constructor & Destructor Documentation.....	462
Member Function Documentation .....	463
Member Data Documentation .....	464

<b>IMRCP.STORE.PROJPROFILES CLASS REFERENCE .....</b>	<b>465</b>
Public Member Functions .....	465
Static Public Member Functions .....	465
Detailed Description.....	465
Member Function Documentation .....	465
<b>IMRCP.GEOSRV.RANGERULES CLASS REFERENCE.....</b>	<b>466</b>
Public Member Functions .....	466
Public Attributes .....	467
Detailed Description.....	467
Constructor & Destructor Documentation.....	467
Member Function Documentation .....	467
Member Data Documentation .....	468
<b>IMRCP.STORE.RAPNCFWRAPPER CLASS REFERENCE .....</b>	<b>468</b>
Public Member Functions .....	468
Additional Inherited Members .....	468
Detailed Description.....	468
Constructor & Destructor Documentation.....	469
Member Function Documentation .....	469
<b>IMRCP.STORE.RAPSTORE CLASS REFERENCE .....</b>	<b>470</b>
Public Member Functions .....	470
Additional Inherited Members .....	470
Detailed Description.....	470
Constructor & Destructor Documentation.....	470
Member Function Documentation .....	470
<b>IMRCP.SYSTEM.DIRECTORY.REGISTEREDBLOCK CLASS REFERENCE ....</b>	<b>471</b>
Public Member Functions .....	471
Detailed Description.....	471
Member Function Documentation .....	471
<b>IMRCP.COLLECT.REMOTEGRID CLASS REFERENCE.....</b>	<b>471</b>
Public Member Functions .....	471
Protected Member Functions.....	471
Protected Attributes.....	471
Additional Inherited Members .....	472
Detailed Description.....	472
Constructor & Destructor Documentation.....	472
Member Function Documentation .....	472
Member Data Documentation .....	473
<b>IMRCP.WEB.REPORTSUBSCRIPTION CLASS REFERENCE .....</b>	<b>474</b>
Classes .....	474
Public Member Functions .....	474
Public Attributes .....	474
Static Public Attributes .....	475
Detailed Description.....	475
Constructor & Destructor Documentation.....	475
Member Function Documentation .....	476
Member Data Documentation .....	478

<b>IMRCP.FORECAST.MDSS.ROADCASTDATA CLASS REFERENCE.....</b>	<b>479</b>
Public Member Functions .....	479
Public Attributes .....	479
Detailed Description.....	479
Member Function Documentation .....	480
Member Data Documentation .....	480
<b>IMRCP.WEB.LAYERS.ROADLAYERSERVLET CLASS REFERENCE .....</b>	<b>481</b>
Public Member Functions .....	481
Protected Member Functions.....	481
Additional Inherited Members .....	481
Detailed Description.....	481
Member Function Documentation .....	481
<b>IMRCP.WEB.SCENARIO CLASS REFERENCE .....</b>	<b>482</b>
Public Member Functions .....	482
Public Attributes .....	482
Detailed Description.....	482
Constructor & Destructor Documentation.....	482
Member Function Documentation .....	483
Member Data Documentation .....	483
<b>IMRCP.WEB.SCENARIOS CLASS REFERENCE.....</b>	<b>484</b>
Public Member Functions .....	484
Additional Inherited Members .....	484
Detailed Description.....	484
Member Function Documentation .....	485
<b>IMRCP.SYSTEM.SCHEDULING CLASS REFERENCE .....</b>	<b>488</b>
Public Member Functions .....	488
Static Public Member Functions .....	488
Detailed Description.....	488
Member Function Documentation .....	488
<b>IMRCP.WEB.SECUREBASEBLOCK CLASS REFERENCE .....</b>	<b>490</b>
Public Member Functions .....	490
Protected Member Functions.....	490
Additional Inherited Members .....	490
Detailed Description.....	490
Constructor & Destructor Documentation.....	491
Member Function Documentation .....	491
<b>IMRCP.GEOSRV.SEGITERATOR CLASS REFERENCE .....</b>	<b>492</b>
Public Member Functions .....	492
Detailed Description.....	492
Member Function Documentation .....	492
<b>IMRCP.WEB.SEGMENTGROUP CLASS REFERENCE.....</b>	<b>493</b>
Public Member Functions .....	493
Public Attributes .....	493
Detailed Description.....	493
Constructor & Destructor Documentation.....	493
Member Data Documentation .....	493

<b>IMRCP.WEB.SESSION CLASS REFERENCE .....</b>	<b>494</b>
Public Member Functions .....	494
Public Attributes .....	494
Protected Attributes.....	494
Detailed Description.....	494
Member Function Documentation .....	494
Member Data Documentation .....	494
<b>IMRCP.WEB.SESSMGR CLASS REFERENCE .....</b>	<b>495</b>
Public Member Functions .....	495
Static Public Member Functions .....	495
Static Protected Attributes.....	495
Detailed Description.....	495
Constructor & Destructor Documentation.....	496
Member Function Documentation .....	496
Member Data Documentation .....	496
<b>IMRCP.STORE.SOURCEUNIT CLASS REFERENCE .....</b>	<b>497</b>
Public Member Functions .....	497
Public Attributes .....	497
Detailed Description.....	497
Constructor & Destructor Documentation.....	497
Member Data Documentation .....	497
<b>IMRCP.STORE.SPATIALFILECACHE CLASS REFERENCE.....</b>	<b>498</b>
Classes .....	498
Public Member Functions .....	498
Public Attributes .....	498
Protected Member Functions.....	498
Protected Attributes.....	498
Additional Inherited Members .....	498
Detailed Description.....	498
Constructor & Destructor Documentation.....	498
Member Function Documentation .....	498
Member Data Documentation .....	500
<b>IMRCP.STORE.SPATIALFILEWRAPPER CLASS REFERENCE .....</b>	<b>501</b>
Public Member Functions .....	501
Public Attributes .....	501
Additional Inherited Members .....	501
Detailed Description.....	501
Member Function Documentation .....	501
Member Data Documentation .....	502
<b>IMRCP.SYSTEM.STRINGPOOL CLASS REFERENCE .....</b>	<b>502</b>
Classes .....	502
Public Member Functions .....	502
Protected Attributes.....	502
Detailed Description.....	503
Constructor & Destructor Documentation.....	503
Member Function Documentation .....	503

<b>Member Data Documentation .....</b>	<b>503</b>
<b>IMRCP.WEB.SUBSCRIPTIONS CLASS REFERENCE .....</b>	<b>504</b>
<b>Public Member Functions .....</b>	<b>504</b>
<b>Protected Member Functions.....</b>	<b>504</b>
<b>Protected Attributes.....</b>	<b>504</b>
<b>Additional Inherited Members .....</b>	<b>504</b>
<b>Detailed Description.....</b>	<b>504</b>
<b>Member Function Documentation .....</b>	<b>504</b>
<b>Member Data Documentation .....</b>	<b>509</b>
<b>IMRCP.FORECAST.MLP.SVMRECORD CLASS REFERENCE.....</b>	<b>509</b>
<b>Public Member Functions .....</b>	<b>509</b>
<b>Public Attributes .....</b>	<b>509</b>
<b>Static Public Attributes .....</b>	<b>510</b>
<b>Detailed Description.....</b>	<b>510</b>
<b>Member Function Documentation .....</b>	<b>510</b>
<b>Member Data Documentation .....</b>	<b>510</b>
<b>IMRCP.SYSTEM.TEXT CLASS REFERENCE .....</b>	<b>512</b>
<b>Static Public Member Functions .....</b>	<b>512</b>
<b>Detailed Description.....</b>	<b>512</b>
<b>Member Function Documentation .....</b>	<b>512</b>
<b>IMRCP.WEB.TILES.TILE CLASS REFERENCE.....</b>	<b>517</b>
<b>Public Member Functions .....</b>	<b>517</b>
<b>Public Attributes .....</b>	<b>517</b>
<b>Protected Member Functions.....</b>	<b>517</b>
<b>Detailed Description.....</b>	<b>517</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>517</b>
<b>Member Function Documentation .....</b>	<b>518</b>
<b>Member Data Documentation .....</b>	<b>518</b>
<b>IMRCP.WEB.TILES.TILEAREA CLASS REFERENCE .....</b>	<b>518</b>
<b>Public Member Functions .....</b>	<b>518</b>
<b>Public Attributes .....</b>	<b>518</b>
<b>Detailed Description.....</b>	<b>519</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>519</b>
<b>Member Function Documentation .....</b>	<b>519</b>
<b>Member Data Documentation .....</b>	<b>519</b>
<b>IMRCP.GEOSRV.OSM.TILEBUCKET CLASS REFERENCE.....</b>	<b>519</b>
<b>Public Member Functions .....</b>	<b>519</b>
<b>Public Attributes .....</b>	<b>519</b>
<b>Detailed Description.....</b>	<b>520</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>520</b>
<b>Member Function Documentation .....</b>	<b>520</b>
<b>Member Data Documentation .....</b>	<b>520</b>
<b>IMRCP.WEB.TILES.TILECACHE CLASS REFERENCE .....</b>	<b>520</b>
<b>Public Member Functions .....</b>	<b>520</b>
<b>Protected Member Functions.....</b>	<b>521</b>
<b>Protected Attributes.....</b>	<b>521</b>

Additional Inherited Members .....	521
Detailed Description.....	521
Member Function Documentation .....	521
Member Data Documentation .....	522
<b>IMRCP.WEB.TILES.TILESERVLET CLASS REFERENCE .....</b>	<b>523</b>
Public Member Functions .....	523
Protected Member Functions.....	523
Protected Attributes.....	523
Static Protected Attributes.....	523
Additional Inherited Members .....	523
Detailed Description.....	523
Member Function Documentation .....	524
Member Data Documentation .....	525
<b>IMRCP.WEB.TILES.TILEUTIL CLASS REFERENCE .....</b>	<b>526</b>
Static Public Member Functions .....	526
Static Public Attributes .....	527
Detailed Description.....	527
Member Function Documentation .....	527
Member Data Documentation .....	531
<b>IMRCP.WEB.TILES.TILEWRAPPER CLASS REFERENCE .....</b>	<b>531</b>
Public Member Functions .....	531
Public Attributes .....	531
Protected Attributes.....	531
Additional Inherited Members .....	532
Detailed Description.....	532
Member Function Documentation .....	532
Member Data Documentation .....	533
<b>IMRCP.STORE.SPATIALFILECACHE.TILEXCACHE CLASS REFERENCE....</b>	<b>533</b>
Public Member Functions .....	533
Protected Member Functions.....	533
Protected Attributes.....	533
Detailed Description.....	533
Member Function Documentation .....	534
Member Data Documentation .....	534
<b>IMRCP.STORE.SPATIALFILECACHE.TILEYCACHE CLASS REFERENCE....</b>	<b>534</b>
Public Member Functions .....	534
Protected Member Functions.....	534
Protected Attributes.....	534
Detailed Description.....	534
Member Function Documentation .....	535
Member Data Documentation .....	535
<b>IMRCP.STORE.NHCWRAPPER.TRACKPARSER CLASS REFERENCE .....</b>	<b>535</b>
Protected Member Functions.....	535
Detailed Description.....	536
Member Function Documentation .....	536
<b>IMRCP.STORE.TRAFFICEVENTCSV CLASS REFERENCE .....</b>	<b>536</b>

<b>Public Member Functions .....</b>	<b>536</b>
<b>Additional Inherited Members .....</b>	<b>536</b>
<b>Detailed Description.....</b>	<b>536</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>536</b>
<b>Member Function Documentation .....</b>	<b>537</b>
<b>IMRCP.STORE.TRAFFICEVENTSTORE CLASS REFERENCE.....</b>	<b>537</b>
<b>Public Member Functions .....</b>	<b>537</b>
<b>Protected Member Functions.....</b>	<b>537</b>
<b>Additional Inherited Members .....</b>	<b>537</b>
<b>Detailed Description.....</b>	<b>537</b>
<b>Member Function Documentation .....</b>	<b>538</b>
<b>IMRCP.STORE.TRAFFICSPEEDSTORE CLASS REFERENCE.....</b>	<b>538</b>
<b>Public Member Functions .....</b>	<b>538</b>
<b>Protected Member Functions.....</b>	<b>538</b>
<b>Additional Inherited Members .....</b>	<b>538</b>
<b>Detailed Description.....</b>	<b>538</b>
<b>Member Function Documentation .....</b>	<b>539</b>
<b>IMRCP.STORE.TRAFFICSPEEDSTOREWRAPPER CLASS REFERENCE .....</b>	<b>539</b>
<b>Public Member Functions .....</b>	<b>539</b>
<b>Additional Inherited Members .....</b>	<b>539</b>
<b>Detailed Description.....</b>	<b>539</b>
<b>Member Function Documentation .....</b>	<b>539</b>
<b>IMRCP.SYSTEM.UNITS.UNITCONV CLASS REFERENCE .....</b>	<b>540</b>
<b>Public Member Functions .....</b>	<b>540</b>
<b>Protected Attributes.....</b>	<b>541</b>
<b>Detailed Description.....</b>	<b>541</b>
<b>Member Function Documentation .....</b>	<b>541</b>
<b>Member Data Documentation .....</b>	<b>541</b>
<b>IMRCP.SYSTEM.UNITS CLASS REFERENCE.....</b>	<b>542</b>
<b>Classes .....</b>	<b>542</b>
<b>Public Member Functions .....</b>	<b>542</b>
<b>Static Public Member Functions .....</b>	<b>542</b>
<b>Protected Attributes.....</b>	<b>542</b>
<b>Detailed Description.....</b>	<b>542</b>
<b>Member Function Documentation .....</b>	<b>543</b>
<b>IMRCP.WEB.USCENPLACELOOKUP CLASS REFERENCE.....</b>	<b>544</b>
<b>Public Member Functions .....</b>	<b>544</b>
<b>Public Attributes .....</b>	<b>544</b>
<b>Static Public Attributes .....</b>	<b>544</b>
<b>Additional Inherited Members .....</b>	<b>544</b>
<b>Detailed Description.....</b>	<b>544</b>
<b>Member Function Documentation .....</b>	<b>544</b>
<b>Member Data Documentation .....</b>	<b>545</b>
<b>IMRCP.WEB.USERPROFILE CLASS REFERENCE.....</b>	<b>546</b>
<b>Public Member Functions .....</b>	<b>546</b>
<b>Public Attributes .....</b>	<b>546</b>

<b>Detailed Description.....</b>	<b>546</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>546</b>
<b>Member Data Documentation .....</b>	<b>547</b>
<b>IMRCP.WEB.USERSETTINGSERVLET CLASS REFERENCE.....</b>	<b>547</b>
<b>Public Member Functions .....</b>	<b>547</b>
<b>Additional Inherited Members .....</b>	<b>547</b>
<b>Detailed Description.....</b>	<b>547</b>
<b>Member Function Documentation .....</b>	<b>547</b>
<b>IMRCP.SYSTEMUTIL CLASS REFERENCE .....</b>	<b>548</b>
<b>Static Public Member Functions .....</b>	<b>548</b>
<b>Detailed Description.....</b>	<b>548</b>
<b>Member Function Documentation .....</b>	<b>549</b>
<b>IMRCP.SYSTEMUTILITY CLASS REFERENCE .....</b>	<b>550</b>
<b>Static Public Member Functions .....</b>	<b>550</b>
<b>Detailed Description.....</b>	<b>551</b>
<b>Member Function Documentation .....</b>	<b>551</b>
<b>IMRCP.GEOSRV.OSM.WAYITERATOR CLASS REFERENCE .....</b>	<b>553</b>
<b>Public Member Functions .....</b>	<b>553</b>
<b>Detailed Description.....</b>	<b>554</b>
<b>Member Function Documentation .....</b>	<b>554</b>
<b>IMRCP.GEOSRV.WAYNETWORKS.WAYMETADATA CLASS REFERENCE ..</b>	<b>554</b>
<b>Public Member Functions .....</b>	<b>554</b>
<b>Public Attributes .....</b>	<b>554</b>
<b>Detailed Description.....</b>	<b>554</b>
<b>Member Function Documentation .....</b>	<b>554</b>
<b>Member Data Documentation .....</b>	<b>555</b>
<b>IMRCP.GEOSRV.WAYNETWORKS CLASS REFERENCE .....</b>	<b>555</b>
<b>Classes .....</b>	<b>555</b>
<b>Public Member Functions .....</b>	<b>555</b>
<b>Additional Inherited Members .....</b>	<b>556</b>
<b>Detailed Description.....</b>	<b>556</b>
<b>Member Function Documentation .....</b>	<b>556</b>
<b>IMRCP.GEOSRV.WAYSNAPINFO CLASS REFERENCE .....</b>	<b>561</b>
<b>Public Member Functions .....</b>	<b>561</b>
<b>Public Attributes .....</b>	<b>562</b>
<b>Detailed Description.....</b>	<b>562</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>562</b>
<b>Member Function Documentation .....</b>	<b>562</b>
<b>Member Data Documentation .....</b>	<b>563</b>
<b>IMRCP.STORE.WEATHERSTORE CLASS REFERENCE .....</b>	<b>563</b>
<b>Public Member Functions .....</b>	<b>563</b>
<b>Protected Member Functions.....</b>	<b>563</b>
<b>Protected Attributes.....</b>	<b>563</b>
<b>Additional Inherited Members .....</b>	<b>563</b>
<b>Detailed Description.....</b>	<b>564</b>
<b>Member Function Documentation .....</b>	<b>564</b>

<b>Member Data Documentation .....</b>	<b>564</b>
<b>IMRCP.FORECAST.MLP.WORK CLASS REFERENCE.....</b>	<b>565</b>
<b>Public Member Functions .....</b>	<b>565</b>
<b>Public Attributes .....</b>	<b>565</b>
<b>Detailed Description.....</b>	<b>565</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>565</b>
<b>Member Data Documentation .....</b>	<b>565</b>
<b>IMRCP.FORECAST.MLP.MLPBLOCK.WORKDELEGATE CLASS REFERENCE</b>	<b>566</b>
<b>Public Member Functions .....</b>	<b>566</b>
<b>Detailed Description.....</b>	<b>566</b>
<b>Member Function Documentation .....</b>	<b>566</b>
<b>IMRCP.FORECAST.MLP.WORKOBJECT CLASS REFERENCE.....</b>	<b>566</b>
<b>Public Member Functions .....</b>	<b>566</b>
<b>Public Attributes .....</b>	<b>566</b>
<b>Detailed Description.....</b>	<b>566</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>566</b>
<b>Member Function Documentation .....</b>	<b>567</b>
<b>Member Data Documentation .....</b>	<b>567</b>
<b>IMRCP.STORE.WXDECSV CLASS REFERENCE .....</b>	<b>567</b>
<b>Public Member Functions .....</b>	<b>567</b>
<b>Additional Inherited Members .....</b>	<b>567</b>
<b>Detailed Description.....</b>	<b>567</b>
<b>Constructor &amp; Destructor Documentation.....</b>	<b>568</b>
<b>Member Function Documentation .....</b>	<b>568</b>
<b>IMRCP.COLLECT.WXDESUB CLASS REFERENCE.....</b>	<b>568</b>
<b>Public Member Functions .....</b>	<b>568</b>
<b>Additional Inherited Members .....</b>	<b>568</b>
<b>Detailed Description.....</b>	<b>568</b>
<b>Member Function Documentation .....</b>	<b>569</b>
<b>IMRCP.STORE.WXDESUBSTORE CLASS REFERENCE .....</b>	<b>570</b>
<b>Public Member Functions .....</b>	<b>570</b>
<b>Protected Member Functions.....</b>	<b>570</b>
<b>Additional Inherited Members .....</b>	<b>570</b>
<b>Detailed Description.....</b>	<b>570</b>
<b>Member Function Documentation .....</b>	<b>570</b>

## LIST OF FIGURES

Figure 1. Diagram. Integrated Modeling for Road Condition Prediction Composition .....	8
Figure 2. Diagram. Integrated Modeling for Road Condition Prediction Process.....	9
Figure 3. Proposed Machine Learning-based Prediction Network Model Algorithm.....	27
Figure 4. Machine Learning-based Prediction Estimator Output Example.....	29
Figure 5. Graphical Representation of the Integrated Model for Road Condition Prediction Machine Learning-based Prediction Process for Hurricane Traffic. ....	32
Figure 6. Integrated Model for Road Condition Prediction Deployment. ....	48



## LIST OF TABLES

Table 1. Variables and State Definitions for Machine Learning-based Prediction .....	27
Table 2. Machine Learning-based Prediction Scenario Identifier Input Categories.....	29
Table 3. Variables and State Definitions for Machine Learning-based Prediction for Hurricane Traffic. ....	32
Table 4. Input Data Available Frequency.....	51
Table 5. Layer Definitions. ....	53
Table 6. Observation type descriptions.....	61
Table 7. Observation types enumeration. ....	62
Table 8. Observation type source descriptions.....	70
Table 9. Observation type synthesis algorithms. ....	72
Table 10.Traffic Alert Definitions .....	76
Table 11. Weather Alert Definitions.....	76
Table 12. Road Condition Alert Definitions.....	77
Table 13. Tropical Storm Categories. ....	77



## LIST OF ABBREVIATIONS

ADCIRC	Advanced Circulation model for oceanic, coastal, and estuarine waters
AHPS	Advanced Hydrologic Prediction Service
ATMS	Advanced Transportation Management System
C2C	Center to Center
CAP	Common Alerting Protocol
ConOps	Concept of Operations
CSV	Comma Separated Value
CV	Connected Vehicle
DEM	Digital Elevation Map
DMS	Dynamic Message Signs
ESS	Environmental Sensor Station
FHWA	Federal Highway Administration
GFS	Global Forecast System
GRIB2	NCEP and WMO's Gridded Binary File Format Version 2
IMRCP	Integrated Model for Road Condition Prediction
LADOTD	Louisiana Department of Transportation and Development
MDSS	Maintenance Decision Support System
METRo	Model of the Environment and Temperature of Roads
MLP	Machine Learning-based Prediction
MRMS	Multi Radar Multi Sensor
NCEP	National Centers for Environmental Predictions
NDFD	National Digital Forecast Database
NHC	National Hurricane Center
NOAA	National Oceanic and Atmospheric Administration
NWS	National Weather Service
RAP	Rapid Refresh weather model
RTMA	Real-Time Mesoscale Analysis
SAD	System Architecture Description
TMC	Transportation Management Center
URL	Uniform Resource Locator
USDOT	United States Department of Transportation
WMO	World Meteorological Organization
WxDE	Weather Data Environment



## **EXECUTIVE SUMMARY**

Transportation system management and operations (TSMO) is on the cusp of dramatic changes due to the increased availability of data and the increasing sophistication of models and systems supporting those operations. Intelligent transportation systems (ITS) are widely deployed and gather data about weather and traffic conditions across road networks. The imminent deployment of connected vehicles (CV) will bring an orders-of-magnitude increase in data availability. This array of data powers traffic and road condition predictions, and as data availability increases, the accuracy and reliability of the models improve. This convergence of opportunities presents enough potential for operational improvements in safety and mobility that the Federal Highway Administration's (FHWA) Road Weather Management Program (RWMP) is initiating research into an integrated model for road condition prediction (IMRCP) to investigate and capture that potential. This system design description (SDD) documents a common understanding among stakeholder groups of system features and components, and serves as a basis for system development activities. Follow-on efforts will implement the design and deploy the system with an operating transportation agency to evaluate its effectiveness.

The SDD consists of an introduction describing the objectives of both the IMRCP system and the SDD itself, a general description of the IMRCP, and a design description of the system components.



## **CHAPTER 1. INTRODUCTION**

### **BACKGROUND**

FHWA has embarked on efforts to describe and create a tool that results from an ensemble of forecast and probabilistic models and incorporates real-time and/or archived data, fusing them to predict current and future overall road/travel conditions for travelers, transportation operators, and maintenance providers.

The foundational elements needed to characterize Integrated Modeling for Road Condition Prediction (IMRCP) were developed in the Phase 1 development of a Concept of Operations and Requirements. The IMRCP Phase 2 work specified, implemented, tested, and evaluated the IMRCP concept in a demonstration deployment with local agencies in the Kansas City metropolitan area. Phase 3 expanded the IMRCP deployment across the entire Kansas City metro area, implemented a machine-learning based traffic model, operated the system for two winter seasons, evaluated the operational results, and updated the system documentation.

### **PURPOSE**

The objectives of IMRCP Phase 4 are to improve and deploy the system in two new locations (Ohio and Louisiana), expanding system capabilities and applicability to extreme events. This will address some gaps and recommendations from Phase 3; deploy to a metropolitan area for planning, monitoring, and post-event assessment of traffic management during adverse weather conditions; deploy to a metro/region/state transportation management center to assess improvements to public safety, evacuation and emergency response to severe/extreme weather conditions; and update the system documentation to support future deployments. This System Design Description updates the Phase 3 design to reflect system changes needed to support the Phase 4 scope and objectives.

### **SCOPE**

IMRCP provides a framework for the integration of road condition monitoring and forecast data to support tactical and strategic decisions by travelers, transportation operators, and maintenance providers. The system will collect and integrate environmental observations and transportation operations data; collect forecast environmental data and operations data when available; initiate road weather and traffic forecasts based on the collected data; generate travel and operational advisories and warnings from the collected real-time and forecast data; and provide the road condition data, forecasts, advisories and warnings to other applications and systems. Road condition and operations data and forecasts to be integrated into the predictions may include atmospheric weather; road (surface) weather; small stream, river, and coastal water levels; road network capacity; road network demand; traffic conditions and forecasts; traffic control states; work zones; maintenance activities and plans; and data related to emergency preparedness and emergency operations.

## **DOCUMENT OVERVIEW**

The structure of this document is generally consistent with the outline of a System or Software Design Description defined in ISO/IEC/IEEE Standard 42010-2011. Some sections herein have been enhanced to accommodate more detailed content than described in the standard. Titles of some sections have been edited to specifically capture that enhancement.

Section 2 provides a general description of the system perspective and stakeholder concerns. It is largely a summary of material described in more detail in the ConOps.

Section 3 documents the system design. The relevant architectural viewpoints are identified, and views and models are described for each viewpoint. Rationales for and correspondence among elements of the views are included in the view and model descriptions. Four viewpoints are described: composition, process, deployment, and related designs.

## CHAPTER 2. GENERAL DESCRIPTION

### SYSTEM PERSPECTIVE

Describing and predicting roadway conditions and events that may impact travel across road networks requires an understanding of and tools for interacting with the system and its operations across all of the road network's stakeholder groups. For example, travelers have an immediate need for information about conditions along their planned route, and they contribute to the aggregate travel conditions along their route by their choices and behaviors. Winter maintenance crews plan ahead for reducing the impact of storms on roadway conditions based on weather forecasts and perhaps on a sophisticated maintenance decision support system (MDSS), but also adapt to conditions on the roadway as they execute those plans. Operators in a transportation management center (TMC) monitor roadway conditions across a network with cameras and sensors accessed through an Advanced Transportation Management System (ATMS), and respond to conditions and events by generating alerts to be published on dynamic message signs (DMS) on the roadside and pushed out to web pages and mobile apps through traveler information systems. In all these examples, stakeholders are making and executing plans, monitoring and adjusting to current conditions, and potentially changing their plans based on their analyses of potential future conditions.

A complete context for prediction of road conditions would have to consider a broad range of stakeholders, their activities, and interactions with the roadway, their decision processes, and the underlying models of the roadway and environmental conditions. Descriptions of the current state of stakeholders and their activities in the IMRCP ConOps therefore focused on identifying the processes and decisions that are affected by currently available roadway condition information and predictions. An analysis of current and imminent road and weather condition models was performed in a previous task and documented in the Integrated Modeling for Road Condition Prediction Model Analysis. The aggregate of these analyses of modeling capabilities and stakeholder interests formed the basis for the architectural views of the functional and system packages for a potential IMRCP system in the ConOps.

Architectural views of a system provide sketches of what an implemented system might look like from various conceptual frameworks. For the IMRCP, the functional view of the system describes the system's purpose: to provide integrated predictions of road weather and traffic conditions. To fulfill this purpose, the system will have to have models for the roadways, weather, traffic, and hydrology; , prediction capabilities and forecasts from other models; and current observations in order to set initial conditions for the predictions.

The system package view describes the system as a set of software packages (components) to be implemented and deployed for operations. The IMRCP system package view divides the system packages into package types, which include data collection, models, forecasting, forecasts, decision support, and interface. The view also illustrates the relationships between these packages and external sources. The functional view and the system package view provide a context for the development and structuring of the system requirements. Specific requirements for the IMRCP can be found in the *Integrated Modeling for Road Condition Prediction System Requirements Specifications* (SRS).

Based on the requirements determined in the SRS, the system architecture is created using a set of architectural views in the *Integrated Modeling for Road Condition Prediction System Architecture Description* (SAD). The composition view describes the system in terms of sets of software components and their relationships. The process view describes the system in terms of data processing functions and flows. The deployment view describes the system in terms of the deployment of components to computing devices or nodes.

## FUSER ROLES

The IMRCP SAD carried over a description of IMRCP end users from the SRS that was used to develop the overall system architecture. IMRCP design and implementation adds additional user roles for administrating the system and for administrating IMRCP road network models.

A system administrator is an operating system user that can set and elevate system privileges, modify IMRCP configuration settings, and manage IMRCP user privileges.

An IMRCP administrator is a user of the IMRCP system that can create and edit road networks and access all of the interfaces available to an IMRCP user. IMRCP administrator privileges are assigned by system administrators.

An IMRCP user has access to the Map, Scenarios, Reports, and User Settings web interfaces. IMRCP user privileges are assigned by system administrators.

## CHAPTER 3. DESIGN DESCRIPTION

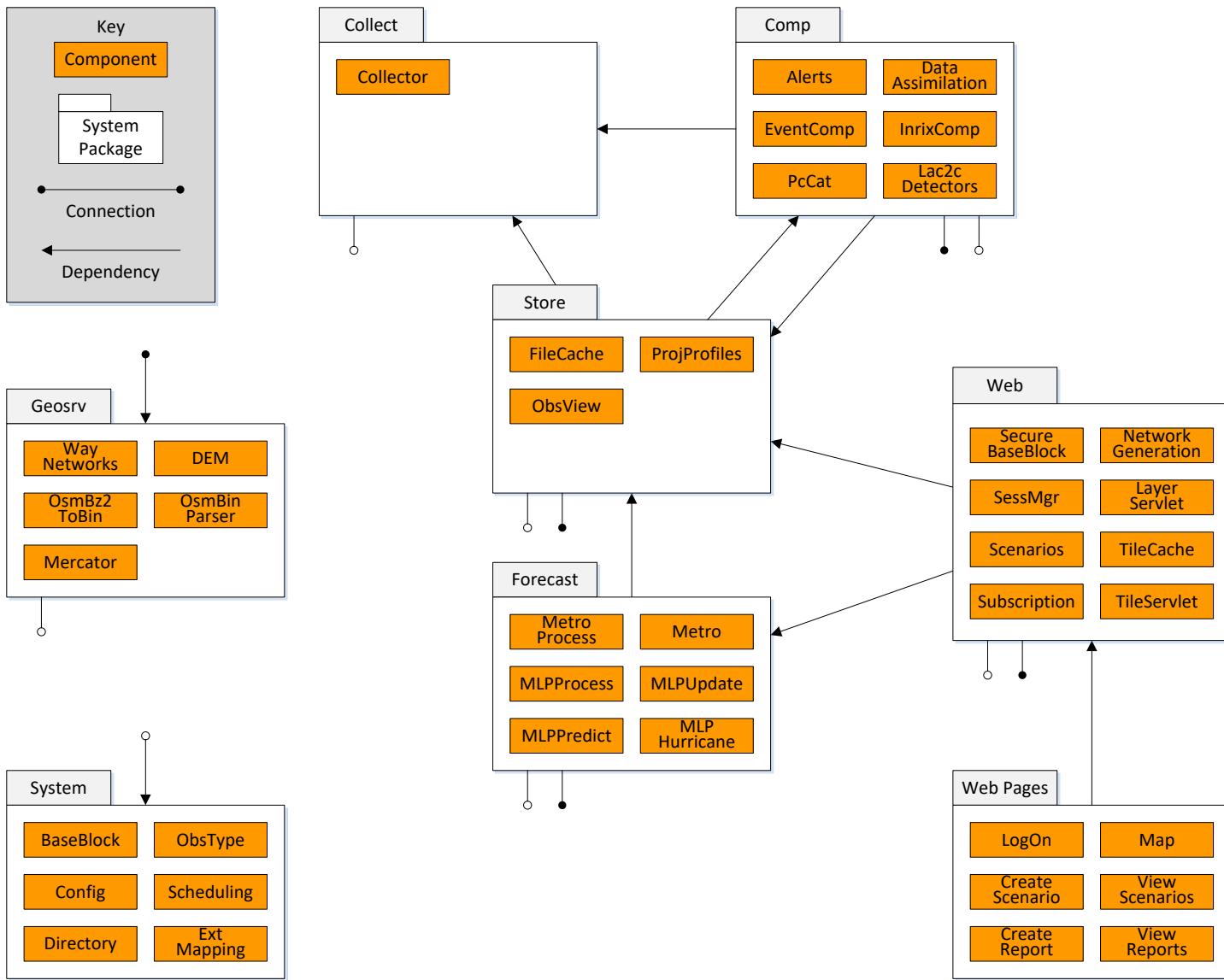
The IMRCP system design is described here as a set of subsystem packages. These subsystem packages were previously identified and described in the System Architecture Description (SAD) composition view, and are used in this design description as organizing entities for the details of the system design. The Unified Modeling Language (UML)<sup>1</sup> package and activity diagrams are used to illustrate the technical concepts and maintain continuity with the SAD. The integrated high-level composition view of the system is shown as a package diagram in Figure 1, and the overall processing is illustrated with the activity diagram in Figure 2. The deployment view from the SAD is filled out with more detail in allocating subsystem packages to particular computing devices on UML deployment diagrams. The requirements from which the design is derived are documented in the *Integrated Modeling for Road Condition Prediction System Requirements Specification*.<sup>2</sup>

Further supporting design details can be found in the appendices and the IMRCP GitHub repository at <https://github.com/OSADP/IMRCP>.

---

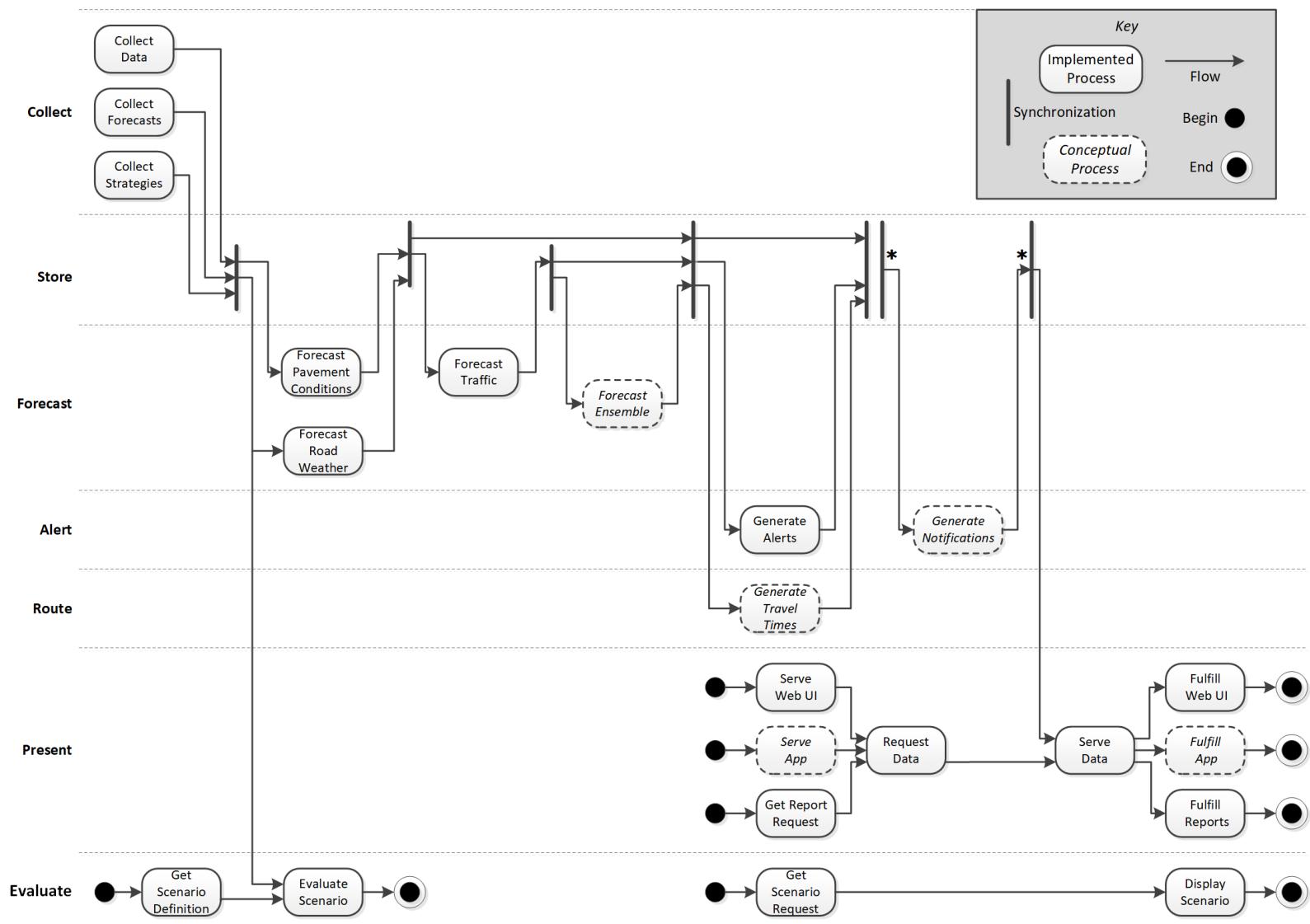
<sup>1</sup> Object Management Group. *OMG Unified Modeling Language™*, Version 2.5. OMG Document Number: formal/2015-03-01. Normative Reference: <http://www.omg.org/spec/UML/2.5>. Available at: <https://www.omg.org/spec/UML/2.5/PDF>, last referenced March 27, 2019.

<sup>2</sup> Leidos, Integrated Modeling for Road Condition Prediction System Requirements. 2016. Unpublished working paper developed under FHWA Contract DTFH61-12-D-00050, Task Order 5022, Integrated Modeling for Road Condition Prediction.



Source: FHWA.

**Figure 1. Diagram. Integrated Modeling for Road Condition Prediction Composition.**



Source: FHWA.

**Figure 2. Diagram. Integrated Modeling for Road Condition Prediction Process.**

## COLLECT

The data collection (Collect) package collects data needed for IMRCP computations and forecast models. Collect modules depend on external systems for making the data available for collection. Collection interfaces, formats, and intervals are determined by the external data sources.

**Appendix A****Error! No bookmark name given.** lists the data availability for the systems accessed by the Collect modules. The package is designed to be extensible and will accommodate new collection modules as new sources are identified and made accessible to the IMRCP modules initially included within the Collect package include:

### Collector

**Background:** IMRCP relies on many different types of data that are collected from various sources. The Collector modules are responsible for gathering the data necessary for other blocks in the system.

**Inputs:** Inputs for Collectors reside in the configuration file and include:

- base URL: The base URL of the source of the collected files
- source file: A formatted string to determine time depended file names
- destination file: A formatted string used to create files and directories to store the collected data
- offset: Schedule offset from midnight UTC in seconds
- period: Number of seconds between collection attempts
- delay: Number of milliseconds until the data collected is valid
- range: Number of milliseconds the data collected is valid
- file frequency: The estimated time in millisecond in-between receiving consecutive files

**Outputs:** Each collector saves the collected files to the configured storage location and sends a notification to the Directory when new files are downloaded so subscribed modules can process or serve the data.

**Process:** Collectors establish a connection to external sources and download available files for the system to use. Depending on the source the method to obtain the data varies, but the outcome is the same.

**Dependencies:** Collectors are dependent on their external source and an Internet connection.

**Known Instances:** ADCIRCEast, AHPSFcst, AHPSObs, CAP, GFS, LAc2c, LAc2cfEU, LADOTD511Line, LADOTD511Point, NDFDQpf, NDFDSky, NDFDTd, NDFDTemp, NDFDWspd, NHC, OhGoConstruction, OhGoIncidents, OhioInrix, Radar, RadarPrecip, RAP, RTMA, WxDE

### ***ADCIRCEast***

The ADCIRCEast collector gets Storm surge and Tide forecast data from the NOAA's Meteorological Development Laboratory. ADCIRC forecasts are provided in GRIB2 format and saved to the configured storage location.

### ***AHPSFcst***

The AHPSFcst collector gets hydrological forecast data from the National Weather Service's Advanced Hydrologic Prediction Service. The maximum 24 hour forecasts are provided in Shapefiles and saved to the configured storage location.

### ***AHPSObs***

The AHPSObs collector gets hydrological observed data from the National Weather Service's Advanced Hydrologic Prediction Service. The currently observed data are provided in Shapefiles and saved to the configured storage location.

### ***CAP***

The CAP collector gets weather alerts published by the National Weather Service's Common Alerting Protocol for the entire United States of America. The current alerts are provided in a tar.gz file that contains Shapefiles and are saved to the configured storage location.

### ***GFS***

The GFS collector gets long term atmospheric weather forecasts from the National Weather Service's Global Forecast System. GFS forecasts are provided in GRIB2 format and before they are saved to the configured storage location unused observation type data grids are filtered out of the file to save space.

### ***LAc2c***

The LAc2c collector connects directly to LADOTD's center to center feed and downloads the traffic detector XML file. The XML files are gzipped, saved to the configured storage location, and then processed by the Computational package into IMRCP's binary traffic speed file format.

### ***LAc2cfEU***

The LAc2cfEU collector connects directly to LADOTD's center to center feed and downloads the full event update XML file. The XML files are gzipped, saved to the configured storage location, and then processed by the Computational package into IMRCP's incident and work zone CSV file format.

### ***LADOTD511Line***

The LADOTD511Line collector gets incident and work zone data associated with a line string from LADOTD's 511 system. The line string feed is a single file that contains every event

logged to the system, with new records continually being appended to the file. IMRCP keeps a copy of the most recent version of the file and saves new events and updates to events in IMRCP's incident and work zone CSV file format to the configured storage location.

#### ***LADOTD511Point***

The LADOTD511Point collector gets incident and work zone data associated with a point from LADOTD's 511 system. The point feed is a single file that contains every event logged to the system with new records continually being appended to the file. IMRCP keeps a copy of the most recent version of the file and saves new events and updates to events in IMRCP's incident and work zone CSV file format to the configured storage location.

#### ***NDFDQpf***

The NDFDQpf collector gets Quantitative Precipitation Forecast (QPF) data from the National Weather Service's National Digital Forecast Database. NDFD forecasts are provided in GRIB2 format and saved to the configured storage location.

#### ***NDFDSky***

The NDFDSky collector gets cloud cover forecast data from the National Weather Service's National Digital Forecast Database. NDFD forecasts are provided in GRIB2 format and saved to the configured storage location.

#### ***NDFDTd***

The NDFDTd collector gets dew point forecast data from the National Weather Service's National Digital Forecast Database. NDFD forecasts are provided in GRIB2 format and saved to the configured storage location.

#### ***NDFDTemp***

The NDFDTemp collector gets air temperature forecast data from the National Weather Service's National Digital Forecast Database. NDFD forecasts are provided in GRIB2 format and saved to the configured storage location.

#### ***NDFDWspd***

The NDFDWspd collector gets wind speed forecast data from the National Weather Service's National Digital Forecast Database. NDFD forecasts are provided in GRIB2 format and saved to the configured storage location.

#### ***NHC***

The NHC collector gets hurricane cone and track forecast data from the National Hurricane Center. Hurricane cone and track forecasts are provided in a .zip archive in .kmz format and saved to the configured storage location.

### *OhGoConstruction*

The OhGoConstruction collector gets work zone data from ODOT's OhGo API. The XML files are gzipped, saved to the configured storage location, and then processed by the Computational package into IMRCP's incident and work zone CSV file format.

### *OhGoIncidents*

The OhGoIncidents collector gets incident data from ODOT's OhGo API. The XML files are gzipped, saved to the configured storage location, and then processed by the Computational package into IMRCP's incident and work zone CSV file format.

### *OhioInrix*

The OhioInrix collector gets traffic speed data from Inrix's Segment Speeds API. The entire state of Ohio cannot be downloaded in a single call to the API because of limits imposed by Inrix. Therefore, the state is broken up into regions, and a JSON Object containing the data is downloaded for each region and added to a JSON Array that is gzipped and saved to a single file in the configured storage location. The Computational package then converts the data into IMRCP's binary traffic speed file format.

### *Radar*

The Radar collector gets radar reflectivity data from the National Weather Service's Multi-Radar Multi-Sensor product. MRMS observation files are provided in GRIB2 format and saved to the configured storage location.

### *RadarPrecip*

The RadarPrecip collector gets precipitation rate data from the National Weather Service's Multi-Radar Multi-Sensor product. MRMS observation files are provided in GRIB2 format and saved to the configured storage location.

### *RAP*

The RAP collector gets atmospheric weather forecasts from the National Weather Service's Rapid Refresh product. RAP forecasts are provided in GRIB2 format and saved to the configured storage location.

### *RTMA*

The RTMA collector gets real-time atmospheric weather forecasts from the National Weather Service's Real-Time Mesoscale Analysis product. RTMA forecasts are provided in GRIB2 format and saved to the configured storage location.

## ***WxDE***

The WxDE collector gets ESS observation data from a Weather Data Environment subscription. WxDE observations are provided in CSV format and saved to the configured storage location.

## **STORE**

### **FileCache**

**Background:** IMRCP collects and stores a wide variety of data types that have different temporal and spatial extents as well as storage formats. FileCache acts as the base class for all data stores and has generic functions for loading files in and out of memory and serving data through the `getData` function. Child classes use specific implementations of the `FileWrapper` class that understand how to read the different file formats used throughout the system to create observation objects that are temporarily stored in memory.

**Inputs:** Inputs for FileCaches are the data files produced by Collector and Computational modules as well as values that reside in the configuration file and include:

- `subobs`: An array of observation types that this store provides
- `format`: An array of formatted strings used to construct file names available for this store
- `offset`: Schedule offset from midnight UTC in seconds
- `period`: Number of seconds between attempts to remove unused files from memory
- `max forecast`: Number of milliseconds of the longest possible forecast of files loaded by this store
- `file frequency`: The estimated time in millisecond in-between receiving consecutive files
- `append`: A boolean declaring if the store loads files that can be appended to once the file is already in memory
- `limit`: The maximum number of files that can be loaded into memory at a time
- `cachetimeout`: Number of milliseconds a file can be in the cache without being accessed before it is considered unused
- `slashes`: Number of slashes from the end of the file path to the base directory for this store
- `initfiles`: Number of files to attempt to load at system start up

**Outputs:** FileCache modules create observation objects.

**Process:** FileCache modules receive a notification when new data or files are available and attempt to load them into memory. If the cache is full, the two files that are least recently used are removed from the cache to make room for the new file. FileCache modules also regularly check for unused files in the cache on a schedule. Whenever a query for data is made from another module, first the cache is checked to determine if a file is already in memory that is valid for the request. If there isn't a valid file (a file that exists and matches the temporal and geographic query parameters) in the cache, the FileCache determines possible directories to check and finds the “best” forecast file (the most recent valid forecast file that was received at or before the query reference time) based on the query parameters and loads it into memory. Now if

there is a valid file in memory, the query is fulfilled and any observation objects that match the query are returned.

**Dependencies:** FileCache modules depend on Collector and Computational modules to collect and save files to the server.

**Known Instances:** ADCIRCStore, AHPSFcstStore, AHPSObsStore, CAPStore, GFSAlertsStore, GFSPcCatStore, GFSStore, LADOTD511LineStore, LADOTD511PointStore, LADOTDEventsStore, LaTrafficSpeedStore, MetroStore, MLPStoreLA, MLPStoreOH, MRMSPcCatStore, NDFDPcCatAlertsStore, NDFDPcCatStore, NDFDQpfStore, NDFDSkyStore, NDFDTdStore, NDFDTempStore, NDFDWspdStore, NHCStore, OhGoConstructionStore, OhGoIncidentsStore, OhioTrafficSpeedStore, RadarPrecipStore, RadarStore, RAPAlertsStore, RAPPcCatAlertsStore, RAPPcCatStore, RAPStore, RTMAStore, TPVTDASStore, TSSRFDASStore, WxDEStore

#### ***ADCIRCStore***

The ADCIRCStore creates forecast combined storm surge and tide observations from the original GRIB2 files from the National Weather Service's ADCIRC model on a 2.5 km grid.

#### ***AHPSFcstStore***

The AHPSFcstStore creates forecast flood stage observations from the Advanced Hydrologic Prediction Service's Shapefiles for each AHPS station and forecast pavement state and flooded road event observations for AHPS station that have JSON inundation files.

#### ***AHPSObsStore***

The AHPSObsStore creates observed flood stage observations from the Advanced Hydrologic Prediction Service's Shapefiles for each AHPS station and observed pavement state and flooded road event observations for AHPS station that have JSON inundation files.

#### ***CAPStore***

The CAPStore creates observed and forecasted weather alerts for the entire United States of America from the Common Alerting Protocol's Shapefiles.

#### ***GFSAlertsStore***

The GFSAlertsStore creates forecasted weather alerts from IMRCP's CSV observation files.

#### ***GFSPcCatStore***

The GFSPcCatStore creates forecasted precipitation category observations on the same 25 km grid as GFS from IMRCP's gridded binary observation files.

### *GFSStore*

The GFSStore creates forecasted air temperature, dew point, precipitation rate, precipitation type, relative humidity, snow depth, surface pressure, visibility, wind gust speed, and wind speed observations from the filtered Global Forecast System's GRIB2 files on a 25 km grid.

### *LADOTD511LineStore*

The LADOTD511LineStore creates incident and work zone observations from IMRCP's incident and work zone CSV files.

### *LADOTD511PointStore*

The LADOTD511PointStore creates incident and work zone observations from IMRCP's incident and work zone CSV files.

### *LADOTDEventsStore*

The LADOTDEventsStore creates incident and work zone observations from IMRCP's incident and work zone CSV files.

### *LaTrafficSpeedStore*

The LaTrafficSpeedStore creates traffic speed observations from IMRCP's binary traffic speed files.

### *MetroStore*

The MetroStore creates forecasted liquid depth, pavement state, pavement temperature, subsurface temperature, and snow depth observations for roadway segments from IMRCP's binary METRo files.

### *MLPStoreLA*

The MLPStoreLA creates forecasted traffic speed observations for roadway segments from IMRCP's MLP CSV files.

### *MLPStoreOH*

The MLPStoreOH creates forecasted traffic speed observations for roadway segments from IMRCP's MLP CSV files.

### *MRMSPcCatStore*

The MRMSPcCatStore creates inferred precipitation category observations on the same 1 km grid as MRMS from IMRCP's gridded binary observation files.

### *NDFDPcCatAlertsStore*

The NDFDPcCatAlertsStore creates forecasted precipitation category alerts from IMRCP's CSV observation files.

### *NDFDPcCatStore*

The NDFDPcCatStore creates inferred precipitation category forecasts on the same 2.5 km grid as NDFD from IMRCP's gridded binary observation files.

### *NDFDQpfStore*

The NDFDQpfStore creates forecasted precipitation rate observations from the original GRIB2 files from the National Digital Forecast Database's on a 2.5 km grid.

### *NDFDSkyStore*

The NDFDSkyStore creates forecasted cloud cover observations from the original GRIB2 files from the National Digital Forecast Database's on a 2.5 km grid.

### *NDFDTdStore*

The NDFDTdStore creates forecasted dew point observations from the original GRIB2 files from the National Digital Forecast Database's on a 2.5 km grid.

### *NDFDTempStore*

The NDFDTempStore creates forecasted air temperature observations from the original GRIB2 files from the National Digital Forecast Database's on a 2.5 km grid.

### *NDFDWspdStore*

The NDFDWspdStore creates forecasted wind speed observations from the original GRIB2 files from the National Digital Forecast Database's on a 2.5 km grid.

### *NHCStore*

The NHCStore creates forecasted tropical storm category, track, and cone observations from the National Hurricane Center's original KMZ files for the Atlantic, Eastern Pacific, and Central Pacific basins.

### *OhGoConstructionStore*

The OhGoConstructionStore creates work zone observations from IMRCP's incident and work zone CSV files.

### *OhGoIncidentsStore*

The OhGoIncidentsStore creates incident observations from IMRCP's incident and work zone CSV files.

### *OhioTrafficSpeedStore*

The OhioTrafficSpeedStore creates traffic speed observations from IMRCP's binary traffic speed files.

### *RadarPrecipStore*

The RadarPrecipStore creates precipitation rate observations from the original GRIB2 files from the National Weather Service's MRMS on a 1 km grid.

### *RadarStore*

The RadarStore creates radar reflectivity observations from the original GRIB2 files from the National Weather Service's MRMS on a 1 km grid.

### *RAPAlertsStore*

The RAPAlertsStore creates forecasted weather alerts from IMRCP's CSV observation files.

### *RAPPcCatAlertsStore*

The RAPPcCatAlertsStore creates forecasted precipitation category alerts from IMRCP's CSV observation files.

### *RAPPcCatStore*

The RAPPcCatStore creates forecasted precipitation category observations from IMRCP's gridded binary observation files.

### *RAPStore*

The RAPStore creates forecasted precipitation rate, precipitation type, surface pressure, visibility, and wind speed observations from the National Weather Service's RAP original GRIB2 files on a 13 km grid.

### *RTMAStore*

The RTMAStore creates forecasted air temperature, cloud cover, dew point, surface pressure, visibility, wind direction, wind gust speed, and wind speed observations from the original GRIB2 files from the National Weather Service's RTMA on a 2.5 km grid.

### *TPVTDAStore*

The TPVTDAStore creates kriged pavement temperature observations from IMRCP's gridded binary observation files on a 1 km grid.

### *TSSRFDAStore*

The TSSRFDAStore creates kriged subsurface temperature observations from IMRCP's gridded binary observation files on a 1 km grid.

### *WxDEStore*

The WxDEStore creates air temperature, dew point, liquid depth, pavement state, pavement temperature, precipitation category, precipitation rate, precipitation type, relative humidity, subsurface temperature, surface pressure, visibility, wind direction, wind gust speed, and wind speed observations from Weather Data Environment's CSV subscription files for ESS stations.

## **ObsView**

**Background:** The Observation View module is the main endpoint for other modules within the system to request specific observation types from the data Store. Through the Directory, it knows the observation types that each store provides. The Observation View aggregates all the requested data from the individual stores and returns the observation objects to the requesting modules.

**Inputs:** Inputs to the Observation View Store module include requests from other modules within the system for specific observation types within specific geographic and temporal domains.

**Outputs:** The data requested by other modules within the system are the outputs of the Observation View module.

**Process:** Modules within the IMRCP system request specific observation type data within a specific geographic and temporal domain from the Observation View module using the `getData` method. The Observation View module then requests the specified data from the modules containing those data types. The modules containing the specified data types return the data to the Observation View module and the Observation View module returns the data to the module that requested it.

**Dependencies:** The Observation View module is dependent on the other modules in the data Store.

**Known Instances:** ObsView

### **ProjProfiles**

**Background:** Geodetic information for observations and forecasts are stored in different projection coordinate systems throughout the data Store. To save memory and processing time,

projection profiles common among the sources are cached in memory . For example, all data files from RTMA can use the same projection. This singleton manages the creation and caching of the system's projection profiles.

**Inputs:** ProjProfiles uses projection coordinate system grids.

**Outputs:** ProjProfiles returns projection profile objects.

**Process:** When a data file is loaded into memory by a FileCache that requires a projection profile to determine its spatial information, the projection coordinate system grid is passed to ProjProfiles. If a projection profile for that grid does not exist, ProjProfiles creates and caches the projection profile. Now ProjProfiles can serve that projection profile to data files in the system.

**Dependencies:** The ProjProfiles singleton does not have any dependencies.

**Known Instances:** ProjProfiles singleton

## SYSTEM

The System package provides the base system operation modules and utilities. These components are the first to be instantiated on system startup and provide services that enable other components to interact within the system.

### Directory

**Background:** The Directory is the main system component that initializes the system by identifying and managing BaseBlocks and services available within the system. Use of a Directory component enhances system extensibility and maintainability relative to a closed system.

**Inputs:** Inputs for the Directory reside in the configuration file and include:

- classes: A string array that contains the BaseBlocks to instantiate and manage. The strings come in pairs, the first being the fully qualified Java name, and the second the name for the instance. An “@” can be placed at the front of the fully qualified Java name to represent a BaseBlock that will be instantiated by the web server’s container manager instead of the Directory. An “#” can be placed at the front of the fully qualified Java name to have the Directory ignore that entry.
- prefer: A string array that contains sources that provide observation types that other sources also provide. Each string in the array will then have its own configuration value whose key is the source name and the value is an integer representing its preference. A lower number has higher preference.
- preferprecip: A string array that contains sources that provide precipitation rate observations and forecasts. Each string in the array will then have its own configuration value whose key is the source name with “\_precip” appended at the end and the value is an integer representing its preference. A lower number has higher preference.

**Outputs:** The Directory provides notifications from one BaseBlock to another.

**Process:** Directory is the first servlet instantiated by the web server's container manager and initializes the rest of the system. First the Config and Scheduling components are instantiated. Next the BaseBlock instances in the classes array are instantiated, initialized, registered to the Directory, and queued to start. Since some BaseBlocks depend on data from other BaseBlocks to start, the Directory dynamically determines the instantiation order based off of the BaseBlock's configuration, ensuring that BaseBlocks are only started after their dependencies have finished their start method.

**Dependencies:** The Directory is dependent on Config and the web server's container manager.

**Known Instances:** Directory

## ObsType

**Background:** The different observation types in the system have metadata associated with them including name, integer value, English and metric units, and enumerated values that are managed by the ObsType class. It also contains the RangeRule objects that are used to bucket observation values to aid with presentation.

**Inputs:** Inputs for ObsType reside in the configuration file and include:

- lrain: light rain threshold in kg/(m<sup>2</sup>\*s)
- mrain: medium rain threshold in kg/(m<sup>2</sup>\*s)
- lsnow: light snow threshold in kg/(m<sup>2</sup>\*s)
- msnow: medium snow threshold in kg/(m<sup>2</sup>\*s)
- raintemp: temperature for rain threshold in K
- snowtemp: temperature for snow threshold in K
- lightrainmm: light rain threshold in mm/hr
- medrainmm: medium rain threshold in mm/hr
- lightsnowmm: light snow threshold in mm/hr
- medsnowmm: medium snow threshold in mm/hr
- rules: a string array of observation types that have configured RangeRules

**Outputs:** The ObsType class returns requested metadata associated with the observation types.

**Process:** Upon receiving the first request from another component, configuration values are retrieved from Config. All of the observation types and enumerated values are defined in the static block of the class. System modules can query ObsType for observation type's name, English or metric units, enumerated values, and RangeRules.

**Dependencies:** ObsType is dependent on Config.

**Known Instances:** n/a – a static class that does not get instantiated

## Configuration [Config]

**Background:** The Configuration component contains the configuration parameters for other system components providing developmental and operational flexibility. For example, by editing the configuration file different components can be instantiated and resource locations can be changed without recompiling the source code.

**Inputs:** The Configuration component input is the name of the requesting class, the instance name, the key, and the default value which are used to look up configured values from a configuration file.

**Outputs:** Config returns integer, string, or string array values from the configuration file.

**Process:** The Configuration component takes the class name, instance name, and key name for a configured value. If a match for the requested value is found in the configuration file, it is returned; otherwise the default value defined by the class requesting the value is returned.

**Dependencies:** The Configuration component is dependent on the configuration file.

**Known Instances:** Config

## Scheduling

**Background:** The Scheduling component provides scheduling services to other system components, such as collecting forecast files on a regular fixed interval. The Scheduling and Configuration components work together to orchestrate the process-related operations of the system.

**Inputs:** The inputs to Scheduling interfaces include the Runnable component (in most cases a BaseBlock) to be scheduled, period of execution and midnight offset, or delay time.

**Outputs:** Scheduling returns a schedule id to ensure that only the component that requested the scheduled tasks to be created can request them to be canceled.

**Process:** Scheduling uses a built in Java thread pool to execute tasks. During the start method of a BaseBlock that has a task to be executed on a regular schedule, it requests a schedule to be created and receives a schedule id from Scheduling. The schedule id is used as a parameter to request a BaseBlock's task to be canceled.

**Dependencies:** The Scheduling component does not have any dependencies.

**Known Instances:** Scheduling

## ExtMapping

**Background:** The External Mapping component provides mapping and look up capabilities for system objects from external source to their corresponding IMRCP system object. This allows

IMRCP to associate observations and forecasts from external systems to the correct system object inside of IMRCP.

**Inputs:** The ExtMapping module receives object identifiers from external system.

**Outputs:** The ExtMapping module returns IMRCP object identifiers.

**Process:** A System Administrator must generate the appropriate mapping files for each external system to associate external system objects to IMRCP system objects. An example of a necessary mapping would be Inrix XD segments to IMRCP ways objects. At system start up, the ExtMapping reads the configured mapping files and creates the corresponding lookup map in memory so other modules can request an IMRCP object identifier given an external object identifier.

**Dependencies:** The ExtMapping module is dependent on a System Administrator creating mapping files and on the Configuration component to specify the file paths of the mapping files.

**Known Instances:** ExtMapping

## COMPUTATION

The Computation package contains modules that use algorithms to compute new observations based on other system observations and conditions. For example, alert generation is based on a set of rules in the form of logical statements about transportation system conditions. A “slick pavement” alert could be based on a measurement of ice on a roadway segment, or on an assessment of pavement surface temperature less than a configured threshold temperature with precipitation present along a roadway segment. The level of alert (i.e., advisory, watch, or warning) depends on the confidence and likelihood of the conditions. An observation or measurement of a condition would merit a warning, whereas an assessment based on future regional conditions might only warrant an advisory.

### Alerts

**Background:** Alerts modules can generate alerts for configured situations related to exceptional weather, hydrological, or traffic observations/predictions.

**Inputs:** Alerts modules use data from the data Store and rule sets found in the configuration file.

**Outputs:** Alert modules save generated alerts to a file and send a notification to the Directory that new alerts are available.

**Process:** Alerts modules subscribe to data Stores to receive notifications when new data to process are available. The data are gathered and compared with a set of rules in the form of logical statements defined in the configuration file about transportation system conditions. If the data meets the requirements of a rule, an alert is generated and saved to a file. Rules for Alerts were defined for the sources available at the time of implementation. Different logical statements can be defined to meet the needs of future sources.

**Dependencies:** Alerts modules are dependent on the data Store.

**Known Instances:** GFSAlerts, GFSPcCatAlerts, NDFDPcCatAlerts, RAPAlerts, RAPPcCatAlerts

## DataAssimilation

**Background:** IMRCP uses algorithms based on observed and forecasted conditions to predict weather and roadway conditions for large areas. Some external data sources and types only provide limited spatial coverage of conditions. To be able to make predictions for the desired spatial extents in IMRCP, Data Assimilation techniques are used to combine a numerical model, namely a class of Kriging algorithms, with observations to have a more complete coverage of conditions. Currently these techniques are used to approximate surface and sub-surface temperatures across road networks using observed values from ESS.

**Inputs:** DataAssimilation modules use data from the data Store.

**Outputs:** DataAssimilation modules save computed approximate values to a file and send a notification to the Directory that new data is available.

**Process:** DataAssimilation modules run on a fixed schedule (defined by the configured values for the period and midnight offset). Each execution cycle, the most recent data is gathered from the data Store to process. The data assimilation algorithm (inverse weighted distance) is run on the gathered data and produces a grid of approximate values of the desired observation type.

**Dependencies:** Data Assimilation modules are dependent on the data Store.

**Known Instances:** TPVTDA, TSSRFDA

## EventComp

**Background:** External sources that provide incident and roadwork data do not have a common standard or interface used to format and maintain active events. EventComp modules process the raw event data feeds to maintain a list of active events and save any updates to an event, including once the event is no longer active.

**Inputs:** EventComp modules receive a notification from the Directory when a new event data feed file is available to process.

**Outputs:** EventComp modules save event observations to a file and send a notification to the Directory that new data is available.

**Process:** Once a notification is received that a new event data feed file is available, EventComp modules process that file. The event records in the data feed are compared to the module's record of active events. New events are added to the module's list of active events and existing events are checked for updates or closure. Any change to the active events list is saved to a file.

**Dependencies:** EventComp modules are dependent on Collector blocks that collect event data feeds.

**Known Instances:** LAc2cEventsComp, LADOTD511Line, LADOTD511Point, OhGoConstruction, OhGoIncidents

## InrixComp

**Background:** The data Store expects traffic speed data to be stored in a compact binary file. Since Inrix traffic speed data is received in multiple JSON files each collection cycle, the InrixComp module processes those files into the desired binary format.

**Inputs:** InrixComp modules receive a notification from the Directory when a new set of Inrix traffic speed data files is available to be processed.

**Outputs:** InrixComp modules save traffic speed records to a binary file and send a notification to the Directory that new data is available.

**Process:** InrixComp modules parse through Inrix traffic speed data JSON files extracting the Id, timestamp, and speed value for each record and save those values to a binary file.

**Dependencies:** InrixComp modules are dependent on Collector blocks that collect Inrix traffic speed data.

**Known Instances:** OhioInrix

## Precipitation Category [PcCat]

**Background:** Many weather forecast products provide precipitation forecasts as a category, instead of a numeric rate. IMRCP collects raw precipitation rate and precipitation type data from different sources, so to have easily understandable precipitation forecasts, rates and types are converted into categories.

**Inputs:** PcCat modules receive a notification from the Directory when new weather forecast data is available to be processed.

**Outputs:** PcCat modules save precipitation category observations and forecasts in a gridded binary file and send a notification to the Directory that new data is available.

**Process:** PcCat modules convert precipitation rate and type observations and forecasts into easily understandable precipitation categories. If the precipitation type is not provided by the source of the precipitation rate, air temperature is used to infer the type. Since the observations and forecasts source data is in the form of gridded data, a precipitation category value is computed for each cell in the grid, and saved to a separate IMRCP gridded data binary file in the configured location.

**Dependencies:** PcCat modules are dependent on Collector blocks that collect precipitation rate, precipitation type, and air temperature data.

**Known Instances:** PcCat

### LAc2cDetectors

**Background:** The MLP traffic prediction model expects traffic speed data in 5 minute averages. Traffic speed data is collected every minute from the Louisiana C2C feed so the LAc2cDetectors computation module averages the speeds collected and saves the values in another file.

**Inputs:** The LAc2cDetectors module receives a notification from the Directory when new traffic speed data is available from the LAc2c Collector.

**Outputs:** The LAc2cDetectors module saves 5 minute traffic speed averages to a file and sends a notification to the Directory that new data is available.

**Process:** When the LAc2cDetectors module is notified that new traffic speed data is available it determines if it is the correct minute offset to roll up the most recent files into a 5 minute average. For each segment in the data files, it determines the average speed and writes that value to a file.

**Dependencies:** The LAc2cDetectors module is dependent on the LAc2c Collector.

**Known Instances:** LAc2cDetectors

## FORECAST

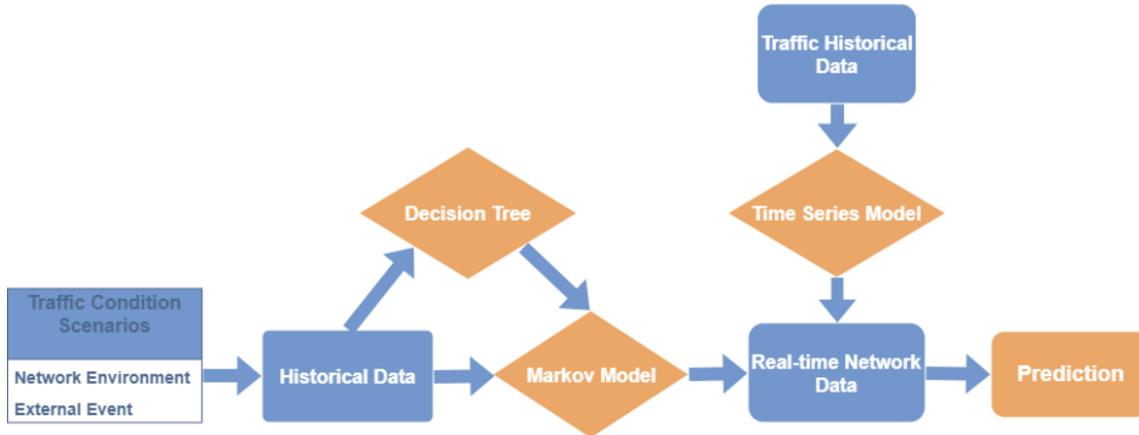
The Forecast package contains modules that produce real-time near term and long-term forecasts. There are two prominent forecasting models included in IMRCP: MLP and METRo.

### Machine Learning-based Prediction

The Machine Learning-based Prediction (MLP) package predicts traffic network conditions given a set of system variables that include weather, work zones, incidents, and special events. MLP is a comprehensive, data-driven prediction module that uses a Markov process to explicitly characterize the probabilistic transition between traffic states under different external conditions (e.g., weather, incidents).

A Markov stochastic process is used to model the randomly evolving system with the assumption that future states depend only on the current state. A Markov transition matrix consists of a set of probabilities that are used to represent the transition probabilities between different traffic states. It is built based on archived data and can be applied online using real-time feeds to generate precise prediction models and results. With a calibrated Markov model, the probability of transition between traffic states under different external conditions can be computed. The algorithm considers that the environment variables (e.g., weather) and the external event variables (e.g., incidents) affect the transition probability matrices between different traffic states. A decision tree model constructed based on the historical data is used to determine whether the external events will affect the traffic states and whether the Markov model will be applied. The time series model takes online data as input to reflect the most current traffic

conditions observed in the field. This makes the prediction model robust, particularly during special conditions that have traffic patterns that are different from regular scenarios.



Source: FHWA.

**Figure 3. Proposed Machine Learning-based Prediction Network Model Algorithm.**

Variables used in the IMRCP MLP model are presented in Table 1. Variables and State Definitions for Machine Learning-based Prediction. A total of 13 variables are categorized into four groups, including network environment, external event, and traffic condition.

**Table 1. Variables and State Definitions for Machine Learning-based Prediction.**

Node Group	Variable	States	State Definitions
<b>Group 1: Network Environment</b>	Direction	-Eastbound -Southbound -Westbound -Northbound	
	Weather	-Clear -Light rain, clear visibility -Light rain, reduced visibility -Light rain, low visibility -Moderate rain, clear visibility -Moderate rain, reduced visibility -Moderate rain, low visibility -Heavy rain, reduced visibility -Heavy rain, low visibility -Light snow, clear visibility -Moderate snow, reduced visibility -Heavy snow, reduced visibility -Heavy snow, low visibility	00mm/h; visibility >3300ft < 2.5mm/h; > 3300ft < 2.5mm/h; 330 - 3300ft < 2.5mm/h; < 330ft 2.5 - 7.6mm/h; > 3300ft 2.5 - 7.6mm/h; 330 - 3300ft 2.5 - 7.6mm/h; < 330ft ≥ 7.6mm/h; 330 - 3300ft ≥ 7.6mm/h; < 330ft ; >3300ft* ; 1650 - 3300ft* ; 330 - 1650ft* ; < 330 ft*
	DayOfWeek	-Weekend -Weekday	Saturday, Sunday Monday - Friday
	TimeOfDay	-Morning -AM peak -Off-peak -PM peak	1AM - 6AM (5 hrs) 6AM - 10 AM (4hrs) 10AM - 4PM (6hrs) 4PM - 8PM (4hrs)

<b>Node Group</b>	<b>Variable</b>	<b>States</b>	<b>State Definitions</b>
		-Night	8PM - 1AM (5 hrs)
<b>Group 2: External Event</b>	IncidentDownstream	-No incident -Incident	
	IncidentOnLink	-No incident -Incident	
	IncidentUpstream	-No incident -Incident	
	Work zone	-No work zone -Work zone	
	RampMetering	-No ramp metering -Ramp metering	
	SpecialEvents	-Special event Type 1 -Special event Type 2 -Special event Type 3 -No special event	Local special events as defined by the agency, such as football games
<b>Group 3: Traffic Management Strategies</b>	VSL	-0 -1	- vsl off - vsl on
	Contraflow	- 0 - 1 - 2 - 3 - 4	- no contraflow - add one lane - add two lanes - close one lane - close two lanes
<b>Group 4: Traffic Condition</b>	Speed (mph)		

Source: FHWA.

\* Snowfall's intensity is determined by visibility.

### *Machine Learning-based Prediction Scenario Identifier*

**Background:** The MLP Scenario Identifier module within the MLP package contains two parts: a Scenario Identifier and a Markov Model Generator, the latter of which is computed from the archived data. The Scenario Identifier is a submodule that identifies the specific scenario the current traffic condition belongs to and then selects or generates corresponding models for traffic state prediction. The Markov Model is developed based on historical traffic state transitions and is used to explain the stochastic evolutions of traffic states for each link.

**Inputs:** Inputs to the MLP Scenario Identifier module include weather, roadwork, incidents, traffic control strategies, special events, and traffic state data. These datasets are obtained from

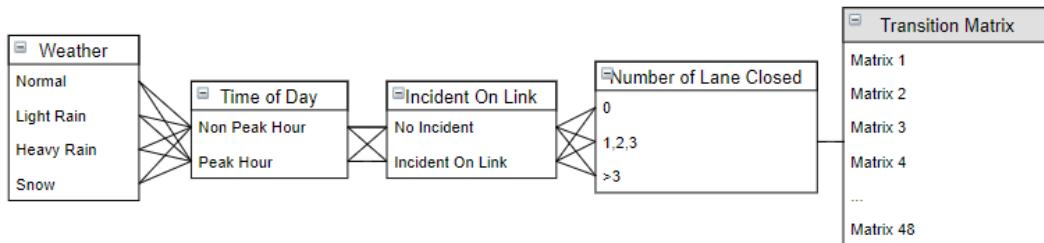
the data Store using the `getData` method from `ObsView`. They are categorized into the four groups shown in Table 4: Network Environment, External Events, Traffic Management Strategies and Traffic Condition.

**Table 2. Machine Learning-based Prediction Scenario Identifier Input Categories.**

Group 1: Network Environment	Group 2: External Events	Group 3: Traffic Management Strategies	Group 4: Traffic Condition
<ul style="list-style-type: none"> <li>• Direction</li> <li>• Weather</li> <li>• Time of day</li> <li>• Day of week</li> </ul>	<ul style="list-style-type: none"> <li>• Incident downstream</li> <li>• Incident on this link</li> <li>• Incident upstream</li> <li>• Work zone</li> <li>• Ramp metering</li> <li>• Special events</li> </ul>	<ul style="list-style-type: none"> <li>• VSL</li> <li>• Contraflow</li> </ul>	<ul style="list-style-type: none"> <li>• Speed</li> </ul>

Source: FHWA.

**Outputs:** The MLP Scenario Identifier module generates clustered groups of links and different traffic states based on historical traffic data. The Markov Model computes a transition probability matrix between different states for each scenario of system variables. Figure 4. Machine Learning-based Prediction Estimator Output Example. provides an example of the output from the MLP Scenario Identifier.



Source: FHWA.

**Figure 4. Machine Learning-based Prediction Estimator Output Example.**

**Process:** This module uses an approach based on the K-means clustering algorithm and pattern recognition for scenario identification. The K-means clustering method is used to define and categorize the traffic states into different levels of congestion, from free flow to heavily congested. Each of the traffic states is assigned with a distribution of traffic speeds and volumes based on the archived data. Then roadway links are clustered into different groups based on factors such as the link volume pattern and functional type. For each detector and traffic condition scenario (based on the weather, roadwork, traffic and other data inputs), the module computes and estimates the transition matrix that includes the transition probabilities between the traffic states.

**Dependencies:** The MLP Scenario Identifier module depends on the weather, roadwork, incidents, traffic control strategies and traffic state data from data Store.

## *Machine Learning-based Prediction Traffic Predictor*

**Background:** The MLP Traffic Predictor is a module within the MLP package that predicts likely future network traffic states for a specified prediction horizon (such as 15 min, 30 min, 1 hour, or 2 hours) under the specific network environment and external event conditions. The Predictor contains two parts: a One-Shot Predictor and an Online Predictor. Both Predictors use the information on current network link traffic states and other system variables, such as work zones and incidents, as model inputs to predict future network states. The One-Shot Predictor uses past-7 days' traffic state data and the forecasted next 7 days weather data to predict the future traffic state without feeding with real-time traffic data. In the One-Shot Predictor, only the future weather data are used as the scenario identifier. The purpose of the One-Shot Predictor is to provide an approximate prediction of the traffic speed for the next 7 days under forecasted weather conditions. On the other hand, the Online Predictor considers different transition probabilities between traffic states under different external conditions (e.g., weather, incident) and uses the time series model to account for the latest trends and observations from the field. Therefore, it is able to accurately predict traffic state evolution under different external conditions and adjust the prediction based on real-time field observations.

**Inputs:** The MLP Traffic Predictor inputs contain four groups of data. The first group is Network Environment (e.g., weather, time, date). The second is External Events, such as incidents, work zones, and special events. The third group is traffic management strategies, such as variable speed limit (VSL) and contraflow. The fourth group is real-time and archived data of traffic data as clustered into different traffic states. These inputs are similar to the inputs in Table 2**Error! Reference source not found..** All these data inputs are current information on the current network link provided by the data Store. The outputs from the MLP Scenario Identifier also serve as input to this module.

**Outputs:** The MLP Traffic Predictor provides future network traffic conditions (i.e., speed) for the next specified prediction horizon (15 mins, 30 mins, 1 hour or 2 hours). For Oneshot Predictor, the prediction horizon ranges from 1 day to 7 days at a minimum time interval of 15 minutes.

**Process:** A time series model is applied to predict traffic speeds under normal traffic conditions. When the network environment changes or an external event occurs, the decision tree model is applied to determine whether the external events will affect the traffic speed. If yes, then the appropriate Markov transition matrix is used to predict the traffic state for the next 5-min interval. Then the speed estimate is calculated using the means of speed distributions for different traffic states (extracted from the archived data), the time series prediction, and transition matrix probabilities. For each 5-minute time interval prediction, data from the previous 7 days are taken as inputs for the time series model.

**Dependencies:** The MLP Predictor depends on current weather, roadwork, incidents and traffic control strategies from the data Store and the outputs from the MLP Scenario Identifier. Particularly, both archived data and real-time feeds are used.

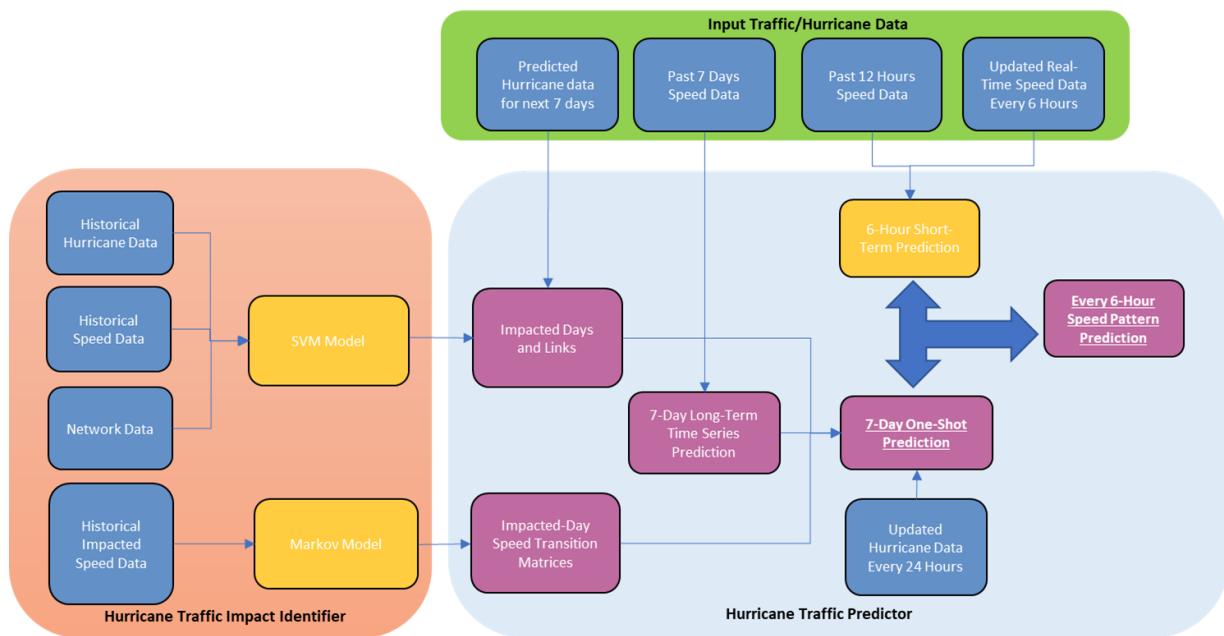
## Machine Learning-based Prediction for Hurricane Traffic

The Machine Learning-based Prediction (MLP) package for Hurricane Evacuation predicts traffic network conditions given a set of system variables that include network environment and hurricane forecast data. The MLP Hurricane Evacuation package contains two classes: MLP Hurricane Traffic Impact Identifier and MLP Hurricane Traffic Predictor.

Similar to the MLP package of non-hurricane scenario, the Markov stochastic method is used to predict the transition probabilities between different traffic states under hurricane conditions. The algorithm considers that the occurrence of hurricane affects the transition probability matrices between different traffic states. A support vector machine (SVM) model based on the historical data is used to determine whether the hurricane will affect the traffic states and therefore whether the Markov model will be applied. The time series model takes online data as input to reflect the most current traffic conditions observed in the field.

Source: FHWA.

Figure 5 shows a graphical representation of the IMRCP MLP for Hurricane Evacuation algorithmic flow. The module first understands whether the traffic state on the link will be affected by the upcoming hurricane and on what days the effect will occur. The module then pulls the historical traffic data during past hurricanes and calculates inputs for the Markov and time series models. The One-shot Predictor gives a prediction of the traffic state for 7 consecutive days after the day the hurricane warning is issued given the historical traffic data, the network environment data, and the hurricane forecast data. New online data feeds are also used as inputs for the prediction in the Online Predictor module to ensure the predicted traffic conditions can best reflect real-time conditions.



Source: FHWA.

**Figure 5. Graphical Representation of the Integrated Model for Road Condition Prediction Machine Learning-based Prediction Process for Hurricane Traffic.**

Variables used in the IMRCP MLP model are presented in Table 3. A total of 14 variables are categorized into three groups, including network environment, external event, and traffic condition.

**Table 3. Variables and State Definitions for Machine Learning-based Prediction for Hurricane Traffic.**

Node Group	Variable	States	State Definitions
<b>Group 1: Network Environment</b>	DayOfWeek	-Weekend -Weekday	Saturday, Sunday Monday - Friday
	Direction	-Eastbound -Southbound -Westbound -Northbound	
	Reference		Roadway name
	Latitude		Latitude GPS coordinate of the link centroid
	Longitude		Longitude GPS coordinate of the link centroid
	Length (m)		Link length
<b>Group 2: Hurricane</b>	Hurricane Status	-1 -2 -3	- TD, EX, SD, SS, LO, DB - TS - HU
	Day after Warning		Number of days after the hurricane warning issued
	MinPressure (millibars)		minimum pressure of hurricane center
	MaxSpeed (knot)		Maximum sustained wind
<b>Group 3: Traffic Condition</b>	Speed (mph)		

Source: FHWA.

#### *Machine Learning-based Prediction Hurricane Traffic Impact Identifier*

**Background:** The MLP Hurricane Traffic Impact Identifier is a module within the MLP hurricane package that contains two parts: an Impacted Links Identifier and a Markov Model Generator, both are computed from the archived data. The Impacted Links Identifier is a submodule within the MLP Hurricane Estimator that identifies the traffic state 1) on which links and 2) on what days after hurricane warning are to be impacted by the hurricane given the network geographic information and the forecasted upcoming hurricane conditions. The Markov

Model is developed based on historical traffic state transitions and is used to explain the stochastic evolutions of traffic states for each impacted link.

**Inputs:** Inputs to the MLP Hurricane Traffic Impact Identifier module include link coordinates, data record time, hurricane information, and traffic state data. These datasets are obtained from the data Store. They are categorized into the three groups shown in Table 3: Network Environment, Hurricane Condition and Traffic Condition.

**Outputs:** The MLP Hurricane Traffic Impact Identifier module generates clustered groups of links with labeled days of impact and different traffic states based on historical traffic and hurricane data. The Markov Model computes a transition probability matrix between different states for links identified as impacted links on impact days.

**Process:** This module uses an approach based on support vector machine (SVM) for impacted links identification. The SVM model is trained using the historical hurricane data, network environment data, and historical traffic data to indicate whether a significant speed drop took place for a specific link under a specific hurricane condition. The inputs for the SVM model are the aggregated speed data during historical hurricane periods. The original speed data are aggregated from 5-minute interval to 1-hour interval and the significant speed drop is defined as the difference between the 24-hour average speed and the minimum speed is greater than 10 mph. The trained SVM model is used to identify if traffic speed on a link will be significantly impacted by an upcoming hurricane given the network environment data and the forecasted hurricane data.

**Dependencies:** The MLP Hurricane Traffic Impact Identifier module depends on the network, hurricane, and traffic state data from data Store.

### *Machine Learning-based Prediction Hurricane Traffic Predictor*

**Background:** The MLP Hurricane Traffic Predictor is a module within the MLP hurricane package that predicts likely future network traffic states for a specified prediction horizon after the hurricane warning is issued (such as 1 hour, 2 hours, 1 day, 2 days, or 7 days) under specific network environment and hurricane conditions. The module contains two parts: a One-Shot Predictor and an Online Predictor. The One-Shot Predictor uses the traffic state data during historical hurricanes and the outputs from the Traffic Impact Identifier as model inputs to predict the future network states during the periods after the hurricane warning issued. The Online Predictor uses information on current network link traffic states and outputs of the One-Shot Predictor to predict future network states. The Predictor uses the time series model to account for the latest trends and observations from the field. Therefore, it is able to predict traffic state evolution under forecasted hurricane conditions and adjust the prediction based on real-time field observations and updated hurricane data.

**Inputs:** The MLP Hurricane Traffic Predictor inputs contain three groups of data. The first group is Network Environment (e.g., link id, time, date). The second is the output data from the MLP Hurricane Traffic Impact Identifier or hurricane impact identification data (e.g., the traffic state on which links are to be impacted by the upcoming hurricane and on what days the impact will occur). The third group is real-time and archived data of traffic data as clustered into

different traffic states. All these data inputs are current information on the current network link provided by the data Store.

**Outputs:** The MLP Traffic Predictor provides future network traffic speed for the next specified prediction horizon (15 mins, 30 mins, 1 hour or 2 hours). For the One-Shot Predictor, the prediction horizon ranges from 1 day to 7 days at 1-hour interval. As for the Online Predictor, the prediction horizon is 6 hours at 1-hour interval.

**Process:** A time series model is applied to predict traffic speeds for links that are not marked as impacted links by the MLP Hurricane Traffic Impact Identifier. When the link is marked as impacted links for the upcoming hurricane, the One-Shot Predictor is used to predict the traffic state for the next 7 days at 1-hour interval. Then the speed estimates for the impacted days are calculated using the means of speed distributions for different traffic states (extracted from the archived data), the time series prediction, and transition matrix probabilities. The Oneshot Predictor is updated every day with the latest hurricane forecast data. In the meantime, the Online Predictor is used to predict the traffic state for the next 6 hours at 1-hour interval. The prediction generated by the Online Predictor is computed by the last 12 hours of real-time traffic state data and the output of the One-Shot Predictor.

**Dependencies:** The MLP Predictor depends on the current network environment, hurricane, and traffic state data from the data Store and the outputs from the MLP Hurricane Traffic Impact Identifier. Particularly, both archived data and real-time feeds are used.

## MLPPredict

**Background:** MLPPredict produces real-time short-term traffic speed forecasts using the calibrated Markov transition matrices along with the most current traffic speed, work zone, incident, and weather data and a long time series of previous traffic speeds.

**Inputs:** MLPPredict modules require current traffic speed, work zone, incident, and weather data, a long time series of previous traffic speeds, and roadway segment metadata.

**Outputs:** MLPPredict modules create short-term traffic speed predictions.

**Process:** A rolling window of the most recent 26 hours of traffic speed, work zone, incident, and weather data is maintained for each roadway segment used for input into the MLP model. Roadway segment metadata and the 26 hours of data are passed into the calibrated Markov model. Predicted traffic speeds are applied to the corresponding roadway segments and any upstream segments within a configured distance that failed to have a prediction generated.

**Dependencies:** MLPPredict modules are dependent on the data Store, a calibrated Markov model, Machine Learning-based Prediction Scenario Identifier, Machine Learning-based Prediction Traffic Predictor, and MLPUpdate modules that generate the long time series of traffic speeds.

**Known Instances:** MLPPredictLA, MLPPredictOH

## **MLPUpdate**

**Background:** To assist in the creation of real-time traffic speed predictions in the MLPPredict modules, MLPUpdate creates the long time series of traffic speed data to be used in case of a gap in the live data feeds.

**Inputs:** MLPUpdate modules require historic traffic speed, work zone, incident, and weather data and roadway segment metadata.

**Outputs:** MLPUpdate modules create long time series traffic speed data.

**Process:** Due to the large amount of processing and data required to generate a long time series of traffic speed data for each roadway segment, MLPUpdate runs its process once a week. Traffic speed, work zone, incident, and weather data from the previous 7 days is accumulated and passed into the calibrated Markov model along with roadway segment metadata to compute and store the long time series for each road segment.

**Dependencies:** MLPUpdate modules are dependent on the data Store and a calibrated Markov model.

**Known Instances:** MLPUpdateLA, MLPUpdateOH

## **MLPHurricane**

**Background:** MLPHurricane produces real-time traffic speed forecasts that factor in current and forecasted hurricane conditions before landfall to help decision making regarding evacuation.

**Inputs:** MLPHurricane modules require current and historic traffic speed data and hurricane forecasts.

**Outputs:** MLPHurricane modules create traffic speed predictions.

**Process:** When a new hurricane forecast is available, the forecast path of the hurricane is used to determine if any roadway segments configured for the MLPHurricane model will be affected by the hurricane. If the hurricane's path intersects the LA roadway network, the model is executed. First the SVM Oneshot Prediction component is given the relevant hurricane and traffic data to determine which segments will be impacted by the hurricane. Next the 7-Day Oneshot component is run to generate a long term prediction (7 days) of traffic speeds. Finally the Online Prediction component is used to predict traffic speeds for the next 6 hours.

**Dependencies:** MLPHurricane modules are dependent on the data Store, Machine Learning-based Prediction Hurricane Traffic Impact Identifier, Machine Learning-based Prediction Hurricane Traffic Predictor, and hurricane forecasts from the National Hurricane Center.

**Known Instances:** MLPHurricane

## **MLPProcess**

**Background:** The Scenarios module uses the MLP Oneshot method to make long-term speed predictions for roadway segments included in created scenarios.

**Inputs:** MLPProcess requires current and historic traffic speed, work zone, incident, and weather data, VSL changes, roadway segment metadata, and number of lanes available.

**Outputs:** MLPProcess creates long-term traffic speed predictions.

**Process:** After a Scenario is created and queued to process, the Scenarios module instantiates an MLPProcess object, passing it the configured changes to VSL and number of lanes available. Similar to the MLPUpdate process, traffic speed, work zone, incident, and weather data is accumulated to create a long time series of traffic speed data for each of the roadway segments specified in the Scenario. Then those long time series, roadway segment metadata, and scenario parameters are passed into the MLP Oneshot method to generate speed predictions up to 7 days in the future for the Scenario.

**Dependencies:** MLPProcess instances are dependent on the data Store, a calibrated MLP model, roadway segment definitions, and the Scenarios module.

**Known Instances:** n/a – MLPProcess is not a BaseBlock and instantiated on demand by Scenarios

## **METRo**

**Background:** The METRo model was developed by The Canadian Meteorological Center of Environment Canada as a standard pavement thermal modeling tool that is part of its road weather forecasting suite. It is widely used and adapted in many winter maintenance decision support systems, including NCAR’s Pikalert suite. IMRCP uses METRo as its Road Temperature and Snow Module. Performance and model improvements have been made to METRo inside of IMRCP to allow for wide spread predictions that keep track of the water and snow/ice reservoirs from one prediction cycle to the next.

**Inputs:** The METRo module takes many atmospheric weather parameters (air temperature, dew point, wind speed, precipitation rate, precipitation type, pressure, and cloud cover) both observed and forecasted, to run. Pavement state, pavement temperature, subsurface temperature, water reservoir, and snow/ice reservoir results from previous METRo predictions are used to initialize the state for subsequent predictions, if there is not another source providing those observations. Metadata for road segments including geo coordinates and if the segment is a bridge is needed as input to the model as well.

**Outputs:** The METRo module saves pavement state, pavement temperature, subsurface temperature, water depth, and snow/ice depth predictions to a file and sends a notification to the Directory that new data is available.

**Process:** The METRo module runs on a schedule, executing every 10 minutes in the current deployment. Atmospheric and road weather observations and forecasts, including the predictions

and water and snow/ice reservoir levels from the previous METRo run, are gathered from the data Store using the `getData` and `getReading` interfaces for each segment in each network. To avoid extra processing, segments are categorized by location and whether or not they are a bridge. The model is then run for each categorized location and the results are applied to each segment in that category.

**Dependencies:** The METRo module is dependent on the data Store for providing all of the needed atmospheric and road weather observations and forecasts. It is also dependent on the WayNetworks module for metadata describing the segments that are used to determine the locations to run the METRo model.

**Known Instances:** Metro

### MetroProcess

**Background:** The Scenarios module uses the METRo model to make 24-hour pavement state and temperature predictions for the roadway segments included in created scenarios.

**Inputs:** MetroProcess require many forecasted atmospheric and road weather parameters (air temperature, dew point, wind speed, precipitation rate, precipitation type, pressure, cloud cover, road temperature, subsurface temperature, and pavement state) and whether roadway segments are to be treated or plowed.

**Outputs:** MetroProcess creates long-term pavement state and temperature predictions.

**Process:** After a Scenario is created and queued to process, the Scenarios module instantiates a MetroProcess object, passing it the configured actions of the Scenario, namely which roadway segments are to be plowed or treated and at what time. For each roadway segment in the Scenario, the METRo model is ran 24 times, once for each hour of the Scenario. The outputs of each hour's run are used as inputs for the next hour's run.

**Dependencies:** MetroProcess is dependent on the data Store, roadway segment definitions, and the Scenarios module.

**Known Instances:** n/a – MetroProcess is not a BaseBlock and instantiated on demand by Scenarios

### GEO SERVICES [GEOSRV]

The Geo Services (GeoSrv) package maintains the fundamental geographical description of the roadway system and its components.

### WayNetworks

**Background:** The WayNetworks module defines and describes the different deployment networks and all of the ways (roadway segments) contained in each network. A way is a set of two or more points called nodes. Metadata for each way, including number of lanes, speed limit, and elevation data, is also stored by the module.

**Inputs:** The WayNetworks module creates and uses the networks configuration file. Requests from the NetworkGeneration webpage are made to create, finalize, reprocess, and delete networks.

**Outputs:** The WayNetwork module saves any updates to a network in the network configuration file and serves network and way definitions to other modules.

**Process:** At system start up, the WayNetworks module reads the networks configuration file and loads the existing network and way definitions into memory to be available for other system modules. Queries for way objects can be made by id or geo location. System administrators can use the Network webpage to manage networks; operations include creating, finalizing, reprocessing, and deleting a network. Creating a network has this module save the network's bounding polygon, label, and filter options to the network configuration file. Finalizing a network runs algorithms (merge, split, separate) designed to convert the existing roadway network into a more robust transportation model. Reprocessing a network flags that it is no longer finalized and queues it to be reprocessed which allows the network to get any updates from the OSM database. Deleting a network removes the network from the networks configuration file.

**Dependencies:** The WayNetworks module is dependent on networks being created by a System Administrator using the Network webpage, which uses Open Street Map way and node definitions as a starting point for creating the ways and nodes used by IMRCP.

**Known Instances:** WayNetworks

## DEM

**Background:** Mapbox provides a raster tileset that contains global elevation data. The Mapbox Terrain-DEM contains raw height values stored in PNG tiles that can be decoded to raw heights in meters. To have access to the API, a Mapbox access token is required. IMRCP downloads and caches the elevation tiles on demand. Tiles are only cached for a configured time, currently one month, so infrequent elevation changes are handled as needed. The Digital Elevation Map (DEM) is used to lookup the elevation for a particular geo coordinate when the elevation is needed and not otherwise provided directly by a data source.

**Inputs:** DEM takes a geo-coordinate (lon/lat pair) as input.

**Outputs:** DEM returns the elevation at the given geo-coordinate in meters.

**Process:** When an elevation request is made, first the geo-coordinate is converted into map tile coordinates at a specific zoom level and generates the file name of the desired tile. Next the module checks if the file is cached or if it needs to be updated because it is out of date. If needed, the tile is downloaded from the Mapbox API and cached on the server. Finally the corresponding pixel in the PNG of the geo-coordinate is determined and its value is decoded and returned as an elevation in meters.

**Dependencies:** The DEM module is dependent on the Mapbox API.

## **Known Instances:** DEM

### **Mercator**

**Background:** Most open source and commercial Maps API providers use the Spherical Mercator projection coordinate system since it is easy to work with and preserves shapes and angles. Therefore IMRCP contains the Mercator module which provides convenience methods for converting lon/lat, Mercator, and map tile coordinates from one coordinate system to another.

**Inputs:** The Mercator class accepts lon/lat, Mercator, or map tile coordinates.

**Outputs:** The Mercator class returns lon/lat, Mercator, or map tile coordinates.

**Process:** The Mercator class converts lon/lat, Mercator, and map tile coordinates from one coordinate system to another.

**Dependencies:** The Mercator class does not have any dependencies.

**Known Instances:** n/a – Mercator is not a BaseBlock and is instantiated on demand

### **OsmBz2ToBin**

**Background:** The definitions of roadway segments in IMRCP are derived from the Open Street Map database. OSM files for entire states can be downloaded in an XML format compressed by the bz2 algorithm. These files end up being very large and take a considerable time to open and process. Therefore IMRCP converts these files into a binary representation that is much more compact and faster to process.

**Inputs:** OsmBz2ToBin requires an OSM .xml.bz2 file.

**Outputs:** OsmBz2ToBin saves a binary representation of an OSM .xml.bz2 file.

**Process:** OsmBz2ToBin decompresses and loads an OSM .xml.bz2 file into memory which includes the definitions of ways and nodes and their associated set of key-value tags. While the ways and nodes are being processed a string pool of all the tags is created as well as a hashed spatial index for the ways. The string pool is written to the file first. The nodes and ways are written next, referencing any tag by index, instead of writing the same string multiple times. Ways reference nodes by file position, instead of by id. A separate binary index file is written which stores the hashed spatial index which allows for quick lookup of ways given a geo-coordinate.

**Dependencies:** OsmBz2ToBin objects are dependent on the OSM database

**Known Instances:** n/a – this class is not a BaseBlock and is instantiated on demand

### **OsmBinParser**

**Background:** OsmBinParser is used to load the files created by the OsmBz2ToBin class.

**Inputs:** OsmBinParser uses OSM .bin and hashed spatial index files.

**Outputs:** OsmBinParser creates OsmWay and OsmNode objects as outputs.

**Process:** OsmBinParser can load an entire OSM .bin file or a specific part of it based off of a given geo-coordinate using the hashed spatial index file, creating the OsmWay and OsmNode objects defined by the files.

**Dependencies:** OsmBinParser objects are dependent on OSM .bin and .bin.ndx files

**Known Instances:** n/a – this class is not a BaseBlock and is instantiated on demand

## WEB

### SecureBaseBlock

**Background:** To ensure that only known users have access to the system and to prevent potential distributed denial-of-service (DDoS) attacks, SecureBaseBlock, a child class of BaseBlock, was developed with security algorithms to be used for every request made from the webpages. User privileges and access are assigned by a system administrator.

**Inputs:** SecureBaseBlocks accept GET or POST request URLs.

**Outputs:** SecureBaseBlock module outputs are specific to the implemented child classes.

**Process:** GET or POST requests are passed to a SecureBaseBlock from the web server's container manager based on the URL pattern. First the SecureBaseBlock ensures that the request comes from a known, logged-in IMRCP user and that they have the correct permission for the component. Next the SecureBaseBlock determines if the user has the sufficient privileges for the operation. If both are true, then the request is processed, if not the request is ignored.

**Dependencies:** SecureBaseBlocks are dependent on the web server's container manager to pass them requests.

Known Instances: AreaLayerServlet, capvt, dphlnkvt, dphsnvt, essvt, gstdvnt, KRTPVTTileCache, KTSSRFTileCache, NetworkGeneration, PccatTileCache, PointsLayerServlet, RadarTileCache, rnrvt, RoadConditionAlerts, RoadLayerServlet, Scenarios, spdlnkvt, spdwndvt, SscstTileCache, stpvtvt, Subscriptions, TairTileCache, tpvtvt, TrafficAlerts, trflnkvt, TropicalStormPoints, UserSettingsServlet, visvt, WeatherAlerts

## NetworkGeneration

**Background:** A robust and well defined roadway network is essential to a successful IMRCP deployment. To assist in the arduous task of creating a roadway network, a set of tools and algorithms were developed to help automate the task as much as possible.

**Inputs:** NetworkGeneration accepts a polygon that defines the border of the desired network and a set of options to start the creation of a roadway network.

**Outputs:** A finalized roadway network is the product of NetworkGeneration after a created network has any necessary actions and edits performed on it.

**Process:** Using the Network webpage an IMRCP administrator can create a new roadway network and perform actions on the segments in the network. Actions include add, remove, merge, and split segment. After all desired actions are performed the network can be submitted to be finalized which runs multiple algorithms (merge, split, separate) on the segments to convert the existing roadway network into a more robust transportation model.

**Dependencies:** NetworkGeneration is dependent on the Open Street Map database to extract roadway segment definitions and metadata.

**Known Instances:** NetworkGeneration

### SessMgr

**Background:** Only authorized users with a valid session are allowed to access the system's user interfaces. SessMgr manages registered users, passwords, permissions, and browser sessions.

**Inputs:** The users file, username, password, and security token are inputs to SessMgr.

**Outputs:** Session objects with security tokens are generated by SessMgr.

**Process:** The users file is read at system startup to get the list of registered users' usernames, encrypted passwords, and permissions. Registered users log into the system by providing their username and password. If valid credentials are provided, a Session is created in memory and a security token is passed back to the browser and used for the remainder of the user's Session to authenticate access to other pages and interfaces. Sessions are removed from memory when a user logs out or is inactive for a specified amount of time.

**Dependencies:** The SessMgr module is dependent on a System Administrator using the main method outside of the system to create and add records to the users file.

**Known Instances:** SessMgr

### Scenarios

**Background:** One of IMRCP's purposes is to assist in planning how to use resources during storms, scheduled maintenance, and rush hour traffic. To achieve this purpose, the Scenarios tool was developed, which allows users to create Scenarios and generate a 24 hour prediction for road conditions and traffic based on associating actions (plowing, chemical treatment, Variable Speed Limit change, and number of available lanes change) with segment groups.

**Inputs:** The Scenarios component receives segment groups with associated time series of actions and a reference time.

**Outputs:** The Scenarios component generates 24 hour predictions of road conditions and traffic speeds.

**Process:** The Scenarios component receives a request to save a scenario template from the Create Scenarios webpage which includes the definitions of the segment groups and associated time series of actions for each group of the Scenario. Scenario templates can then be edited via the Create Scenarios webpage or given a reference time to process. Requests are processed in serial and generate a 24 hour prediction of road conditions and traffic speeds that can be viewed via the View Scenarios webpage.

**Dependencies:** The Scenarios component is dependent on the data Store and the METRo and MLP models to provide predictions.

**Known Instances:** Scenarios

### Subscriptions

**Background:** Subscriptions enables IMRCP to provide data reports and subscriptions to external systems and end users. One time reports can be helpful in analyzing past events while recurring subscriptions can be used to receive the most recent forecasts at a regular interval.

**Inputs:** The Subscriptions component receives information necessary to create a report or subscription. This includes the requested observation types, geographic context (a set of roadway segments), reference time, offset time, and duration.

**Outputs:** The Subscriptions component generates data reports to fulfill the requested reports or subscriptions. Available data reports are displayed and can be downloaded via the View Reports webpage.

**Process:** The Subscriptions component runs on a regular schedule. Each period of execution, it checks to see if there are any reports or subscriptions that need to be fulfilled and processes them. For each report or subscription being processed, a request for the desired data is made to the data Store and results are formatted into a data report.

**Dependencies:** The Subscriptions component is dependent on the data Store and the Create Report webpage.

**Known Instances:** Subscriptions

### LayerServlet

**Background:** Detailed data requests from the Map webpage are processed by different LayerServlets depending on the geographic extents of the request, which can be a single point, polyline (roadway segment), or polygon.

**Inputs:** The LayerServlet components receive a reference time, query time, and geographic extent.

**Outputs:** The LayerServlet components return a list of Observation objects.

**Process:** Users can left-click on point, road, or area layers on the Map webpage to request detailed data at that location. LayerServlet components query the data Store for the desired time and location and return a list of all observations that match the query.

**Dependencies:** The LayerServlet components are dependent on the data Store and the web server's container manager to pass them requests.

**Known Instances:** AreaLayerServlet, PointsLayerServlet, RoadLayerServlet

## TileCache

**Background:** Creating data views for data sets that cover large areas for the Map webpage can take a lot of processing time. To help alleviate processing burden and slow response times, the data views are tiled and cached by the server.

**Inputs:** The TileCache components receive a reference time, query time, and tile coordinates (x,y, zoom) as well as a configured observation type.

**Outputs:** The TileCache components return vector tiles that represent data requested for the given tile coordinate.

**Process:** The Map webpage sends reference time, query time, and tile coordinate requests to TileCache components. To process a request, the cache is checked to see if any matching tiles have been generated. If there are valid tiles in the cache, they are returned. Otherwise the data Store is queried for the desired time and matching data is processed into a vector tile view using configured RangeRules to bucket similar values together. The vector tiles are cached for future use and returned to fulfill the current request.

**Dependencies:** The TileCache components are dependent on configured RangeRules and the data Store.

**Known Instances:** capvt, gstdvnt, KRTPVTTileCache, KTSSRFTileCache, PccatTileCache, RadarTileCache, spdwndvt, SscstTileCache, TairTileCache, TropicalStormPoints, visvt

## TileServlet

**Background:** Creating data views for smaller data sets for the Map webpage can be done on the fly for each request. TileServlet components create data views for observations associated with a single point or roadway segment.

**Inputs:** The TileServlet components receive a reference time, query time, and tile coordinates (x,y, zoom) as well as a configured observation type.

**Outputs:** The TileServlet components return vector tiles that represent data requested for the given tile coordinate.

**Process:** The Map webpage sends reference time, query time, and tile coordinate requests to TileServlet components. To process a request, the data Store is queried for the desired time and

matching data is processed into a vector tile view using configured RangeRules to bucket similar values together. The vector tiles are returned to fulfill the current request.

**Dependencies:** The TileServlet components are dependent on configured RangeRules and the data Store.

**Known Instances:** dphlnkvt, dphsnvt, essvt, rmmvt, RoadConditionAlerts, spdlnkvt, stpvtvt, tpvtvt, TrafficAlerts, trflnkvt, WeatherAlerts

## WEB PAGES

The WebPages package provides the primary graphical user interfaces to the IMRCP system outputs. Only registered users have access and permission to use them.

### LogOn

**Background:** The LogOn page is the first page users view when accessing IMRCP and allow them to submit their username and password to log into the system. It can also be used to display system messages, like scheduled maintenance times, to users.

**Inputs:** The LogOn webpage receives a username and password.

**Outputs:** The LogOn webpage returns a security token used to authenticate other API calls for the current session.

**Process:** Users enter their username and password to submit them for authentication. If valid credentials are provided, a security token is returned and stored by the browser to authenticate other pages and API calls for the user's current session.

**Dependencies:** The LogOn webpage is dependent on the SessMgr component to authenticate credentials and generate a security token.

### Map

**Background:** The Map webpage provides a selectable, layered presentation of archived, current, and forecast traffic and weather conditions across the roadway networks. Map layers are described in Appendix B. Alert layers are further described in Appendix D.

The map provides several sets of controls for setting the view context. Typical map interactions (e. g., pan, zoom) set the spatial context for the view. Time controls set the temporal context for the view which consist of a reference time and query time. Using the calendar control, the reference time can be set, meaning that only data that was available at or before that time will be displayed. Date and time sliders are available at the bottom of the map to select the query time. Going back in time from the reference time enables viewing observations and model results as they were current at that time. Going forward in time from the reference time enables forecast data to be viewed as it was available at that reference time. Layer controls set the map overlays, icons, and display options. Selection categories are based on map graphical constructs. Layers available for each category correspond to particular data elements available from the data Store.

“Point” constructs are used to locate and identify observations from sensors (mobile and stationary) and alerts to localized events (e.g., incidents on roadways). “Road” constructs are used to locate and describe attributes of roadway segments including traffic and pavement conditions. “Area” constructs are used to represent areal attributes such as atmospheric conditions and weather alert areas. Users can left-click on any construct to receive detailed data at that location for the current reference and query time in a table view.

Map “Settings” enable to the user to select whether the map refreshes itself to maintain the reference time at “now”. Users can also save their settings causing the map to remember which layers are selected, the current map position, and zoom level for subsequent uses of the map.

**Inputs:** The Map webpage takes input from users using mouse and keyboard to interact with the different controls to set the spatial and temporal contexts and view data.

**Outputs:** The Map webpage displays archived, current, and forecast traffic and weather data on a map and table.

**Process:** After a user logs on to the system, the Map webpage is the default page. Users interact with the controls to view data for different spatial and temporal extents.

**Dependencies:** The Map webpage is dependent on LayerServlet, TileServlet, and TileCache modules to provide the different views of system data.

### Create Scenario

**Background:** The Create Scenario webpage provides a map interface to create and edit scenario templates which can then be given a name and reference time and submitted to process, generating a 24 hour prediction for road and traffic conditions.

**Inputs:** The Create Scenario webpage takes input from users using mouse and keyboard to interact with the controls.

**Outputs:** The Create Scenario webpage lists current scenario templates and their details.

**Process:** By following the instructions found in the dialog box, users create scenario templates by providing a name and creating roadway segment groups. Once a group is created, users associate a time series of actions to be applied to each group. Scenario templates are saved so that can be edited later or processed for different reference times. When a scenario template is complete, users enter a name and select a reference time to submit the template as a scenario to be processed. The 24 hour predictions that are generated by a processed scenario can be viewed via the View Scenarios webpage.

**Dependencies:** The Create Scenario webpage is dependent on the WayNetworks module to display the available networks and roadway segments and the Scenarios module to process all of the requests made by users.

## **View Scenarios**

**Background:** The View Scenarios webpage provides a map and table interface to view the 24 hour predictions for road and traffic conditions generated by a processed scenario.

**Inputs:** The View Scenarios webpage takes input from users using mouse and keyboard to interact with the controls.

**Outputs:** The View Scenarios webpage lists scenarios that have been processed and displays the predicted road and traffic conditions on the map and in a table.

**Process:** Users select which Scenario they would like to view. They can select which observation type to be displayed on the roadway segments on the map. The time to view can be selected by the time slider at the bottom of the map. By clicking on a roadway segment, a table will be displayed containing data and metadata for that segment.

**Dependencies:** The View Scenarios webpage is dependent on the WayNetworks module for roadway segment geometry definitions and the Scenarios module to serve the predictions created for each Scenario.

## **Create Report**

**Background:** The Create Report webpage provides a map interface to create data reports and subscriptions.

**Inputs:** The Create Report webpage takes input from users using mouse and keyboard to interact with the controls.

**Outputs:** The Create Report webpage provides the URL to download data reports.

**Process:** By following the instructions found in the dialog box, users create a report or subscription by first selecting the roadway segments they would like included in their report or subscription. Next users provide a name and select up to five observation types. In the same dialog, users choose whether they are creating a report or a subscription and select the reference time, offset, and duration. Once all the parameters are set, the report or subscription can be submitted to be processed.

**Dependencies:** The Create Report webpage is dependent on the WayNetworks module to display the available networks and roadway segments and the Subscription module to process all of the requests made by users.

## **View Reports**

**Background:** The View Report webpage provides a list interface to view and download available data files for reports and subscriptions.

**Inputs:** The View Report webpages takes input from users using mouse and keyboard to make selections on the interface.

**Outputs:** The View Reports webpage provides data files to download.

**Process:** A list of pending and available reports and subscriptions is displayed when the page is accessed. Left-clicking on a report downloads the data file for that report. Selecting a subscription will list all of the available data files associated with that subscription. Each data file can be downloaded by left-clicking on its name.

**Dependencies:** The View Reports webpage is dependent on the Subscriptions module to serve the data files created to fulfill each report and subscription.

## Network

**Background:** The Network webpage provides a map interface to create and edit roadway networks. Only System Administrators have permission to access this page.

**Inputs:** The Network webpage takes input from users using mouse and keyboard to interact with the controls.

**Outputs:** The Network webpage displays all of the networks and metadata about them.

**Process:** When the page is first accessed, the bounds of all existing networks are displayed on the map. Selecting the bounds of a network loads the detailed geometry of the network and allows the user to enter different modes to edit the network. Roadway segments can be added, removed, merged, or split. Once all desired edits are complete, the network can be submitted to be finalized. Users can export the network as an OSM .xml file at any time.

**Dependencies:** The Network webpage is dependent on the Open Street Maps database and the WayNetworks module to process all of the requests made by users.

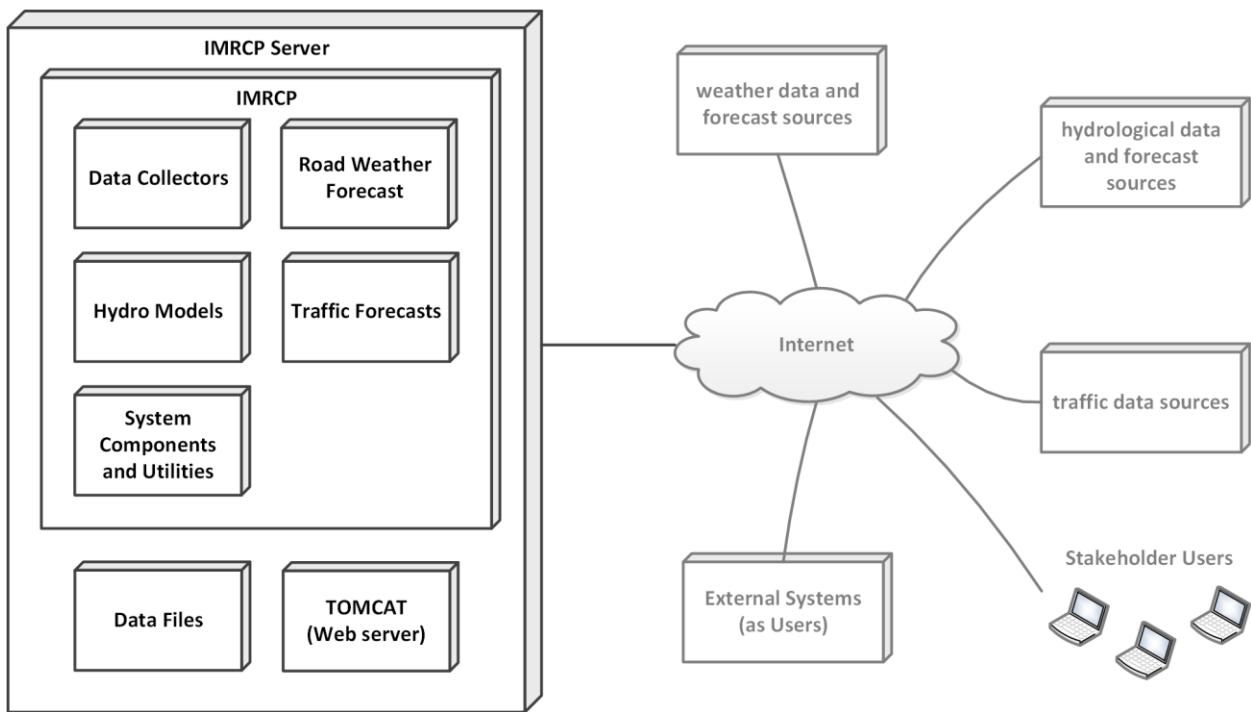
## SYSTEM COMPUTING INFRASTRUCTURE

This deployment of IMRCP is a system demonstration prototype. As such, the focus is on the development, and it is desirable to simplify the deployment in order to minimize system management overhead. It provides distributed user access for the development team, the review team, and the partner prototype agency. The following factors contribute to determining this configuration:

- IMRCP data services and computational services are closely linked to and benefit from co-location to reduce latencies and remote network calls.
- Other data sources, (for example, atmospheric weather and hydrology) are provided by external web services that do not drive any particular deployment solution.
- Potential future phase operational deployments would be linked to transportation management centers (TMCs) and integrated management solutions. The bulk of the real-time operational and traffic data comes from transportation management systems, and the majority of the end users are either agency personnel or travelers for whom data is already sourced from TMCs and their associated systems. It makes sense within that

context to anticipate and demonstrate a deployment as a forecast “appliance” rather than a distributed system with the potential limitations of traffic predictions as external services.

The demonstration system deployment is shown in Figure 6. The IMRCP system software and the Apache Tomcat Web server will be deployed on a common server. The server will use a high-bandwidth connection to the Internet to access data contributors and to provide access to IMRCP forecast products for stakeholders and systems. This configuration will be subject to review and re-evaluation during the development process to assure the project and system needs are being met.



Source: FHWA, 2018.

**Figure 6. Integrated Model for Road Condition Prediction Deployment.**

## CHAPTER 4. REFERENCES

- ISO/IEC/IEEE. *Systems and Software Engineering – Life Cycle Processes – Requirements Engineering*. ISO/IEC/IEEE 29148:2011.
- ISO/IEC/IEEE. *Systems and software engineering – Architecture description*. ISO/IEC/IEEE 42010:2011.
- Leidos. 2015. *Integrated Modeling for Road Condition Prediction Model Analysis*. 2015.  
Unpublished working paper developed under FHWA Contract DTFH61-12-D-00050,  
Task Order 5022, Integrated Modeling for Road Condition Prediction.
- Leidos. 2015. *Integrated Modeling for Road Condition Prediction Concept of Operations*.  
Unpublished working paper developed under FHWA Contract DTFH61-12-D-00050,  
Task Order 5022, Integrated Modeling for Road Condition Prediction.
- Leidos. 2016. *Integrated Modeling for Road Condition Prediction System Requirements*.  
Unpublished working paper developed under FHWA Contract DTFH61-12-D-00050,  
Task Order 5022, Integrated Modeling for Road Condition Prediction.
- Lukasik, et al. 2011. *Assessment of Emerging Opportunities for Real-time, Multimodal Decision Support Systems in Transportation Operations*. Report Number FHWA-JPO-10-058.  
Washington, DC: FHWA.



## APPENDIX A. INPUT DATA AVAILABILITY

All collectors poll for new files or updated data feeds on a regular schedule. Some sources, primarily National Weather Service forecasts, store multiple days' worth of files. Those collectors try to download any file that IMRCP has not collected by comparing the list of file available and files already stored in IMRCP's persistent storage. The polling frequency does not always match the expected availability because files are not always available at the exact same offset between collection cycles due to processing and uploading speeds.

**Table 4. Input Data Available Frequency.**

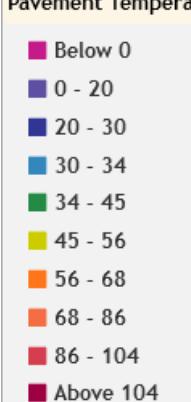
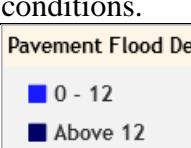
<b>Source</b>	<b>Polling Frequency</b>	<b>Expected Availability</b>	<b>Time range</b>
<b>ADCIRC</b>	30 minutes	New files 4 times a days	up to 180 hour forecasts (IMRCP uses hours 6-120)
<b>AHPS</b>	5 minutes	Updated 4 non-regular times an hour	1 hour for observations 24 hours for forecasts
<b>CAP</b>	5 minutes	Updated possibly every minute	Alerts and watches up to 3 days
<b>GFS</b>	6 hours, retries after 20 minutes if an error occurs	New files 4 times a day	up to 384 hour forecasts (IMRCP hours every 3 <sup>rd</sup> hour in-between hours 54 and 168)
<b>Inrix</b>	5 minutes	Updated every 5 minutes	5 minute observations
<b>LAc2c</b>	1 minute	Updated every 1 minute	1 minute observations
<b>LADOTD5 11</b>	5 minutes	Updated every 5 minutes	varies based on scheduled work zones
<b>MRMS</b>	1 minute	New files every 2 minutes	2 minute observations
<b>NDFD</b>	10 minutes	New files every 1 hour	up to 66 hour forecasts starting 1 hour after collection
<b>NHC</b>	30 minutes	New files approximately every 6 hours when there is an active tropical storm	120 hour forecasts
<b>OhGo</b>	5 minutes	Updated every 5 minutes	5 minute observations and varies based on scheduled work zones
<b>RAP</b>	10 minutes	New files every 1 hour	21 hour forecasts starting at collection time
<b>RTMA</b>	10 minutes	New file every 1 hour	1 hour forecast starting at collection time
<b>WxDE</b>	10 minutes	Updated every 10 minutes	up to 1 hour observations

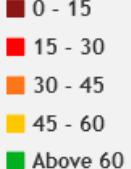
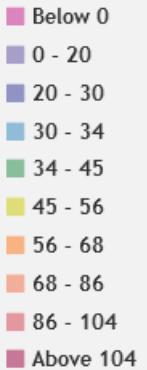
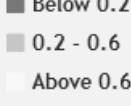
Source: FHWA



## APPENDIX B. LAYER DEFINITIONS

**Table 5. Layer Definitions.**

Layer	Observation Type	Legend	
Pavement State	STPVT	<b>Pavement State</b> ✕ 	The Pavement State layer categories are consistent with the METRo model pavement state categories. A flooded state is projected from local inundation calculations, where available.
Pavement Temperature	TPVT	<b>Pavement Temperature (F)</b> ✕ 	The Pavement Temperature map layer is divided into levels based on pavement behaviors at temperature intervals. Pavement temperatures between 30°F and 34°F indicate a transition to freezing conditions. Salt treatment may be effective on pavement at temperatures between 20°F and 29°F. Salt loses its effectiveness as an anti-icing agent below 20°F.
Pavement Snow Depth	DPHSN	<b>Pavement Snow Depth (in)</b> ✕ 	The Pavement Snow Depth layer is represented in bands for noticeable, actionable, and significant impacts on travel conditions.
Pavement Flood Depth	DPHLNK	<b>Pavement Flood Depth (in)</b> ✕ 	The flood depth calculation is available only at specific locations, and an absence of a flood depth indication is not necessarily evidence of an absence of flooding.

Layer	Observation Type	Legend
Traffic	TRFLNK	<b>Traffic</b> ✕  The Traffic layer is divided into five categories ranging from standstill traffic to traffic moving at or near the local speed limit.
Traffic Speed	SPDLNK	<b>Traffic Speed (mph)</b> ✕  The Traffic Speed layer is divided into five equal bands ranging from 0 to 75 mph.
Air Temperature	TAIR	<b>Air Temp (F)</b> ✕  The Air Temperature layer is divided into layers based on typical temperature behaviors. The narrow band at 32 F highlights the precipitation freezing point.
Surface Visibility	VIS	<b>Surface Visibility (mi)</b> ✕  The Surface Visibility layer remains white until the visibility is below 0.6 mi (1 km). Travelers can be significantly affected by visibility below this point.

<b>Layer</b>	<b>Observation Type</b>	<b>Legend</b>	
Wind Speed	SPDWND	<b>Wind Speed (mph)</b> ✖ <ul style="list-style-type: none"> <li>Below 5</li> <li>5 - 15</li> <li>15 - 25</li> <li>25 - 39</li> <li>39 - 57</li> <li>57 - 74</li> <li>74 - 85</li> <li>85 - 96</li> <li>96 - 111</li> <li>111 - 130</li> <li>130 - 144</li> <li>144 - 157</li> <li>Above 157</li> </ul>	The Wind Speed bands mark increasing intensity up to and through the tropical storm and hurricane levels.
Wind Gust Speed	GSTWND	<b>Wind Gust Speed (mph)</b> ✖ <ul style="list-style-type: none"> <li>Below 5</li> <li>5 - 15</li> <li>15 - 25</li> <li>25 - 39</li> <li>39 - 57</li> <li>57 - 74</li> <li>74 - 85</li> <li>85 - 96</li> <li>96 - 111</li> <li>111 - 130</li> <li>130 - 144</li> <li>144 - 157</li> <li>Above 157</li> </ul>	The Wind Gust Speed bands use the same levels as Wind Speed.

Layer	Observation Type	Legend
Radar	RDR0	<p>Radar (dBZ) ✕</p> <ul style="list-style-type: none"> <li>5 - 10</li> <li>10 - 15</li> <li>15 - 20</li> <li>20 - 25</li> <li>25 - 30</li> <li>30 - 35</li> <li>35 - 40</li> <li>40 - 45</li> <li>45 - 50</li> <li>50 - 55</li> <li>55 - 60</li> <li>60 - 65</li> <li>65 - 70</li> <li>70 - 75</li> </ul>
Precipitation Rate & Type	PCCAT	<p>Precip Rate and Type ✕</p> <ul style="list-style-type: none"> <li>Rain - Light</li> <li>Rain - Moderate</li> <li>Rain - Heavy</li> <li>Frz Rain - Light</li> <li>Frz Rain - Moderate</li> <li>Frz Rain - Heavy</li> <li>Snow - Light</li> <li>Snow - Moderate</li> <li>Snow - Heavy</li> <li>Ice Pellets - Light</li> <li>Ice Pellets - Moderate</li> <li>Ice Pellets - Heavy</li> </ul>

Layer	Observation Type	Legend
Surge and Tide	DPHLIQ	<p><b>Surge and Tide (ft) ✕</b></p> <ul style="list-style-type: none"> <li>■ Below 0.3</li> <li>■ 0.3 - 0.6</li> <li>■ 0.6 - 0.9</li> <li>■ 0.9 - 1.2</li> <li>■ 1.2 - 1.5</li> <li>■ 1.5 - 1.8</li> <li>■ 1.8 - 2.1</li> <li>■ 2.1 - 2.4</li> <li>■ 2.4 - 2.7</li> <li>■ 2.7 - 3.0</li> <li>■ 3.0 - 3.3</li> <li>■ 3.3 - 3.6</li> <li>■ 3.6 - 3.9</li> <li>■ Above 3.9</li> </ul>
Kriged Pavement Temp	KRTPVT	<p><b>Kriged Pavement Temp (F) ✕</b></p> <ul style="list-style-type: none"> <li>■ Below -14.0</li> <li>■ 14.0 - 15.8</li> <li>■ 15.8 - 17.6</li> <li>■ 17.6 - 19.4</li> <li>■ 19.4 - 21.2</li> <li>■ 21.2 - 23.0</li> <li>■ 23.0 - 24.8</li> <li>■ 24.8 - 26.6</li> <li>■ 26.6 - 28.4</li> <li>■ 28.4 - 30.2</li> <li>■ 30.2 - 32.0</li> <li>■ 32.0 - 35.6</li> <li>■ 35.6 - 39.2</li> <li>■ 39.2 - 42.8</li> <li>■ 42.8 - 46.4</li> <li>■ 46.4 - 50.0</li> <li>■ 50.0 - 53.6</li> <li>■ 53.6 - 57.2</li> <li>■ 57.2 - 60.8</li> <li>■ 60.8 - 64.4</li> <li>■ 64.4 - 68.0</li> <li>■ Above 68.0</li> </ul>

Layer	Observation Type		Legend
Kriged Subsurface Temp	KTSSRF	<b>Kriged Subsurface Temp (F)</b> ✕ <ul style="list-style-type: none"> <li><span style="background-color: #4f81bd; display: inline-block; width: 10px; height: 10px;"></span> Below -14.0</li> <li><span style="background-color: #5cb85c; display: inline-block; width: 10px; height: 10px;"></span> 14.0 - 15.8</li> <li><span style="background-color: #5bc0de; display: inline-block; width: 10px; height: 10px;"></span> 15.8 - 17.6</li> <li><span style="background-color: #4db6ac; display: inline-block; width: 10px; height: 10px;"></span> 17.6 - 19.4</li> <li><span style="background-color: #4db6ac; display: inline-block; width: 10px; height: 10px;"></span> 19.4 - 21.2</li> <li><span style="background-color: #4db6ac; display: inline-block; width: 10px; height: 10px;"></span> 21.2 - 23.0</li> <li><span style="background-color: #4db6ac; display: inline-block; width: 10px; height: 10px;"></span> 23.0 - 24.8</li> <li><span style="background-color: #4db6ac; display: inline-block; width: 10px; height: 10px;"></span> 24.8 - 26.6</li> <li><span style="background-color: #4db6ac; display: inline-block; width: 10px; height: 10px;"></span> 26.6 - 28.4</li> <li><span style="background-color: #4db6ac; display: inline-block; width: 10px; height: 10px;"></span> 28.4 - 30.2</li> <li><span style="background-color: #4db6ac; display: inline-block; width: 10px; height: 10px;"></span> 30.2 - 32.0</li> <li><span style="background-color: #4db6ac; display: inline-block; width: 10px; height: 10px;"></span> 32.0 - 35.6</li> <li><span style="background-color: #4db6ac; display: inline-block; width: 10px; height: 10px;"></span> 35.6 - 39.2</li> <li><span style="background-color: #4db6ac; display: inline-block; width: 10px; height: 10px;"></span> 39.2 - 42.8</li> <li><span style="background-color: #4db6ac; display: inline-block; width: 10px; height: 10px;"></span> 42.8 - 46.4</li> <li><span style="background-color: #ff8c00; display: inline-block; width: 10px; height: 10px;"></span> 46.4 - 50.0</li> <li><span style="background-color: #ff8c00; display: inline-block; width: 10px; height: 10px;"></span> 50.0 - 53.6</li> <li><span style="background-color: #ff8c00; display: inline-block; width: 10px; height: 10px;"></span> 53.6 - 57.2</li> <li><span style="background-color: #ff8c00; display: inline-block; width: 10px; height: 10px;"></span> 57.2 - 60.8</li> <li><span style="background-color: #ff8c00; display: inline-block; width: 10px; height: 10px;"></span> 60.8 - 64.4</li> <li><span style="background-color: #ff8c00; display: inline-block; width: 10px; height: 10px;"></span> 64.4 - 68.0</li> <li><span style="background-color: #ff8c00; display: inline-block; width: 10px; height: 10px;"></span> Above 68.0</li> </ul>	Spatial estimates of subsurface temperature are computed by IMRCP from measurements at environmental sensor stations (ESS) using Kriging statistical methods.
NWS Alerts	EVT	<b>NWS Alerts</b> ✕ <ul style="list-style-type: none"> <li><span style="background-color: #d9534f; display: inline-block; width: 10px; height: 10px;"></span> Fire</li> <li><span style="background-color: #ff8c00; display: inline-block; width: 10px; height: 10px;"></span> Heat</li> <li><span style="background-color: #ffd700; display: inline-block; width: 10px; height: 10px;"></span> Storm/Tornado</li> <li><span style="background-color: #cccccc; display: inline-block; width: 10px; height: 10px;"></span> Wind/Fog/Smoke</li> <li><span style="background-color: #a08030; display: inline-block; width: 10px; height: 10px;"></span> Air Quality</li> <li><span style="background-color: #e0c0a0; display: inline-block; width: 10px; height: 10px;"></span> Earthquake/Volcano</li> <li><span style="background-color: #80e6ee; display: inline-block; width: 10px; height: 10px;"></span> Winter Storm</li> <li><span style="background-color: #ff00ff; display: inline-block; width: 10px; height: 10px;"></span> Freeze</li> <li><span style="background-color: #9370db; display: inline-block; width: 10px; height: 10px;"></span> Cold</li> <li><span style="background-color: #4f81bd; display: inline-block; width: 10px; height: 10px;"></span> Flood</li> <li><span style="background-color: #5cb85c; display: inline-block; width: 10px; height: 10px;"></span> Lake/Marine/Coastal</li> <li><span style="background-color: #5bc0de; display: inline-block; width: 10px; height: 10px;"></span> Tropical Storm</li> <li><span style="background-color: #a0522d; display: inline-block; width: 10px; height: 10px;"></span> Special Weather</li> <li><span style="background-color: #9acd32; display: inline-block; width: 10px; height: 10px;"></span> Other</li> <li><span style="background-color: #555555; display: inline-block; width: 10px; height: 10px;"></span> Other</li> </ul>	The NWS Alerts Layer is categorized based on the type of alert issued by the NWS.

Layer	Observation Type	Legend	
Tropical Storm Cone	n/a	<b>Tropical Storm Cone</b>   Subtropical Depression  Subtropical Storm  Tropical Depression  Tropical Storm  Hurricane  Major Hurricane	<p>Tropical storm cones are represented from National Hurricane Center (NHC) data to show the potential path and current category of tropical storms.</p>

Source: FHWA, 2019



## APPENDIX C. OBSERVATION TYPE DEFINITIONS

**Table 6. Observation type descriptions.**

Name	Description
COVCLD	total cloud cover
DIRWND	wind direction
DPHLIQ	liquid inundation depth
DPHLNK	link depth
DPHSN	snow inundation depth
EVT	event
GSTWND	wind speed gust
KRTPVT	kriged pavement temperature
KTSSRF	kriged subsurface temperature
MPLOW	MAC main plow
PCCAT	precipitation category
PRSUR	surface pressure
RH	relative humidity
RTEPC	precipitation rate
RTLIQM	liquid material rate
RTPREM	prewet material rate
RTSLDM	solid material rate
SPDLNK	average speed of vehicles on each link
SPDWND	wind speed
SSCST	extra tropical storm surge combined surge and tide
STG	flood stage
STPVT	pavement state
TAIR	air temperature
TDEW	dew point
TPLIQM	liquid material type
TPLOW	MAC tow plow
TPPREM	prewet material type
TPSLDM	solid material type
TPVT	pavement temperature
TRFLNK	traffic
TRSCAT	tropical storm category
TRSCNE	tropical storm cone
TRSTRK	tropical storm track
TSSRF	subsurface temperature
TYPPC	precipitation type
VIS	surface visibility
WPLOW	MAC wing plow

**Table 7. Observation types enumeration.**

Name	Enumeration	Description
EVT	101	light-winter-precip
	102	moderate-winter-precip
	103	heavy-winter-precip
	104	light-precip
	105	moderate-precip
	106	heavy-precip
	107	low-visibility
	108	flood-stage-action
	109	flood-stage-flood
	201	dew-on-roadway
	202	frost-on-roadway
	203	blowing-snow
	204	icy-roadway
	301	incident
	302	workzone
	303	slow-traffic
	304	very-slow-traffic
	305	flooded-road
	306	lengthy-queue
	307	unusual-congestion
	399	test
	512	accident
	513	serious-accident
	514	injury-accident
	515	minor-accident
	516	multi-vehicle-accident
	517	numerous-accidents
	518	accident-involving-a-bicycle
	519	accident-involving-a-bus
	520	accident-involving-a-motorcycle
	521	accident-involving-a-pedestrian
	522	accident-involving-a-train
	523	accident-involving-a-truck
	524	accident-involving-a-semi-trailer
	525	accident-involving-a-hazardous-materials
	526	earlier-accident
	527	medical-emergency
	528	secondary-accident
	529	rescue-and-recovery-work-removed
	530	accident-investigation-work
	531	incident
	532	stalled-vehicle
	533	abandoned-vehicle
	534	disabled-vehicle

Name	Enumeration	Description
	535	disabled-truck
	536	disabled-semi-trailer
	537	disabled-bus
	538	disabled-train
	539	vehicle-spun-out
	540	vehicle-on-fire
	541	vehicle-in-water
	542	vehicles-slowing-to-look-at-accident
	543	jackknifed-semi-trailer
	544	jackknifed-trailer-home
	545	jackknifed-trailer
	546	spillage-occurring-from-moving-vehicle
	547	acid-spill
	548	chemical-spill
	549	fuel-spill
	550	hazardous-materials-spill
	551	oil-spill
	552	spilled-load
	553	toxic-spill
	554	overturned-vehicle
	555	overturned-truck
	556	overturned-semi-trailer
	557	overturned-bus
	558	derailed-train
	559	stuck-vehicle
	560	truck-stuck-under-bridge
	561	bus-stuck-under-bridge
	562	accident-cleared
	563	incident-cleared
	1000	Extreme Fire Danger
	1001	Fire Warning
	1002	Fire Weather Watch
	1003	Red Flag Warning
	1004	Heat Advisory
	1005	Excessive Heat Warning
	1006	Excessive Heat Watch
	1007	Severe Thunderstorm Warning
	1008	Severe Thunderstorm Watch
	1009	Storm Warning
	1010	Storm Watch
	1011	Tornado Warning
	1012	Tornado Watch
	1013	Severe Weather Statement
	1014	High Wind Warning
	1015	High Wind Watch

Name	Enumeration	Description
	1016	Wind Advisory
	1017	Extreme Wind Warning
	1018	Brisk Wind Advisory
	1019	Blowing Dust Advisory
	1020	Dust Storm Warning
	1021	Dense Fog Advisory
	1022	Dense Smoke Advisory
	1023	Air Quality Alert
	1024	Air Stagnation Advisory
	1025	Ashfall Advisory
	1026	Ashfall Warning
	1027	Earthquake Warning
	1028	Volcano Warning
	1029	Winter Storm Warning
	1030	Winter Storm Watch
	1031	Winter Weather Advisory
	1032	Ice Storm Warning
	1033	Blizzard Warning
	1034	Blizzard Watch
	1035	Avalanche Warning
	1036	Avalanche Watch
	1037	Blowing Snow Advisory
	1038	Snow and Blowing Snow Advisory
	1039	Heavy Snow Warning
	1040	Sleet Advisory
	1041	Sleet Warning
	1042	Snow Advisory
	1043	Freeze Warning
	1044	Freeze Watch
	1045	Freezing Drizzle Advisory
	1046	Freezing Fog Advisory
	1047	Freezing Rain Advisory
	1048	Freezing Spray Advisory
	1049	Frost Advisory
	1050	Hard Freeze Warning
	1051	Hard Freeze Watch
	1052	Wind Chill Advisory
	1053	Wind Chill Warning
	1054	Wind Chill Watch
	1055	Extreme Cold Warning
	1056	Extreme Cold Watch
	1057	Flash Flood Statement
	1058	Flash Flood Warning
	1059	Flash Flood Watch
	1060	Flood Advisory

Name	Enumeration	Description
	1061	Flood Statement
	1062	Flood Warning
	1063	Flood Watch
	1064	Hydrologic Advisory
	1065	Hydrologic Outlook
	1066	Beach Hazards Statement
	1067	Coastal Flood Advisory
	1068	Coastal Flood Statement
	1069	Coastal Flood Warning
	1070	Coastal Flood Watch
	1071	Gale Warning
	1072	Gale Watch
	1073	Hazardous Seas Warning
	1074	Hazardous Seas Watch
	1075	Heavy Freezing Spray Warning
	1076	Heavy Freezing Spray Watch
	1077	High Surf Advisory
	1078	High Surf Warning
	1079	Lake Effect Snow Advisory
	1080	Lake Effect Snow and Blowing Snow Advisory
	1081	Lake Effect Snow Warning
	1082	Lake Effect Snow Watch
	1083	Lakeshore Flood Advisory
	1084	Lakeshore Flood Statement
	1085	Lakeshore Flood Warning
	1086	Lakeshore Flood Watch
	1087	Lake Wind Advisory
	1088	Low Water Advisory
	1089	Marine Weather Statement
	1090	Rip Current Statement
	1091	Small Craft Advisory
	1092	Special Marine Warning
	1093	Tsunami Advisory
	1094	Tsunami Warning
	1095	Tsunami Watch
	1096	Hurricane Force Wind Warning
	1097	Hurricane Force Wind Watch
	1098	Hurricane Statement
	1099	Hurricane Warning
	1100	Hurricane Watch
	1101	Hurricane Wind Warning
	1102	Hurricane Wind Watch
	1103	Tropical Storm Warning
	1104	Tropical Storm Watch
	1105	Tropical Storm Wind Warning

Name	Enumeration	Description
	1106	Tropical Storm Wind Watch
	1107	Typhoon Statement
	1108	Typhoon Warning
	1109	Typhoon Watch
	1110	Hazardous Weather Outlook
	1111	Special Weather Statement
	1112	911 Telephone Outage
	1113	Administrative Message
	1114	Child Abduction Emergency
	1115	Civil Danger Warning
	1116	Civil Emergency Message
	1117	Evacuation Immediate
	1118	Hazardous Materials Warning
	1119	Law Enforcement Warning
	1120	Local Area Emergency
	1121	Nuclear Power Plant Warning
	1122	Radiological Hazard Warning
	1123	Shelter In Place Warning
	1124	Test
	5888	impassable
	5889	almost-impassable
	5890	passable-with-care
	5891	passable
	5892	surface-water-hazard
	5893	danger-of-hydroplaning
	5894	wet-pavement
	5895	treated-pavement
	5896	slippery
	5897	low-ground-clearance
	5898	at-grade-level-crossing
	5899	mud-on-roadway
	5900	leaves-on-roadway
	5901	loose-sand-on-roadway
	5902	loose-gravel
	5903	fuel-on-roadway
	5904	oil-on-roadway
	5905	road-surface-in-poor-condition
	5906	melting-tar
	5907	uneven-lanes
	5908	rough-road
	5909	rough-crossing
	5910	ice
	5911	icy-patches
	5912	black-ice
	5913	ice-pellets-on-roadway

Name	Enumeration	Description
	5914	ice-build-up
	5915	freezing-rain
	5916	wet-and-icy-roads
	5917	melting-snow
	5918	slush
	5919	frozen-slush
	5920	snow-on-roadway
	5921	packed-snow
	5922	packed-snow-patches
	5923	plowed-snow
	5924	wet-snow
	5925	fresh-snow
	5926	powder-snow
	5927	granular-snow
	5928	froazen-snow
	5929	crusted-snow
	5930	deep-snow
	5931	snow-drifts
	5932	drifting-snow
	5933	expected-snow-accumulation
	5934	current-snow-accumulation
	5935	sand
	5936	gravel
	5937	paved
	5938	dry-pavement
	5939	snow-cleared
	5940	pavement-conditions-improved
	5941	skid-hazard-reduced
	5942	pavement-conditions-cleared
MPLOW	0	Plow up
	1	Plow down

Name	Enumeration	Description
PCCAT	0 1 2 3 4 5 6 7 8 9 10 11 12 101 102 104 105 106	no-precipitation light-rain moderate-rain heavy-rain light-freezing-rain moderate-freezing-rain heavy-freezing-rain light-snow moderate-snow heavy-snow light-ice moderate-ice heavy-ice other unknown light-unidentified moderate-unidentified heavy-unidentified
STG	0 1 2 3 4 5	not-defined no-action action flood moderate major
STPVT	1 2 3 4 5 6 7 8 9 10 11 12 13 14 20 21 22 23 30	other error dry trace-moisture wet chemically-wet ice-warning ice-watch snow-warning snow-watch absorption dew frost absorption-at-dewpoint ice/snow slush melting-snow icing-rain flooded
TPLOW	0 1	Plow up Plow down

Name	Enumeration	Description
TRSCAT	479 642 809 1057 1072 37345 37360	Tropical Depression Hurricane Major Hurricane Tropical Depression Tropical Storm Subtropical Depression Subtropical Storm
TRSCNE	479 642 809 1057 1072 37345 37360	Tropical Depression Hurricane Major Hurricane Tropical Depression Tropical Storm Subtropical Depression Subtropical Storm
TRSTRK	479 642 809 1057 1072 37345 37360	Tropical Depression Hurricane Major Hurricane Tropical Depression Tropical Storm Subtropical Depression Subtropical Storm
TYPPC	0 1 2 3 4 5 6	none rain snow ice-pellets freezing-rain other unknown
WPLOW	0 1	Plow up Plow down

**Table 8. Observation type source descriptions.**

Source	Forecast/ Observation	Spatial Extent	Temporal Extent	Observation Types
ADCIRC	forecasts	2.5 km x 2.5 km grid for CONUS	1 hour forecasts for 120 hours starting 6 hours after collection	SSCST
AHPS	observations and forecasts	Individual stations	Most recent observed values and 24 hour forecast	EVT, STG, STPVT
CAP	observations and forecasts	County and custom polygons	Varies	EVT
Geotab	observations	Individual vehicles	Observations valid for 1 minute	MPLOW, RTLIQM, RTPREM, RTSLDM, SPDLNK, TPLIQM, TPLOW, TPPREM, TPSLDM, TPVT, WPLOW
GFS	forecasts	25 km x 25 km grid for entire world	3 hour forecasts for 168 hours starting 54 hours after collection	DPHSN, GSTWND, PCCAT, PRSUR, RH, RTEPC, SPDWND, TAIR, TDEW, TYPPC, VIS
IMRCP	observations	Area surrounding individual stations	Observations valid for 1 hour	KRTPVT, KTSSRF
IMRCP	forecasts	2.5 km x 2.5 km grid for CONUS	1 hour forecasts for 72 hours starting 1 hour after collection	PCCAT
IMRCP	observations	1 km x 1 km grid for CONUS	Observations valid for 4 minutes	PCCAT
Inrix	observations	Individual segments	Observations valid for 5 minutes	SPDLNK
Lac2c	observations	Individual geo coordinates	Observations valid for 1 minute	EVT, SPDLNK
Ladotd511	observations	Individual geo coordinate points and polylines	Varies	EVT
MEtro	forecasts	Individual segments	2 minute forecasts for 1 hour, then 20 minute forecasts for 11 hours	DPHLIQ, DPHSN, STPVT, TPVT, TSSRF
mlp	forecasts	Individual segments	15 minute forecasts for 2 hours, or 1	SPDLNK, TRFLNK

<b>Source</b>	<b>Forecast/ Observation</b>	<b>Spatial Extent</b>	<b>Temporal Extent</b>	<b>Observation Types</b>
			hour forecasts for 24 hours	
mrrms	observations	1 km x 1 km grid for CONUS	Observations valid for 4 minutes	RDR0, RTEPC
ndfd	forecasts	2.5 km x 2.5 km grid for CONUS	1to 3 hour forecasts for 72 hours starting 1 hour after collection	COVCLD, RTEPC, SPDWND, TAIR, TDEW
nhc	forecasts	Tropical storm cones of probability	6 hour forecasts for 120 hours	TRSCAT, TRSCNE, TRSTRK
ohgo	observations	Individual geo coordinates	Observations valid for 5 minutes	EVT
rap	forecasts	13 km x 13 km grid for CONUS	1 hour forecasts for 21 hours	PCCAT, PRSUR, RTEPC, SPDWND, TYPPC, VIS
rtma	forecasts	2.5 km x 2.5 km grid for CONUS	1 hour forecast	COVCLD, DIRWND, GSTWND, PRSUR, SPDWND, TAIR, TDEW, VIS
wxde	observations	Individual stations	Observations valid for 1 hour	DIRWND, DPHLNK, GSTWND, PCCAT, PRSUR, RH, RTEPC, SPDWND, STPVT, TAIR, TDEW, TPVT, TSSRF, TYPPC, VIS

**Table 9. Observation type synthesis algorithms.**

Name	Description	Source – Observations	Source – Predictions
dphliq	liquid inundation depth	Model of the Environment and Temperature of Roads (METRo) is run for each link in the road network model to determine liquid inundation depth estimations.	METRo is run for each link in the road network model to determine liquid inundation depth predictions.
dphlnk	link depth	AHPS stage observations at select locations in the road network model are collected when new values are available. These values are used to determine the flood depth on links based on inundation mapping provided by NOAA/NWS.	AHPS stage predictions at three locations in the road network model are collected when new values are available. These values are used to determine the flood depth on links based on inundation mapping provided by NOAA.
dphsn	snow inundation depth	METRo is run for each link in the road network model to determine pavement snow depth estimations. The snow inventory is tracked from previous runs.	METRo is run for each link in the road network model to determine pavement snow depth predictions. The snow inventory is tracked from each run to the next, accounting for new accumulation and melting.
evt	event	Workzone and Incident event details are collected from contributing transportation management centers. National Weather Service (NWS) Common Alerting Protocol (CAP) alert events are collected from NWS. CAP alerts affecting counties use previously stored county definitions to display on the map. CAP alerts affecting areas other than counties use the area definition provided in the CAP alert to display on the map.	Workzone and Incident event details are collected from contributing transportation management centers. NWS CAP alert events are collected from NWS. CAP alerts affecting counties use previously stored county definitions to display on the map. CAP alerts affecting areas other than counties use the area definition provided in the CAP alert to display on the map.

Name	Description	Source – Observations	Source – Predictions
pccat	precipitation category	<p>The precipitation category is determined based on observation TYPPC and RTEPC.</p> <ul style="list-style-type: none"> <li>• Light Freezing Rain: RTEPC <math>\leq</math> <math>7.056 \times 10^{-5}</math> kg/m<sup>2</sup>-s and TYPPC = [freezing rain]</li> <li>• Medium Freezing Rain: <math>7.056 \times 10^{-5} &lt; \text{RTEPC} \leq 7.056 \times 10^{-4}</math> kg/m<sup>2</sup>-s and TYPPC = [freezing rain]</li> <li>• Heavy Freezing Rain: <math>7.056 \times 10^{-4} &lt; \text{RTEPC}</math> kg/m<sup>2</sup>-s and TYPPC = [freezing rain]</li> <li>• Light Snow: RTEPC <math>\leq 7.056 \times 10^{-5}</math> kg/m<sup>2</sup>-s and TYPPC = [snow]</li> <li>• Medium Snow: <math>7.056 \times 10^{-5} &lt; \text{RTEPC} \leq 7.056 \times 10^{-4}</math> kg/m<sup>2</sup>-s and TYPPC = [snow]</li> <li>• Heavy Snow: <math>7.056 \times 10^{-4} &lt; \text{RTEPC}</math> kg/m<sup>2</sup>-s and TYPPC = [snow]</li> <li>• Light Ice Pellets: RTEPC <math>\leq 7.056 \times 10^{-5}</math> kg/m<sup>2</sup>-s and TYPPC = [ice pellets,]</li> <li>• Medium Ice Pellets: <math>7.056 \times 10^{-5} &lt; \text{RTEPC} \leq 7.056 \times 10^{-4}</math> kg/m<sup>2</sup>-s and TYPPC = [ice pellets,]</li> <li>• Heavy Ice Pellets: <math>7.056 \times 10^{-4} &lt; \text{RTEPC}</math> kg/m<sup>2</sup>-s and TYPPC = [ice pellets]</li> <li>• Light Rain: RTEPC <math>\leq 7.056 \times 10^{-4}</math> kg/m<sup>2</sup>-s and TYPPC = [rain]</li> </ul>	<p>The precipitation category is determined based on predicted TYPPC and RTEPC.</p> <ul style="list-style-type: none"> <li>• Light Freezing Rain: RTEPC <math>\leq 7.056 \times 10^{-5}</math> kg/m<sup>2</sup>-s and TYPPC = [freezing rain]</li> <li>• Medium Freezing Rain: <math>7.056 \times 10^{-5} &lt; \text{RTEPC} \leq 7.056 \times 10^{-4}</math> kg/m<sup>2</sup>-s and TYPPC = [freezing rain]</li> <li>• Heavy Freezing Rain: <math>7.056 \times 10^{-4} &lt; \text{RTEPC}</math> kg/m<sup>2</sup>-s and TYPPC = [freezing rain]</li> <li>• Light Snow: RTEPC <math>\leq 7.056 \times 10^{-5}</math> kg/m<sup>2</sup>-s and TYPPC = [snow]</li> <li>• Medium Snow: <math>7.056 \times 10^{-5} &lt; \text{RTEPC} \leq 7.056 \times 10^{-4}</math> kg/m<sup>2</sup>-s and TYPPC = [snow]</li> <li>• Heavy Snow: <math>7.056 \times 10^{-4} &lt; \text{RTEPC}</math> kg/m<sup>2</sup>-s and TYPPC = [snow]</li> <li>• Light Ice Pellets: RTEPC <math>\leq 7.056 \times 10^{-5}</math> kg/m<sup>2</sup>-s and TYPPC = [ice pellets,]</li> <li>• Medium Ice Pellets: <math>7.056 \times 10^{-5} &lt; \text{RTEPC} \leq 7.056 \times 10^{-4}</math> kg/m<sup>2</sup>-s and TYPPC = [ice pellets,]</li> <li>• Heavy Ice Pellets: <math>7.056 \times 10^{-4} &lt; \text{RTEPC}</math> kg/m<sup>2</sup>-s and TYPPC = [ice pellets]</li> <li>• Light Rain: RTEPC <math>\leq 7.056 \times 10^{-4}</math> kg/m<sup>2</sup>-s and TYPPC = [rain]</li> </ul>

Name	Description	Source – Observations	Source – Predictions
		<ul style="list-style-type: none"> <li>• Medium Rain: <math>7.056 \times 10^{-4} &lt; RTEPC \leq 2.117 \times 10^{-3}</math> kg/m<sup>2</sup>-s and TYPPC = [rain]</li> <li>• Heavy Rain: <math>2.117 \times 10^{-3} &lt; RTEPC</math> kg/m<sup>2</sup> and TYPPC = [rain]</li> </ul>	<ul style="list-style-type: none"> <li>• Medium Rain: <math>7.056 \times 10^{-4} &lt; RTEPC \leq 2.117 \times 10^{-3}</math> kg/m<sup>2</sup>-s and TYPPC = [rain]</li> <li>• Heavy Rain: <math>2.117 \times 10^{-3} &lt; RTEPC</math> kg/m<sup>2</sup> and TYPPC = [rain]</li> </ul>
STpvt	pavement state	<p>METRo is run for each link in the road network model to determine pavement state estimations.</p> <ul style="list-style-type: none"> <li>• Dry Road: The water reservoir contains less than 0.01 mm and the ice/snow reservoir contains less than .2 mm of water equivalent.</li> <li>• Wet road: The water reservoir contains more than 0.01 mm of water.</li> <li>• Ice/Snow: The ice/snow reservoir contains more than 0.2 mm of water equivalent.</li> <li>• Water/Snow: Both of the reservoirs (water and ice/snow) contain more than 0.2 mm of water equivalent.</li> <li>• Dew: Condensation on the road when the temperature of the surface of the road is above the freezing point.</li> <li>• Frost: Condensation on the road when the temperature of the surface of the road is below the freezing point or water already present on the road is turning into ice.</li> </ul>	<p>METRo is run for each link in the road network model to determine pavement state predictions.</p> <ul style="list-style-type: none"> <li>• Dry Road: Each reservoir (water and ice/snow) contains less than 0.01 mm of liquid water equivalent.</li> <li>• Wet road: The water reservoir contains more than 0.01 mm of water.</li> <li>• Ice/Snow: The ice/snow reservoir contains more than 0.2 mm of water equivalent.</li> <li>• Water/Snow: Both of the reservoirs (water and ice/snow) contain more than 0.2 mm of water equivalent.</li> <li>• Dew: Condensation on the road when the temperature of the surface of the road is above the freezing point.</li> <li>• Frost: Condensation on the road when the temperature of the surface of the road is below the freezing point or water already present on the road is turning into ice.</li> </ul>
trflnk	traffic	The estimated speed value for each link is divided by the speed limit for that link.	The predicted speed value for each link is divided by the speed limit for that link.



## APPENDIX D. ALERT DEFINITIONS

**Table 10.Traffic Alert Definitions**

Type	Algorithm	Extent	Reference	Notify	Icon
Incident	EVT=Incident	point	TMC	n/a	
Work Zone	EVT=Work zone	link	TMC	n/a	

Source: FHWA, 2019

**Table 11. Weather Alert Definitions.**

Type	Algorithm	geoExtent	Reference	Notify	Icon
Medium Winter Precip	PCCAT= [Medium Freezing Rain, Medium Snow, Medium Ice Pellets]	area	Pikalert	Y	
Heavy Winter Precip	PCCAT= [Heavy Freezing Rain, Heavy Snow, Heavy Ice Pellets]	area	Pikalert	Y	
Medium Precip	PCCAT= [Medium Rain]	area	Pikalert	N	
Heavy Precip	PCCAT= [Heavy Rain]	area	Pikalert	Y	
Flood Stage Action	n/a	point	AHPS	n/a	
Flood Stage Flood	n/a	point	AHPS	n/a	
Low Visibility	VIS < 0.2 mi	area	RAP	Y	

Source: FHWA, 2019

**Table 12. Road Condition Alert Definitions.**

Type	Algorithm	geoExtent	Reference	Notify	Icon
Low Visibility	VIS < 0.2 mi	area	RAP	Y	
Ice on Bridge	STPVT= [ice]	segment	METRo	Y	
Flooded Road	DPHLNK > 0 in.	segment	AHPS	Y	

Source: FHWA, 2019

**Table 13. Tropical Storm Categories.**

Type	Algorithm	Extent	Reference	Notify	Icon
Subtropical Depression	From NHC	point	NHC	n/a	
Subtropical Storm	From NHC	point	NHC	n/a	
Tropical Depression	From NHC	point	NHC	n/a	
Tropical Storm	From NHC	point	NHC	n/a	
Hurricane	From NHC	point	NHC	n/a	
Major Hurricane	From NHC	point	NHC	n/a	

## Appendix E. Class Documentation

---

### *imrcp.collect.AHPS Class Reference*

#### Public Member Functions

- **AHPS ()**
- boolean **start ()** throws Exception
- void **reset ()**
- void **execute ()**
- void **downloadFile ()**

#### Additional Inherited Members

---

#### Detailed Description

Collects .shp files from the National Weather Service's Advanced Hydrologic Prediction Services that contain observed and forecasted flood stages for rivers and streams across the United States.

#### Author

Federal Highway Administration

---

#### Constructor & Destructor Documentation

##### *imrcp.collect.AHPS.AHPS ()*

Default constructor.

---

#### Member Function Documentation

##### *void imrcp.collect.AHPS.downloadFile ()*

Downloads the configured file

##### *void imrcp.collect.AHPS.execute ()*

Attempts to make a connection to the **AHPS** website and downloads a data file if there is a new one available

Reimplemented from **imrcp.system.BaseBlock** (*p.95*).

##### *void imrcp.collect.AHPS.reset ()*

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.collect.Collector** (*p.114*).

##### *boolean imrcp.collect.AHPS.start () throws Exception*

Attempts to download the file and then sets a schedule to execute on a fixed interval.

*Returns*

true if no Exceptions are thrown

*Exceptions*

<i>Exception</i>	
------------------	--

Reimplemented from **imrcp.system.BaseBlock** (*p.99*).

---

*The documentation for this class was generated from the following file:*

- collect/AHPS.java

---

## [imrcp.store.AHPSStore Class Reference](#)

### **Public Member Functions**

- `void getData (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)`

### **Protected Member Functions**

- `FileWrapper getNewFileWrapper ()`

### **Additional Inherited Members**

---

#### **Detailed Description**

**FileCache** that manages Advanced Hydrologic Prediction Service (AHPS) flood stage forecast and observation shapefiles.

#### *Author*

Federal Highway Administration

---

#### **Member Function Documentation**

`void imrcp.store.AHPSStore.getData (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)`

Determines the files that match the query and adds any observations that match the query from those files to the **ImrcpResultSet**

Reimplemented from **imrcp.system.BaseBlock** (*p.96*).

*FileWrapper imrcp.store.AHPSStore.getNewFileWrapper () [protected]*

#### *Returns*

a new **AHPSWrapper**

Reimplemented from **imrcp.store.FileCache** (*p.212*).

---

*The documentation for this class was generated from the following file:*

- store/AHPSStore.java
- 

---

## [imrcp.store.AHPSWrapper Class Reference](#)

### **Public Member Functions**

- `void load (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId) throws Exception`
- `void cleanup (boolean bDelete)`

## Public Attributes

- `ArrayList< ArrayList< int[] > > m_oPolygons = new ArrayList()`

## Additional Inherited Members

---

### Detailed Description

Parses and creates **Obs** from Advance Hydrologic Prediction System shapefiles.

#### Author

Federal Highway Administration

---

## Member Function Documentation

`void imrcp.store.AHPSWrapper.cleanup (boolean bDelete)`

Clears the observation list

#### Parameters

<code>bDelete</code>	does nothing for this class
----------------------	-----------------------------

Reimplemented from **imrcp.store.FileWrapper** (p.223).

`void imrcp.store.AHPSWrapper.load (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId) throws Exception`

Parses the AHPS .tgz shapefile, creating flood stage and alert observations. Then it parses the configured inundation polygon (**AHPSStore#m\_sPolygonFile**) and creates the polygons that can be drawn on the map and any alert or pavement state observations if there are roadway segments that are flooded.

Reimplemented from **imrcp.store.FileWrapper** (p.224).

---

## Member Data Documentation

`ArrayList< ArrayList< int[] > > imrcp.store.AHPSWrapper.m_oPolygons = new ArrayList()`

Contains the geometric definitions of inundation polygons. The polygons are defined by a list of rings using **imrcp.system.Arrays**. The format of each ring is [insertion point, hole flag, minx, miny, maxx, maxy, x1, y1, x2, y2, ... xn, yn, x1, y1]

---

*The documentation for this class was generated from the following file:*

- `store/AHPSWrapper.java`
- 

## imrcp.comp.AlertCondition Class Reference

### Public Member Functions

- `AlertCondition (String[] sCondition)`
- boolean `evaluate (double dObsValue)`

## Public Attributes

- int **m\_nObsType**
  - double **m\_dMin**
  - double **m\_dMax**
  - String **m\_sUnits**
- 

## Detailed Description

Stores information about conditions that generate alerts for different observation types.

### Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

*imrcp.comp.AlertCondition.AlertCondition (String[] sCondition)*

Constructs an **AlertCondition** from a String[]

### Parameters

<i>strings</i>	
----------------	--

## Member Function Documentation

*boolean imrcp.comp.AlertCondition.evaluate (double dObsValue)*

Checks if the value matches the condition. If **m\_dMin** and **m\_dMax** are the same it checks if the given value is equal to **m\_dMin** otherwise it checks if the given value is in between **m\_dMin** and **m\_dMax**.  $m_dMin \leq dObsValue < m_dMax$

### Parameters

<i>dObsValue</i>	value to check
------------------	----------------

### Returns

true if the value matches the condition.

---

## Member Data Documentation

*double imrcp.comp.AlertCondition.m\_dMax*

Maximum value to match this condition (exclusive)

*double imrcp.comp.AlertCondition.m\_dMin*

Minimum value to match this condition (inclusive)

*int imrcp.comp.AlertCondition.m\_nObsType*

IMRCP Observation type

*String imrcp.comp.AlertCondition.m\_sUnits*

Units used for **m\_dMin** and **m\_dMax**

---

*The documentation for this class was generated from the following file:*

- comp/AlertCondition.java
- 

## imrcp.comp.Alerts Class Reference

### Public Member Functions

- boolean **start** () throws Exception
- void **reset** ()
- void **process** (String[] sMessage)
- void **createAlerts** (String sStore, int[] nObsTypes, long lStartTime, long lEndTime, long lValidTime)

### Additional Inherited Members

---

### Detailed Description

This class is used to generate **Alerts** when observed or forecasted values meet certain conditions.

#### Author

Federal Highway Administration

---

### Member Function Documentation

**void imrcp.comp.Alerts.createAlerts (String sStore, int[] nObsTypes, long lStartTime, long lEndTime, long lValidTime)**

Creates alerts by comparing observations from the given store in the specified time range to the configured **imrcp.comp.Alerts.Rule**s

#### Parameters

<i>sStore</i>	Name of the FileCache to query
<i>nObsTypes</i>	Observation types to query
<i>lStartTime</i>	start time of query
<i>lEndTime</i>	end time of query
<i>lValidTime</i>	valid(reference time) of query

**void imrcp.comp.Alerts.process (String[] sMessage)**

Called when a message from **imrcp.system.Directory** is received. If the notification is for new data, **Alerts#createAlerts(java.lang.String, int[], long, long, long)** is called.

#### Parameters

<i>sMessage</i>	[BaseBlock message is from, message name, start time in millis for data query, end time in millis for data query, valid time in millis for data query, obstype1, obstype2, ..., obstypen]
-----------------	---

Reimplemented from **imrcp.system.BaseBlock** (p.97).

#### `void imrcp.comp.Alerts.reset ()`

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.system.BaseBlock** (p.98).

#### `boolean imrcp.comp.Alerts.start () throws Exception`

Creates and stores **imrcp.comp.Alerts.Rule**s by parsing the configuration object of this block.

##### *Returns*

true if no exceptions are thrown

##### *Exceptions*

<i>Exception</i>
------------------

Reimplemented from **imrcp.system.BaseBlock** (p.99).

---

*The documentation for this class was generated from the following file:*

- comp/Alerts.java
- 

## imrcp.store.AlertsCsvWrapper Class Reference

### Public Member Functions

- **AlertsCsvWrapper** (int[] nObsTypes)
- **void load** (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId)  
throws Exception

### Additional Inherited Members

---

#### Detailed Description

Contains the logic to parse and create observations for Alert Csv files

#### *Author*

Federal Highway Administration

---

#### Constructor & Destructor Documentation

##### `imrcp.store.AlertsCsvWrapper.AlertsCsvWrapper (int[] nObsTypes)`

Wrapper for **CsvWrapper#CsvWrapper(int[])**

##### *Parameters*

<i>nObsTypes</i>
------------------

observation types this file contains

---

## Member Function Documentation

`void imrcp.store.AlertsCsvWrapper.load (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId) throws Exception`

Parses and creates observations from the given file. Alert Csv files can have lines appended to them while the file is already in memory so each time this method is called `m_oCsvFile` is reopened and skips any lines that have already been read.

Reimplemented from `imrcp.store.CsvWrapper` (*p.125*).

---

*The documentation for this class was generated from the following file:*

- store/AlertsCsvWrapper.java
- 

## imrcp.store.AlertsStore Class Reference

### Public Member Functions

- `void getData (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)`

### Protected Member Functions

- `FileWrapper getNewFileWrapper ()`

### Additional Inherited Members

---

### Detailed Description

`FileCache` that manages alert csv files generated by `imrcp.comp.Alerts`

#### Author

Federal Highway Administration

---

## Member Function Documentation

`void imrcp.store.AlertsStore.getData (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)`

Determines the files that match the query and calls `CsvStore#getDataFromFile(imrcp.store.ImrcpResultSet, int, long, long, int, int, int, long, imrcp.store.CsvWrapper)` for each of those files

Reimplemented from `imrcp.store.CsvStore` (*p.123*).

`FileWrapper imrcp.store.AlertsStore.getNewFileWrapper () [protected]`

#### Returns

a new `AlertsCsvWrapper` with the configured observation types

Reimplemented from `imrcp.store.CsvStore` (*p.124*).

---

*The documentation for this class was generated from the following file:*

- store/AlertsStore.java
- 

## [imrcp.web.layers.AreaLayerServlet Class Reference](#)

### Public Member Functions

- void **reset** ()

### Protected Member Functions

- void **buildObsResponseContent** (JsonGenerator oOutputGenerator, **ObsRequest** oObsRequest) throws Exception
- void **buildObsChartResponseContent** (JsonGenerator oOutputGenerator, **ObsChartRequest** oObsRequest) throws Exception

### [Additional Inherited Members](#)

---

#### Detailed Description

Handles requests from the IMRCP Map UI when Area layer objects are clicked or when a chart for an areal observations is requested

#### Author

Federal Highway Administration

---

#### Member Function Documentation

**void imrcp.web.layers.AreaLayerServlet.buildObsChartResponseContent (JsonGenerator oOutputGenerator, ObsChartRequest oObsRequest) throws Exception [protected]**

Add the response to the given JSON stream for requests made from the IMRCP Map UI to create a chart for areal observations

Reimplemented from **imrcp.web.layers.LayerServlet** (*p.322*).

**void imrcp.web.layers.AreaLayerServlet.buildObsResponseContent (JsonGenerator oOutputGenerator, ObsRequest oObsRequest) throws Exception [protected]**

Add the response to the given JSON stream for requests made from the IMRCP Map UI when an Area Layer object is clicked.

Reimplemented from **imrcp.web.layers.LayerServlet** (*p.323*).

**void imrcp.web.layers.AreaLayerServlet.reset ()**

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.web.layers.LayerServlet** (*p.324*).

---

*The documentation for this class was generated from the following file:*

- web/layers/AreaLayerServlet.java
- 

## imrcp.system.Arrays Class Reference

### Static Public Member Functions

- static double[] **newDoubleArray** ()
  - static double[] **newDoubleArray** (int nCapacity)
  - static Iterator< double[]> **iterator** (double[] dSrc, double[] dDest, int nStart, int nStep)
  - static double[] **ensureCapacity** (double[] dVals, int nDemand)
  - static int **size** (double[] dVals)
  - static double[] **add** (double[] dVals, double d1)
  - static double[] **add** (double[] dVals, double d1, double d2)
  - static double[] **addAndUpdate** (double[] dVals, double d1, double d2)
  - static double[] **add** (double[] dVals, double[] dMore)
  - static int[] **newIntArray** ()
  - static int[] **newIntArray** (int nCapacity)
  - static Iterator< int[]> **iterator** (int[] nSrc, int[] nDest, int nStart, int nStep)
  - static int[] **ensureCapacity** (int[] nVals, int nDemand)
  - static int **size** (int[] nVals)
  - static int[] **add** (int[] nVals, int n1)
  - static int[] **add** (int[] nVals, int n1, int n2)
  - static int[] **add** (int[] nVals, int[] nMore)
  - static int[] **insert** (int[] nVals, int n1, int nIndex)
  - static int **binarySearch** (int[] nVals, int n1)
- 

### Detailed Description

Contains methods for creating and using growable arrays. These were implemented to use in place of ArrayLists for primitive types to use less memory by avoiding using **java.lang.Integer**, **java.lang.Double**, and **java.lang.Long**

### Author

Federal Highway Administration

---

### Member Function Documentation

#### **static double[] imrcp.system.Arrays.add (double[] dVals, double d1) [static]**

Adds the given value to the given growable array and returns the reference of the array that contains all of the original values and the new value. When using this method always assign the given growable array to the return value of the function since it could be a different reference if the array needed to grow.

#### Parameters

<i>dVals</i>	growable array to add to
<i>d1</i>	value to add

### Returns

The reference of the growable array after the value has been added, this could be a different reference than the one passed into the method.

**static double[] imrcp.system.Arrays.add (double[] dVals, double d1, double d2) [static]**

Adds the given values to the given growable array in order and returns the reference of the array that contains all of the original values and the new values. When using this method always assign the given growable array to the return value of the function since it could be a different reference if the array needed to grow.

### Parameters

<i>dVals</i>	growable array to add to
<i>d1</i>	first value to add
<i>d2</i>	second value to add

### Returns

The reference of the growable array after the values have been added, this could be a different reference than the one passed into the method.

**static double[] imrcp.system.Arrays.add (double[] dVals, double[] dMore) [static]**

Copies the values in the given "more" array to the end of the given growable array and returns the reference of the array that contains all of the original values and the new values. When using this method always assign the given growable array to the return value of the function since it could be a different reference if the array needed to grow.

### Parameters

<i>dVals</i>	growable array to add to
<i>dMore</i>	array of values to add to the end of growable array

### Returns

The reference of the growable array after the values have been added, this could be a different reference than the one passed into the method.

**static int[] imrcp.system.Arrays.add (int[] nVals, int n1) [static]**

Adds the given value to the given growable array and returns the reference of the array that contains all of the original values and the new value. When using this method always assign the given growable array to the return value of the function since it could be a different reference if the array needed to grow.

### Parameters

<i>nVals</i>	growable array to add to
<i>n1</i>	value to add

### Returns

The reference of the growable array after the value has been added, this could be a different reference than the one passed into the method.

**static int[] imrcp.system.Arrays.add (int[] nVals, int n1, int n2) [static]**

Adds the given values to the given growable array in order and returns the reference of the array that contains all of the original values and the new values. When using this method

always assign the given growable array to the return value of the function since it could be a different reference if the array needed to grow.

#### Parameters

<i>nVals</i>	growable array to add to
<i>n1</i>	first value to add
<i>n2</i>	second value to add

#### Returns

The reference of the growable array after the values have been added, this could be a different reference than the one passed into the method.

**static int[] imrcp.system.Arrays.add (int[] nVals, int[] nMore) [static]**

Copies the values in the given "more" array to the end of the given growable array and returns the reference of the array that contains all of the original values and the new values. When using this method always assign the given growable array to the return value of the function since it could be a different reference if the array needed to grow.

#### Parameters

<i>nVals</i>	growable array to add to
<i>nMore</i>	array of values to add to the end of growable array

#### Returns

The reference of the growable array after the values have been added, this could be a different reference than the one passed into the method.

**static double[] imrcp.system.Arrays.addAndUpdate (double[] dVals, double d1, double d2) [static]**

This method is used for growable arrays that represent geometries. The positions 1,2,3,4 contain the bounding box of the geometry so the array has the format [insertion point, min x, min y, max x, max y, x0, y0, x1, y1, ...] Calls **add(double[], double, double)** then updates the bounding box based off of the x and y coordinates added.

#### Parameters

<i>dVals</i>	growable array to add to
<i>d1</i>	x value to add
<i>d2</i>	y value to add

#### Returns

The reference of the growable array after the values have been added, this could be a different reference than the one passed into the method.

**static int imrcp.system.Arrays.binarySearch (int[] nVals, int n1) [static]**

Wrapper for **java.util.Arrays#binarySearch(int[], int, int, int)** passing the 1 for the start and the insertion point for the end, essentially searching the entire growable array.

#### Parameters

<i>nVals</i>	growable array to search for the given value, the array must be sorted
--------------	--

<i>n1</i>	value to search for in the array
-----------	----------------------------------

#### Returns

index of the search key, if it is contained in the array within the specified range; otherwise, (- (insertion point) - 1). The insertion point is defined as the point at which the key would be inserted into the array: the index of the first element in the range greater than the key, or toIndex if all elements in the range are less than the specified key. Note that this guarantees that the return value will be  $\geq 0$  if and only if the key is found

**static double[] imrcp.system.Arrays.ensureCapacity (double[] dVals, int nDemand) [static]**

Grows the array, if necessary, to be able to add an additional number of elements equal to the given demand.

#### Parameters

<i>dVals</i>	growable array to ensure the capacity on
<i>nDemand</i>	the number of elements that need to be added

#### Returns

if the array didn't need to grow, the reference to the array that was passed into the function. If the array did need to grow, the reference to a new growable array with a larger capacity

**static int[] imrcp.system.Arrays.ensureCapacity (int[] nVals, int nDemand) [static]**

Grows the array, if necessary, to be able to add an additional number of elements equal to the given demand.

#### Parameters

<i>nVals</i>	growable array to ensure the capacity on
<i>nDemand</i>	the number of elements that need to be added

#### Returns

if the array didn't need to grow, the reference to the array that was passed into the function. If the array did need to grow, the reference to a new growable array with a larger capacity

**static int[] imrcp.system.Arrays.insert (int[] nVals, int n1, int nIndex) [static]**

Inserts the given value at the specified position in the growable array.

#### Parameters

<i>nVals</i>	growable array to insert the value in
<i>n1</i>	value to insert into the growable array
<i>nIndex</i>	position to insert the value at

#### Returns

The reference of the growable array after the value has been inserted, this could be a different reference than the one passed into the method.

`static Iterator<double[]> imrcp.system.Arrays.iterator (double[] dSrc, double[] dDest, int nStart, int nStep) [static]`

Gets an iterator for the source double array that copies values into the destination array, starting at the given start position of the source and incrementing by the given step each time `java.util.Iterator#next()` is called.

*Parameters*

<code>dSrc</code>	source array to iterate over
<code>dDest</code>	destination array that gets filled with values from the source array
<code>nStart</code>	initial position to start at in the source array
<code>nStep</code>	number to increment by each time <code>java.util.Iterator#next()</code> is called.

*Returns*

Iterator for the source array

`static Iterator<int[]> imrcp.system.Arrays.iterator (int[] nSrc, int[] nDest, int nStart, int nStep) [static]`

Gets an iterator for the source int array that copies values into the destination array, starting at the given start position of the source and incrementing by the given step each time `java.util.Iterator#next()` is called.

*Parameters*

<code>nSrc</code>	source array to iterate over
<code>nDest</code>	destination array that gets filled with values from the source array
<code>nStart</code>	initial position to start at in the source array
<code>nStep</code>	number to increment by each time <code>java.util.Iterator#next()</code> is called.

*Returns*

Iterator for the source array

`static double[] imrcp.system.Arrays.newDoubleArray () [static]`

Gets a new growable double array

*Returns*

a new growable double array

`static double[] imrcp.system.Arrays.newDoubleArray (int nCapacity) [static]`

Get a new growable double array with the given initial capacity

*Parameters*

<code>nCapacity</code>	initial size of the array
------------------------	---------------------------

*Returns*

a new growable double array with the given capacity

`static int[] imrcp.system.Arrays.newIntArray () [static]`

Gets a new growable int array

*Returns*

a new growable int array

***static int[] imrcp.system.Arrays.newIntArray (int nCapacity) [static]***

Get a new growable int array with the given initial capacity

*Parameters*

<i>nCapacity</i>	initial size of the array
------------------	---------------------------

*Returns*

a new growable int array with the given capacity

***static int imrcp.system.Arrays.size (double[] dVals) [static]***

Gets the number of elements in the array aka the insertion point for the next element.

*Parameters*

<i>dVals</i>	growable array to get the size of
--------------	-----------------------------------

*Returns*

number of elements in the array aka the insertion point for the next element

***static int imrcp.system.Arrays.size (int[] nVals) [static]***

Gets the number of elements in the array aka the insertion point for the next element.

*Parameters*

<i>nVals</i>	growable array to get the size of
--------------	-----------------------------------

*Returns*

number of elements in the array aka the insertion point for the next element

---

*The documentation for this class was generated from the following file:*

- system/Arrays.java
- 

## imrcp.system.AsyncQ< T > Class Template Reference

### Public Member Functions

- **AsyncQ (IRunTarget< T > iDelegate)**
- **AsyncQ (int nMaxThreads, IRunTarget< T > iDelegate)**
- **void setMaxThreads (int nMaxThreads)**
- **void queue (T oT)**
- **void stop ()**
- **void run ()**
- **void run (T e)**

### Protected Member Functions

- **AsyncQ ()**
-

## Detailed Description

Asynchronous queue object processing. Objects are processed as soon as they are added to the queue.

### Parameters

<T>	template type. Must be specified when creating a new instance of <b>AsyncQ</b> .
-----	--

### Author

Federal Highway Administration

### Version

1.0

---

## Constructor & Destructor Documentation

*imrcp.system.AsyncQ< T >.AsyncQ () [protected]*

### Default Constructor

Creates a new instance of **AsyncQ**.

*imrcp.system.AsyncQ< T >.AsyncQ (IRunTarget< T > iDelegate)*

Creates a new instance of **AsyncQ** with the defined run target.

### Parameters

<i>iDelegate</i>	an object whose run method will be executed to process the objects in the queue.
------------------	--

*imrcp.system.AsyncQ< T >.AsyncQ (int nMaxThreads, IRunTarget< T > iDelegate)*

Creates a new instance of **AsyncQ** with the defined run target and the specified thread queue depth.

### Parameters

<i>nMaxThreads</i>	max threads to allocate to the new instance
<i>iDelegate</i>	an object whose run method will be executed to process the objects in the queue.

---

## Member Function Documentation

*void imrcp.system.AsyncQ< T >.queue (T oT)*

Adds an object to the queue for processing. If there are no current objects in the queue, processing will begin.

### Parameters

<i>oT</i>	the object of type T to add to the queue
-----------	--

*void imrcp.system.AsyncQ< T >.run ()*

Process all objects that are in the queue.

`void imrcp.system.AsyncQ< T >.run (T e)`

The default run target performs no operations on the object. This should be overridden in subclasses to perform necessary processing.

*Parameters*

<code>e</code>	target object to run.
----------------	-----------------------

`void imrcp.system.AsyncQ< T >.setMaxThreads (int nMaxThreads)`

Sets the max number of threads to the supplied integer value.

*Parameters*

<code>nMaxThreads</code>	new max threads.
--------------------------	------------------

---

*The documentation for this class was generated from the following file:*

- system/AsyncQ.java
- 

## imrcp.system.BaseBlock Class Reference

### Public Member Functions

- `void startService ()`
- `boolean start () throws Exception`
- `void init ()`
- `void notify (String sMessageName, String... sResource)`
- `void receive (String[] sMessage)`
- `void register ()`
- `long[] status ()`
- `void destroy ()`
- `void stopService ()`
- `boolean stop () throws Exception`
- `void setConfig ()`
- `BlockConfig getConfig ()`
- `ResultSet getData (int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)`
- `void run ()`
- `void execute ()`
- `void executeTest ()`
- `String getName ()`
- `void setName (String sName)`
- `void setLogger ()`
- `void run (String[] e)`
- `void process (String[] e)`

### Public Attributes

- `int m_nId`

### Protected Member Functions

- `BaseBlock ()`
- `void reset ()`
- `void getData (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)`

- synchronized void **setError** ()
- synchronized boolean **checkAndSetStatus** (int nStatus, int nCheck)

#### Protected Attributes

- int **m\_nSchedId**
- String **m\_sInstanceName**
- final AsyncQ<String[]> **m\_oNotifications** = new AsyncQ(this)
- Logger **m\_oLogger**
- BlockConfig **m\_oConfig**
- boolean **m\_bTest** = false

#### Additional Inherited Members

---

#### Detailed Description

This class acts as the base class for system components. It contains the methods and variables to be able to register with the directory to communicate with other components. Extends HttpServlet so that any component of the system can receive http requests.

#### Author

Federal Highway Administration

---

#### Constructor & Destructor Documentation

##### *imrcp.system.BaseBlock.BaseBlock () [protected]*

Default constructor. Sets the status to **INIT** and sets the id

---

#### Member Function Documentation

##### *synchronized boolean imrcp.system.BaseBlock.checkAndSetStatus (int nStatus, int nCheck) [protected]*

Sets the status of the block to nStatus as long as the current status is nCheck. If nCheck is **OVERRIDESTATUS** then the status is set to nStatus regardless of the current status.

#### Parameters

<i>nStatus</i>	value to set the status to
<i>nCheck</i>	value to check the current status to

#### Returns

true if the status is set to nStatus, otherwise false

##### *void imrcp.system.BaseBlock.destroy ()*

Wrapper for **stopService** , this gets called when the container managing servlets is terminating the servlet

Implements **imrcp.system.ImrcpBlock** (p.273).

##### *void imrcp.system.BaseBlock.execute ()*

Child classes implement this function to execute their process on a fixed interval schedule.

Reimplemented in [imrcp.collect.AHPS](#) (p.78), [imrcp.collect.BlueToad](#) (p.104), [imrcp.collect.CAP](#) (p.107), [imrcp.collect.Geotab](#) (p.229), [imrcp.collect.GFS](#) (p.245), [imrcp.collect.Inrix](#) (p.298), [imrcp.collect.LAc2c](#) (p.305), [imrcp.collect.LADOTD511](#) (p.308), [imrcp.collect.NHCKmz](#) (p.390), [imrcp.collect.OhGo](#) (p.422), [imrcp.collect.RemoteGrid](#) (p.472), [imrcp.collect.WxDESub](#) (p.569), [imrcp.comp.DataAssimilation](#) (p.127), [imrcp.comp.PeCat](#) (p.448), [imrcp.forecast.mdss.Metro](#) (p.332), [imrcp.forecast.mlp.MLPHurricane](#) (p.353), [imrcp.forecast.mlp.MLPUpdate](#) (p.365), [imrcp.store.FileCache](#) (p.211), [imrcp.store.SpatialFileCache](#) (p.498), [imrcp.system.Locks](#) (p.326), [imrcp.web.MonitorServlet](#) (p.368), [imrcp.web.NetworkGeneration](#) (p.389), [imrcp.web.Scenarios](#) (p.486), and [imrcp.web.Subscriptions](#) (p.507).

### `void imrcp.system.BaseBlock.executeTest ()`

Child classes implement this function to execute their process on a fixed interval schedule if the `m_bTest` flag is true.

### `BlockConfig imrcp.system.BaseBlock.getConfig ()`

Gets the configuration key/value pairs for this block, `m_oConfig`

#### *Returns*

The configuration key/value pairs

Implements [imrcp.system.ImrcpBlock](#) (p.273).

### `void imrcp.system.BaseBlock.getData (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime) [protected]`

This method is meant to be overridden by children classes to implement how the block returns data. The ImrcpResultSet that is passed as the first parameter is filled with the obs from the block that match the query parameters

#### *Parameters*

<code>oReturn</code>	ImrcpResultSet that will be filled with obs
<code>nType</code>	IMRCP observation type id
<code>lStartTime</code>	start time of the query in milliseconds since Epoch
<code>lEndTime</code>	end time of the query in milliseconds since Epoch
<code>nStartLat</code>	lower bound of latitude in decimal degrees scaled to 7 decimal places
<code>nEndLat</code>	upper bound of latitude in decimal degrees scaled to 7 decimal places
<code>nStartLon</code>	lower bound of longitude in decimal degrees scaled to 7 decimal places
<code>nEndLon</code>	upper bound of longitude in decimal degrees scaled to 7 decimal places
<code>lRefTime</code>	reference time in milliseconds since Epoch (observations received after this time will not be included)

Reimplemented in [imrcp.store.AHPSSStore](#) (p.80), [imrcp.store.AlertsStore](#) (p.85), [imrcp.store.BinObsStore](#) (p.101), [imrcp.store.CAPStore](#) (p.110), [imrcp.store.CsvStore](#) (p.123), [imrcp.store.GeotabStore](#) (p.231), [imrcp.store.GribStore](#) (p.249), [imrcp.store.MetroStore](#) (p.341), [imrcp.store.NHCStore](#) (p.392), [imrcp.store.TrafficEventStore](#) (p.538), [imrcp.store.TrafficSpeedStore](#) (p.539), [imrcp.store.WeatherStore](#) (p.564), and [imrcp.store.WxDESubStore](#) (p.570).

*ResultSet imrcp.system.BaseBlock.getData (int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)*

This method creates a new **ImrcpObsResultSet** and if the status of the block is **RUNNING** or **IDLE** calls **getData(imrcp.store.ImrcpResultSet, int, long, long, int, int, int, long)**

#### Parameters

<i>nType</i>	IMRCP observation type id
<i>lStartTime</i>	start time of the query in milliseconds since Epoch
<i>lEndTime</i>	end time of the query in milliseconds since Epoch
<i>nStartLat</i>	lower bound of latitude in decimal degrees scaled to 7 decimal places
<i>nEndLat</i>	upper bound of latitude in decimal degrees scaled to 7 decimal places
<i>nStartLon</i>	lower bound of longitude in decimal degrees scaled to 7 decimal places
<i>nEndLon</i>	upper bound of longitude in decimal degrees scaled to 7 decimal places
<i>lRefTime</i>	reference time in milliseconds since Epoch (observations received after this time will not be included).

Reimplemented in **imrcp.store.CAPStore** (p.110), **imrcp.store.ObsView** (p.420), and **imrcp.store.TrafficEventStore** (p.538).

*String imrcp.system.BaseBlock.getName ()*

Gets the instance name of this block

#### Returns

The instance name of the block, **m\_sInstanceName**

Implements **imrcp.system.ImrcpBlock** (p.273).

*void imrcp.system.BaseBlock.init ()*

Initializes the **BaseBlock**, wrapper for **setLogger()** and **setConfig()**

Implements **imrcp.system.ImrcpBlock** (p.273).

*void imrcp.system.BaseBlock.notify (String sMessageName, String... sResource)*

Wrapper for **Directory#notifyBlocks(java.lang.String[])** , sending the notification message created from the message name and resource to the BaseBlocks subscribed to this block.

#### Parameters

<i>sMessageName</i>	type of message being sent
<i>sResource</i>	resources of the message

Implements **imrcp.system.ImrcpBlock** (p.273).

*void imrcp.system.BaseBlock.process (String[] e)*

Child classes implement this function to handle notification messages from other blocks.

#### Parameters

<i>e</i>	Notification message from another block
----------	---

Reimplemented in [imrcp.comp.Alerts](#) (p.83), [imrcp.comp.InrixComp](#) (p.299), [imrcp.comp.LAc2cDetectorsComp](#) (p.306), [imrcp.comp.Notifications](#) (p.397), [imrcp.comp.PcCat](#) (p.448), [imrcp.forecast.mlp.MLPHurricane](#) (p.353), [imrcp.forecast.mlp.MLPPredict](#) (p.357), [imrcp.store.BinObsStore](#) (p.101), [imrcp.store.FileCache](#) (p.214), [imrcp.store.SpatialFileCache](#) (p.500), [imrcp.comp.EventComp](#) (p.204), and [imrcp.forecast.mdss.Metro](#) (p.332).

### `void imrcp.system.BaseBlock.receive (String[] sMessage)`

Wrapper for `AsyncQ#queue (java.lang.Object)`, queuing the given notification message to be processed.

#### Parameters

<code>sMessage</code>	Notification message from another <b>BaseBlock</b> this block is subscribed to.
-----------------------	---

Implements [imrcp.system.ImrcpBlock](#) (p.273).

### `void imrcp.system.BaseBlock.register ()`

Registers the **BaseBlock** to the `imrcp.system.Directory`

Implements [imrcp.system.ImrcpBlock](#) (p.273).

### `void imrcp.system.BaseBlock.reset () [protected]`

Sets configurable member variables from the **BlockConfig** object, should be overridden by child classes.

Reimplemented in [imrcp.collect.AHPS](#) (p.78), [imrcp.collect.BlueToad](#) (p.104), [imrcp.collect.CAP](#) (p.107), [imrcp.collect.Collector](#) (p.114), [imrcp.collect.Geotab](#) (p.230), [imrcp.collect.GFS](#) (p.246), [imrcp.collect.Inrix](#) (p.298), [imrcp.collect.LAc2c](#) (p.305), [imrcp.collect.LADOTD511](#) (p.309), [imrcp.collect.NHCKmz](#) (p.391), [imrcp.collect.OhGo](#) (p.422), [imrcp.collect.RemoteGrid](#) (p.472), [imrcp.collect.WxDESub](#) (p.569), [imrcp.comp.Alerts](#) (p.84), [imrcp.comp.DataAssimilation](#) (p.127), [imrcp.comp.EventComp](#) (p.204), [imrcp.comp.InrixComp](#) (p.299), [imrcp.comp.LAc2cDetectorsComp](#) (p.306), [imrcp.comp.LAc2cEventsComp](#) (p.308), [imrcp.comp.LADOTD511Line](#) (p.310), [imrcp.comp.LADOTD511Point](#) (p.312), [imrcp.comp.Notifications](#) (p.397), [imrcp.comp.PcCat](#) (p.448), [imrcp.forecast.mdss.Metro](#) (p.333), [imrcp.forecast.mlp.MLPBlock](#) (p.349), [imrcp.forecast.mlp.MLPHurricane](#) (p.353), [imrcp.forecast.mlp.MLPPredict](#) (p.357), [imrcp.forecast.mlp.MLPUpdate](#) (p.366), [imrcp.geosrv.DEM](#) (p.192), [imrcp.geosrv.WayNetworks](#) (p.561), [imrcp.store.BinObsStore](#) (p.102), [imrcp.store.FileCache](#) (p.214), [imrcp.store.GribStore](#) (p.250), [imrcp.store.MetroStore](#) (p.341), [imrcp.store.WeatherStore](#) (p.564), [imrcp.system.ExtMapping](#) (p.209), [imrcp.system.Locks](#) (p.326), [imrcp.web.LaneServlet](#) (p.317), [imrcp.web.layers.AreaLayerServlet](#) (p.86), [imrcp.web.layers.LayerServlet](#) (p.324), [imrcp.web.layers.RoadLayerServlet](#) (p.481), [imrcp.web.MonitorServlet](#) (p.368), [imrcp.web.NetworkGeneration](#) (p.389), [imrcp.web.NotificationServlet](#) (p.398), [imrcp.web.Scenarios](#) (p.487), [imrcp.web.SecureBaseBlock](#) (p.492), [imrcp.web.Subscriptions](#) (p.508), [imrcp.web.tiles.GribTileCache](#) (p.250), [imrcp.web.tiles.NcfTileCache](#) (p.372), [imrcp.web.tiles.TileCache](#) (p.522), [imrcp.web.tiles.TileServlet](#) (p.525), [imrcp.web.USCenPlaceLookup](#) (p.545), and [imrcp.web.UserSettingsServlet](#) (p.548).

### `void imrcp.system.BaseBlock.run ()`

This method is usually called as a fixed interval execution by **Scheduling**. If this block's status is **IDLE** then `execute ()` (or `executeTest ()` if in test mode) is called and its status is changed to **RUNNING**. If no errors occur during `execute ()` then the status is changed back to **IDLE**.

Reimplemented in [imrcp.forecast.mlp.MLPPredict](#) (p.357), and [imrcp.forecast.mlp.MLPUpdate](#) (p.366).

`void imrcp.system.BaseBlock.run (String[] e)`

This method is called when a notification from another block. If the status of this block is **IDLE** then `process (java.lang.String[])` is called, otherwise the notification is placed back in the queue.

*Parameters*

<code>e</code>	Notification message from another block
----------------	---

`void imrcp.system.BaseBlock.setConfig ()`

Sets `m_oConfig` to a new **BlockConfig** using this blocks class and instance name.

`synchronized void imrcp.system.BaseBlock.setError () [protected]`

Sets the status of the block to **ERROR**

Reimplemented in **imrep.collect.RemoteGrid** (p.472).

`void imrcp.system.BaseBlock.setLogger ()`

Sets the Log4J logger object

`void imrcp.system.BaseBlock.setName (String sName)`

Sets the instance name of this block

*Parameters*

<code>sName</code>	The instance name of the block
--------------------	--------------------------------

Implements **imrcp.system.ImrcpBlock** (p.273).

`boolean imrcp.system.BaseBlock.start () throws Exception`

This function is to be overridden by each **BaseBlock** to implement what the block needs to do to start its service

*Returns*

true if no Exceptions are thrown

*Exceptions*

<code>Exception</code>
------------------------

Reimplemented in **imrep.collect.AHPS** (p.78), **imrcp.collect.BlueToad** (p.104), **imrcp.collect.CAP** (p.107), **imrcp.collect.Geotab** (p.230), **imrcp.collect.GFS** (p.246), **imrcp.collect.Inrix** (p.298), **imrcp.collect.LAc2c** (p.305), **imrcp.collect.LADOTD511** (p.309), **imrcp.collect.NHCKmz** (p.391), **imrcp.collect.OhGo** (p.422), **imrcp.collect.RemoteGrid** (p.472), **imrcp.collect.WxDESub** (p.569), **imrcp.comp.Alerts** (p.84), **imrep.comp.DataAssimilation** (p.127), **imrep.comp.LADOTD511Line** (p.310), **imrep.comp.LADOTD511Point** (p.312), **imrep.comp.PcCat** (p.448), **imrcp.forecast.mdss.Metro** (p.333), **imrcp.forecast.mlp.MLPHurricane** (p.354), **imrcp.forecast.mlp.MLPPredict** (p.358), **imrcp.forecast.mlp.MLPUpdate** (p.366), **imrcp.geosrv.WayNetworks** (p.561), **imrep.store.FileCache** (p.214), **imrcp.store.SpatialFileCache** (p.500), **imrcp.system.ExtMapping** (p.209), **imrcp.system.Locks** (p.326), **imrcp.web.MonitorServlet** (p.368), **imrcp.web.NetworkGeneration** (p.389), **imrcp.web.Scenarios** (p.487), **imrcp.web.Subscriptions** (p.509), **imrcp.web.tiles.NHCTiles** (p.394), **imrcp.web.tiles.TileCache** (p.522), and **imrcp.web.USCenPlaceLookup** (p.545).

`void imrcp.system.BaseBlock.startService ()`

Called by **Directory** once all of the BaseBlocks this **BaseBlock** is subscribed have finished their start up sequence. It calls `reset()` and `start()`. If no problems occur then the the block's status will be **IDLE**

`long[] imrcp.system.BaseBlock.status ()`

Gets a long array with the current status of the block and the last time the status changed.

*Returns*

[status, time in milliseconds since Epoch the since last changed]

Implements **imrcp.system.ImrcpBlock** (p.274).

`boolean imrcp.system.BaseBlock.stop () throws Exception`

This function is to be overridden by each **BaseBlock** to implement what the block needs to do to stop its service and clean up resources.

*Returns*

true if no Exceptions are thrown

*Exceptions*

<i>Exception</i>
------------------

Reimplemented in **imrcp.collect.RemoteGrid** (p.472), **imrcp.comp.PcCat** (p.449), **imrcp.forecast.mdss.Metro** (p.333), and **imrcp.store.FileCache** (p.214).

`void imrcp.system.BaseBlock.stopService ()`

This method stops the service of this **BaseBlock**. If this block has a task scheduled for execution with **Scheduling**, that task is canceled. `stop()` which is overriden by child classes is called and this block status is changed to **STOPPED**

---

## Member Data Documentation

`boolean imrcp.system.BaseBlock.m_bTest = false [protected]`

Flag indicating if this component should run in test mode or not

`int imrcp.system.BaseBlock.m_nId`

System id used by the **imrcp.system.Directory**

`int imrcp.system.BaseBlock.m_nSchedId [protected]`

Schedule id used by **imrcp.system.Scheduling**

`BlockConfig imrcp.system.BaseBlock.m_oConfig [protected]`

Object that contains the configuration key/value pairs

`Logger imrcp.system.BaseBlock.m_oLogger [protected]`

Log4j Logger

`final AsyncQ<String[]> imrcp.system.BaseBlock.m_oNotifications = new  
AsyncQ(this) [protected]`

Queue that stores the notification messages received from other system components

*String imrcp.system.BaseBlock.m\_sInstanceName [protected]*

The name of the system component

---

*The documentation for this class was generated from the following file:*

- system/BaseBlock.java
- 

## imrcp.store.BinObsStore Class Reference

### Public Member Functions

- void **reset** ()
- void **process** (String[] sMessage)
- void **getData** (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)

### Protected Member Functions

- **GriddedFileWrapper getNewFileWrapper** ()

### Additional Inherited Members

---

### Detailed Description

**FileCache** that manages IMRCP's gridded binary observation files

#### Author

Federal Highway Administration

---

### Member Function Documentation

**void imrcp.store.BinObsStore.getData (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)**

Determines the files that match the query and calls **DataObsWrapper#getData(int, long, int, int, int, int)** for each of those files

Reimplemented from **imrcp.system.BaseBlock** (p.96).

*GriddedFileWrapper imrcp.store.BinObsStore.getNewFileWrapper () [protected]*

#### Returns

a new **DataObsWrapper** with the configured observation types

Reimplemented from **imrcp.store.FileCache** (p.212).

**void imrcp.store.BinObsStore.process (String[] sMessage)**

Called when a message from **imrcp.system.Directory** is received. If the message is "file download" and the file name contains the configured string, then the file is processed using the super implementation

#### Parameters

<i>sMessage</i>
-----------------

Reimplemented from **imrcp.store.FileCache** (p.214).

#### *void imrcp.store.BinObsStore.reset ()*

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.store.FileCache** (p.214).

---

*The documentation for this class was generated from the following file:*

- store/BinObsStore.java
- 

## imrcp.system.BlockConfig Class Reference

### Public Member Functions

- **BlockConfig** (String sClass, String sName)
  - String **getString** (String sKey, String sDefault)
  - int **getInt** (String sKey, int nDefault)
  - double **getDouble** (String sKey, double dDefault)
  - String[] **getStringArray** (String sKey, String sDefault)
  - int[] **getIntArray** (String sKey, int nDefault)
- 

### Detailed Description

This class is used to store key/value pairs read from configuration files. Base blocks can then retrieve the values using the different get methods

#### Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

#### *imrcp.system.BlockConfig.BlockConfig (String sClass, String sName)*

Constructs a **BlockConfig** for the given class and instance name by retrieving configuration values from **imrcp.system.Config**

#### Parameters

<i>sClass</i>	fully quanlified class name
<i>sName</i>	instance name of base block

---

## Member Function Documentation

#### *double imrcp.system.BlockConfig.getDouble (String sKey, double dDefault)*

Gets the associated value of the given key as a double. If the key is not found, the default is returned.

*Parameters*

<i>sKey</i>	key to search for
<i>nDefault</i>	default value

*Returns*

The associated value of the given key if the key exists in the map, otherwise the default value.

*int imrcp.system.BlockConfig.getInt (String sKey, int nDefault)*

Gets the associated value of the given key as an integer. If the key is not found, the default is returned.

*Parameters*

<i>sKey</i>	key to search for
<i>nDefault</i>	default value

*Returns*

The associated value of the given key if the key exists in the map, otherwise the default value.

*int[] imrcp.system.BlockConfig.getIntArray (String sKey, int nDefault)*

Gets the associated value of the given key as an integer array. If the key is not found, the default is returned in an array of size 1.

*Parameters*

<i>sKey</i>	key to search for
<i>nDefault</i>	default value

*Returns*

The associated value of the given key if the key exists in the map, otherwise the default value.  
If the default is null, an empty array is returned.

*String imrcp.system.BlockConfig.getString (String sKey, String sDefault)*

Gets the associated value of the given key as a String. If the key is not found, the default is returned.

*Parameters*

<i>sKey</i>	key to search for
<i>sDefault</i>	default value

*Returns*

The associated value of the given key if the key exists in the map, otherwise the default value.

*String[] imrcp.system.BlockConfig.getStringArray (String sKey, String sDefault)*

Gets the associated value of the given key as a String array. If the key is not found, the default is returned in an array of size 1.

*Parameters*

<i>sKey</i>	key to search for
<i>sDefault</i>	default value

#### Returns

The associated value of the given key if the key exists in the map, otherwise the default value. If the default is null, an empty array is returned.

---

*The documentation for this class was generated from the following file:*

- system/BlockConfig.java
- 

## imrcp.collect.BlueToad Class Reference

### Public Member Functions

- void **reset** ()
- boolean **start** ()
- void **execute** ()

### Additional Inherited Members

---

#### Detailed Description

Generic collector for downloading data files from **BlueToad** traffic monitoring systems.

#### Author

Federal Highway Administration

---

### Member Function Documentation

#### `void imrcp.collect.BlueToad.execute ()`

Attempts to log in and download the data file from the configured URL

Reimplemented from **imrcp.system.BaseBlock** (p.95).

#### `void imrcp.collect.BlueToad.reset ()`

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.collect.Collector** (p.114).

#### `boolean imrcp.collect.BlueToad.start ()`

Sets the fixed interval schedule of execution

#### Returns

true

Reimplemented from **imrcp.system.BaseBlock** (p.99).

---

*The documentation for this class was generated from the following file:*

- collect/BlueToad.java

---

## imrcp.system.BufferedInStream Class Reference

### Public Member Functions

- **BufferedInStream** (InputStream oInputStream, int nSize)
- **BufferedInStream** (InputStream oInputStream)
- int **read** () throws IOException
- int **read** (byte[] yBuf, int nOff, int nLen) throws IOException
- long **skip** (long lBytes) throws IOException

### Static Protected Attributes

- static final int **BUFFER\_SIZE** = 16384

---

### Detailed Description

Provides similar functionality as **java.io.BufferedInputStream** without some of the error checking which increases performance.

### Author

Federal Highway Administration

---

### Constructor & Destructor Documentation

*imrcp.system.BufferedInStream.BufferedInStream (InputStream oInputStream, int nSize)*

Constructs a **BufferedInStream** wrapping the given InputStream, using a buffer of the given size

#### Parameters

<i>oInputStream</i>	InputStream to wrap
<i>nSize</i>	Buffer size

*imrcp.system.BufferedInStream.BufferedInStream (InputStream oInputStream)*

Constructs a **BufferedInStream** wrapping the given InputStream, using the default buffer size

#### Parameters

<i>oInputStream</i>
---------------------

---

### Member Data Documentation

*final int imrcp.system.BufferedInStream.BUFFER\_SIZE = 16384 [static], [protected]*

Default 16k buffer size

---

*The documentation for this class was generated from the following file:*

- system/BufferedInStream.java

---

## imrcp.store.ByteObsEntryData Class Reference

### Public Member Functions

- double **getValue** (int nHrz, int nVrt)
- void **setTimeDim** (int nIndex)

### Additional Inherited Members

---

#### Detailed Description

An **EntryData** for IMRCP gridded binary observation files with values that can be stored as bytes.

#### Author

Federal Highway Administration

---

#### Member Function Documentation

**double imrcp.store.ByteObsEntryData.getValue (int nHrz, int nVrt)**

Gets the value of the grid at the given x and y coordinates

##### Parameters

<i>nHrz</i>	x coordinate
<i>nVrt</i>	y coordinate

##### Returns

value of the grid at the given x and y coordinates, if the coordinates are out of range,  
Double.NaN is returned

Reimplemented from **imrcp.store.EntryData** (p.202).

**void imrcp.store.ByteObsEntryData.setTimeDim (int nIndex)**

Entry datas of this type do nothing have multiple time dimensions so does nothing.

Reimplemented from **imrcp.store.EntryData** (p.203).

---

*The documentation for this class was generated from the following file:*

- store/ByteObsEntryData.java
- 

## imrcp.collect.CAP Class Reference

### Public Member Functions

- boolean **start** ()
- void **reset** ()
- void **execute** ()

## Additional Inherited Members

---

### Detailed Description

#### *Author*

Federal Highway Administration

---

### Member Function Documentation

#### `void imrcp.collect.CAP.execute ()`

Attempts to make a connection to the **CAP** server and downloads a data file if there is a new one available

Reimplemented from **imrcp.system.BaseBlock** (*p.95*).

#### `void imrcp.collect.CAP.reset ()`

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.system.BaseBlock** (*p.98*).

#### `boolean imrcp.collect.CAP.start ()`

Sets the fixed interval schedule of execution

#### *Returns*

true

Reimplemented from **imrcp.system.BaseBlock** (*p.99*).

---

*The documentation for this class was generated from the following file:*

- collect/CAP.java
- 

## imrcp.store.CAPObs Class Reference

### Public Member Functions

- **CAPObs** (int nObsTypeId, int nContribId, **Id** oObjId, long lObsTime1, long lObsTime2, long lTimeRecv, int nLat1, int nLon1, int nLat2, int nLon2, short tElev, double dValue, short tConf, String sDetail, String sCapId, String sUrl)
- synchronized Area **getArea** ()
- boolean **matches** (int nObsType, long lStartTime, long lEndTime, long lRefTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon)

### Public Attributes

- ArrayList< int[]> **m\_oPoly** = new ArrayList()
- String **m\_sCapId**
- Area **m\_oArea**
- String **m\_sUrl**

## Additional Inherited Members

---

### Detailed Description

An extension of **Obs** with a few additional fields specific to observations received from the National Weather Service's Common Alerting Protocol system.

### Author

Federal Highway Administration

---

### Constructor & Destructor Documentation

`imrcp.store.CAPObs.CAPObs (int nObsTypeId, int nContribId, Id oObjId, long lObsTime1, long lObsTime2, long lTimeRecv, int nLat1, int nLon1, int nLat2, int nLon2, short tElev, double dValue, short tConf, String sDetail, String sCapId, String sUrl)`

Constructs a new **CAPObs** with the given parameters

#### Parameters

<code>nObsTypeId</code>	IMRCP observation type id, see <code>imrcp.system.ObsType</code> for possible values
<code>nContribId</code>	IMRCP contributor Id which is computed by converting an up to a 6 character alphanumeric string using base 36
<code>oObjId</code>	Id of the object this obs is associated with, in the case of <b>CAPObs</b> this should be <code>imrcp.system.Id#NULLID</code>
<code>lObsTime1</code>	time in milliseconds since Epoch the observation starts being valid
<code>lObsTime2</code>	time in milliseconds since Epoch the observation stops being valid
<code>lTimeRecv</code>	time in milliseconds since Epoch the observation was received, also known as its reference time
<code>nLat1</code>	minimum latitude of the polygon in decimal degrees scaled to 7 decimal places
<code>nLon1</code>	minimum longitude of the polygon in decimal degrees scaled to 7 decimal places
<code>nLat2</code>	maximum latitude of the polygon in decimal degrees scaled to 7 decimal places
<code>nLon2</code>	maximum longitude of the polygon in decimal degrees scaled to 7 decimal places
<code>tElev</code>	elevation in meters
<code>dValue</code>	alarm type, <code>imrcp.system.ObsType#LOOKUP</code> contains a complete list of types
<code>tConf</code>	confidence value
<code>sDetail</code>	Details associated with the obs
<code>sCapId</code>	Identifier used by the CAP system
<code>sUrl</code>	Url for the CAP alert

## Member Function Documentation

### *synchronized Area imrcp.store.CAPObs.getArea ()*

If `m_oArea` is null, converts the polygon stored in `m_oPoly` into an `Area` object. `m_oArea` is set to that object and it is returned.

#### *Returns*

The `Area` that defines the polygon for this `CAPObs` which is stored in `m_oArea`

### *boolean imrcp.store.CAPObs.matches (int nObsType, long lStartTime, long lEndTime, long lRefTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon)*

Does the normal checks of `imrcp.store.Obs#matches(int, long, long, long, int, int, int, int)` except for the spatial extents compares it to the polygon that is defined for the observations.

Reimplemented from `imrcp.store.Obs` (p.403).

---

## Member Data Documentation

### *Area imrcp.store.CAPObs.m\_oArea*

Area object used for clipped with vector tile boundaries

### *ArrayList<int[]> imrcp.store.CAPObs.m\_oPoly = new ArrayList()*

Each `int[]` is a ring that is a part of the polygon that defines the location of the CAP alert. Each ring is a set lon/lat pairs in decimal degrees scaled to 7 decimals places. Each ring should be a closed polygon (the first and last point are the same)

### *String imrcp.store.CAPObs.m\_sCapId*

Identifier used by the CAP system

### *String imrcp.store.CAPObs.m\_sUrl*

Url for the CAP alert

---

*The documentation for this class was generated from the following file:*

- store/CAPObs.java

---

## imrcp.store.CAPStore Class Reference

### Public Member Functions

- `ResultSet getData (int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)`
- `void getData (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)`

### Protected Member Functions

- `FileWrapper getNewFileWrapper ()`

## Additional Inherited Members

---

### Detailed Description

**FileCache** that manages .tar.gz shapefiles received from the National Weather Service's Common Alerting Protocol system.

### Author

Federal Highway Administration

---

### Member Function Documentation

`void imrcp.store.CAPStore.getData (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)`

Determines the files that match the query and adds all of the observations from those files that match the query.

Reimplemented from **imrcp.system.BaseBlock** (*p.96*).

`ResultSet imrcp.store.CAPStore.getData (int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)`

#### Returns

a new **ImrcpCapResultSet** filled with observations that match the query.

Reimplemented from **imrcp.system.BaseBlock** (*p.97*).

`FileWrapper imrcp.store.CAPStore.getNewFileWrapper () [protected]`

#### Returns

a new **CapWrapper**

Reimplemented from **imrcp.store.FileCache** (*p.212*).

---

*The documentation for this class was generated from the following file:*

- store/CAPStore.java
- 

## imrcp.store.CapWrapper Class Reference

### Public Member Functions

- `void load (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId)`  
throws Exception
- `void cleanup (boolean bDelete)`

## Additional Inherited Members

---

## Detailed Description

**FileWrapper** for parsing .tar.gz shapefiles containing alerts from the National Weather Service's Common Alerting Protocol system

### Author

Federal Highway Administration

---

## Member Function Documentation

***void imrcp.store.CapWrapper.cleanup (boolean bDelete)***

Does nothing for CapWrappers

Reimplemented from **imrcp.store.FileWrapper** (p.223).

***void imrcp.store.CapWrapper.load (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId) throws Exception***

Decompresses the .tar.gz file and loads the .shp and .dbf files into memory to be parsed to create observations for the defined CAP alerts that get added to **m\_oObs**

Reimplemented from **imrcp.store.FileWrapper** (p.224).

---

*The documentation for this class was generated from the following file:*

- store/CapWrapper.java

---

## imrcp.web.CenGroup Class Reference

### Public Member Functions

- **CenGroup** (char[] cKey)
- int **compareTo** (char[] o)

---

## Detailed Description

Represents a group of places found from the Cartographic Boundary Files provided by the United States Census Bureau. The places are grouped the first n characters of their name (currently n = 3 in **imrcp.web.USCenPlaceLookup**)

### Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

***imrcp.web.CenGroup.CenGroup (char[] cKey)***

Constructs a new **CenGroup** with the given key in a new char[]

### Parameters

<i>cKey</i>	Characters used to group places
-------------	---------------------------------

## Member Function Documentation

*int imrcp.web.CenGroup.compareTo (char[] o)*

Compares the first n characters to the characters in **m\_cKey** where n is **m\_cKey.length**

*See also*

[java.lang.Comparable::compareTo\(java.lang.Object\)](#)

---

*The documentation for this class was generated from the following file:*

- [web/CenGroup.java](#)
- 

## imrcp.web.CenPlace Class Reference

### Public Member Functions

- [\*\*CenPlace\*\* \(String sLabel\)](#)
- [\*\*CenPlace\*\* \(String sName, int nGeoId, int nStateFp, String\[\] sAbrevs\)](#)
- [\*\*String toString \(\)\*\*](#)

### Public Attributes

- [\*\*String m\\_sName\*\*](#)
- [\*\*int m\\_nGeoId\*\*](#)
- [\*\*int m\\_nStateFp\*\*](#)
- [\*\*String m\\_sLabel\*\*](#)

---

### Detailed Description

Represents a place (state, city, county) found in the Cartographic Boundary Files provided by the United States Census Bureau.

#### *Author*

Federal Highway Administration

---

### Constructor & Destructor Documentation

*imrcp.web.CenPlace.CenPlace (String sLabel)*

Constructs a **CenPlace** with the given label

#### *Parameters*

<i>sLabel</i>	label describing the place
---------------	----------------------------

*imrcp.web.CenPlace.CenPlace (String sName, int nGeoId, int nStateFp, String[] sAbrevs)*

Constructs a **CenPlace** setting the given parameters

#### *Parameters*

<i>sName</i>	name of place
--------------	---------------

<code>nGeoId</code>	geographic id of place
<code>nStateFp</code>	fips code of the state this place is in
<code>sAbbrevs</code>	abbreviation

---

## Member Function Documentation

`String imrcp.web.CenPlace.toString ()`

### Returns

The concatenation of `m_sName`, `m_nStateFp` and `m_nGeoId`

---

## Member Data Documentation

`int imrcp.web.CenPlace.m_nGeoId`

Geographic identifier used by the census

`int imrcp.web.CenPlace.m_nStateFp`

Fips code of the state the place is located

`String imrcp.web.CenPlace.m_sLabel`

Label used to describe the place

`String imrcp.web.CenPlace.m_sName`

Name of place

---

*The documentation for this class was generated from the following file:*

- `web/CenPlace.java`

---

## imrcp.collect.Collector Class Reference

### Public Member Functions

- `void reset ()`

### Protected Member Functions

- `String getDestFilename (long lFileTime, String... sStrings)`
- `String getDestFilename (long lFileTime, int nPageIndex, String... sStrings)`

### Protected Attributes

- `FilenameFormatter m_oDestFile`
- `FilenameFormatter m_oSrcFile`
- `String m_sBaseUrl`
- `int m_nOffset`
- `int m_nPeriod`
- `int m_nDelay`
- `int m_nRange`
- `int m_nFileFrequency`

## Additional Inherited Members

---

### Detailed Description

Base class containing functions and member variables used by various collectors

#### Author

Federal Highway Administration

---

### Member Function Documentation

***String imrcp.collect.Collector.getDestFilename (long lFileTime, int nPageIndex, String... sStrings) [protected]***

Uses `imrcp.collect.Collector#m_oDestFile` to call `with the correct timestamps based off of imrcp.collect.Collector#m_oDestFile` and `imrcp.collect.Collector#m_oDestFile`

#### Parameters

<code>lFileTime</code>	Expected collection time of file in milliseconds since the Epoch
<code>nPageIndex</code>	Forecast index from source files
<code>sStrings</code>	String array of values passed to <code>imrcp.system.FilenameFormatter#format(long, long, long, java.lang.String...)</code>

#### Returns

time dependent file name

Reimplemented in `imrcp.collect.GFS` (p.245).

***String imrcp.collect.Collector.getDestFilename (long lFileTime, String... sStrings) [protected]***

Calls `imrcp.collect.Collector#getDestFilename(long, int, java.lang.String...)` with a file index of zero to get the time dependent file name

#### Parameters

<code>lFileTime</code>	Expected collection time of file in milliseconds since the Epoch
<code>sStrings</code>	String array of values passed to <code>imrcp.system.FilenameFormatter#format(long, long, long, java.lang.String...)</code>

#### Returns

time dependent file name

***void imrcp.collect.Collector.reset ()***

Sets configurable member variables from the `BlockConfig` object, should be overridden by child classes.

Reimplemented from `imrcp.system.BaseBlock` (p.98).

Reimplemented in `imrcp.collect.AHPS` (p.78), `imrcp.collect.BlueToad` (p.104), `imrcp.collect.Geotab` (p.230), `imrcp.collect.GFS` (p.246), `imrcp.collect.Inrix` (p.298),

**imrcp.collect.LAc2c** (p.305), **imrcp.collect.OhGo** (p.422), **imrcp.collect.RemoteGrid** (p.472), and **imrcp.collect.WxDESub** (p.569).

---

## Member Data Documentation

*int imrcp.collect.Collector.m\_nDelay [protected]*

Number of milliseconds from collection time until the forecast is valid

*int imrcp.collect.Collector.m\_nFileFrequency [protected]*

Expected time in milliseconds between collection of consecutive files

*int imrcp.collect.Collector.m\_nOffset [protected]*

Schedule offset from midnight in seconds

*int imrcp.collect.Collector.m\_nPeriod [protected]*

Period of execution in seconds

*int imrcp.collect.Collector.m\_nRange [protected]*

Number of milliseconds the forecast is valid

*FilenameFormatter imrcp.collect.Collector.m\_oDestFile [protected]*

Object used to create time dependent file names to save on disk

*FilenameFormatter imrcp.collect.Collector.m\_oSrcFile [protected]*

Object used to create time dependent file names collected from remote sources

*String imrcp.collect.Collector.m\_sBaseUrl [protected]*

Base URL used for downloading data from remote sources

---

*The documentation for this class was generated from the following file:*

- collect/Collector.java
- 

## imrcp.store.NHCWrapper.ConeParser Class Reference

### Protected Member Functions

- void **parse** (InputStream oIn, int nSize) throws Exception
- 

### Detailed Description

Object used to parse the cone .kml files from the National Hurricane Center

---

## Member Function Documentation

*void imrcp.store.NHCWrapper.ConeParser.parse (InputStream oIn, int nSize) throws Exception [protected]*

Parses the given InputStream which should be at the start of a cone .kml file from the National Hurricane Center.

### Parameters

<i>oIn</i>	cone .kml file
<i>nSize</i>	length of the file in bytes

### Exceptions

<i>Exception</i>
------------------

---

*The documentation for this class was generated from the following file:*

- store/NHCWrapper.java
- 

## imrcp.system.Config Class Reference

### Public Member Functions

- **Config** (String sFilename)
- void **setSchedule** ()
- int **getInt** (String sClass, String sName, String sKey, int nDefault)
- String **getString** (String sClass, String sName, String sKey, String sDefault)
- String[] **getStringArray** (String sClass, String sName, String sKey, String sDefault)
- final void **run** ()
- HashMap< String, String[]> **getProps** (String sClass, String sName)

### Static Public Member Functions

- static **Config getInstance** ()
- 

### Detailed Description

This class is a singleton that manages the configurable parameters for the entire system.

### Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

*imrcp.system.Config.Config (String sFilename)*

Constructs a **Config** with the given IMRCP JSON configuration file path

### Parameters

<i>sFilename</i>
------------------

## Member Function Documentation

### *static Config imrcp.system.Config.getInstance () [static]*

Returns the singleton instance

#### *Returns*

The singleton instance, **g\_oConfig**

### *int imrcp.system.Config.getInt (String sClass, String sName, String sKey, int nDefault)*

Gets the integer configuration value with the given key for the given class and instance name. First the return value is set based on the class name and key. Then if a configuration entry exists based on the instance name and key, the return value is set to that value. If the key cannot be found for class or instance name, the default is returned.

#### *Parameters*

<i>sClass</i>	Fully Qualified Class Name of the Java class the configuration is being looked up for
<i>sName</i>	Instance name the configuration is being looked up for
<i>sKey</i>	Name of the configuration value being looked up
<i>nDefault</i>	Value to return if the key cannot be found

#### *Returns*

The configuration value that matches the given key for the given class name and instance name. Configuration by instance name overrides configuration by class name. If the key cannot be found for class or instance name, the default value is returned.

### *HashMap< String, String[]> imrcp.system.Config.getProps (String sClass, String sName)*

Adds all of the configuration items as String[]s for the given class and instance name to a HashMap and returns that HashMap

#### *Parameters*

<i>sClass</i>	Fully Qualified Class Name of the Java class the configuration is being looked up for
<i>sName</i>	Instance name the configuration is being looked up for

#### *Returns*

A HashMap with all of the configuration items for the given class and instance name with the keys being the name of the configuration item and the values being String[] representations of the configuration values

### *String imrcp.system.Config.getString (String sClass, String sName, String sKey, String sDefault)*

Gets the String configuration value with the given key for the given class and instance name. First the return value is set based on the class name and key. Then if a configuration entry exists based on the instance name and key, the return value is set to that value. If the key cannot be found for class or instance name, the default is returned.

#### Parameters

<i>sClass</i>	Fully Qualified Class Name of the Java class the configuration is being looked up for
<i>sName</i>	Instance name the configuration is being looked up for
<i>sKey</i>	Name of the configuration value being looked up
<i>sDefault</i>	Value to return if the key cannot be found

#### Returns

The configuration value that matches the given key for the given class name and instance name. Configuration by instance name overrides configuration by class name. If the key cannot be found for class or instance name, the default value is returned.

***String[] imrcp.system.Config.getStringArray (String sClass, String sName, String sKey, String sDefault)***

Gets the String array configuration value with the given key for the given class and instance name. First the return value is set based on the class name and key. Then if a configuration entry exists based on the instance name and key, the return value is set to that value. If the key cannot be found for class or instance name, the default is returned.

#### Parameters

<i>sClass</i>	Fully Qualified Class Name of the Java class the configuration is being looked up for
<i>sName</i>	Instance name the configuration is being looked up for
<i>sKey</i>	Name of the configuration value being looked up
<i>sDefault</i>	Value to return if the key cannot be found

#### Returns

The configuration value that matches the given key for the given class name and instance name. Configuration by instance name overrides configuration by class name. If the key cannot be found for class or instance name, the default value is returned. If the default value is not null it is returned as a String[] with length 1 storing the default value.

***final void imrcp.system.Config.run ()***

If the IMRCP JSON configuration file has been modified, reads the file and updates the reference of ***m\_oConfigArray*** to a new JSONArray that contains the contents of the file.

***void imrcp.system.Config.setSchedule ()***

Sets a schedule to execute on a fixed interval.

---

*The documentation for this class was generated from the following file:*

- system/Config.java
- 

***imrcp.system.CsvReader Class Reference***

#### Public Member Functions

- **CsvReader** (InputStream oInputStream, int nCols)
- **CsvReader** (InputStream oInputStream)

- int **readLine** () throws IOException
- boolean **isNull** (int nCol) throws IndexOutOfBoundsException
- double **parseDouble** (int nCol) throws IndexOutOfBoundsException, NumberFormatException
- long **parseLong** (int nCol) throws IndexOutOfBoundsException, NumberFormatException
- float **parseFloat** (int nCol) throws IndexOutOfBoundsException, NumberFormatException
- int **parseInt** (int nCol) throws IndexOutOfBoundsException, NumberFormatException
- String **parseString** (int nCol) throws IndexOutOfBoundsException
- int **parseString** (StringBuilder sBuf, int nCol) throws IndexOutOfBoundsException, NullPointerException
- void **getLine** (StringBuilder sLine) throws IOException

#### Protected Attributes

- int **m\_nCol**
- int[] **m\_nColEnds**
- StringBuilder **m\_sBuf** = new StringBuilder(BUFFER\_SIZE)
- int **m\_nState** = WHITESPACE

#### Static Protected Attributes

- static final int **DEFAULT\_COLS** = 80
- static final int **WHITESPACE** = 0
- static final int **STARTCOL** = 1
- static final int **QUOTECOLUMN** = 2
- static final int **INTERNALQUOTE** = 3

#### Detailed Description

Implementation of a buffered FilterInputStream used to read CSV files.

#### Author

Federal Highway Administration

#### Constructor & Destructor Documentation

##### *imrcp.system.CsvReader.CsvReader (InputStream oInputStream, int nCols)*

Constructs a **CsvReader** for the given InputStream and allocates memory for the given number of columns

#### Parameters

<i>oInputStream</i>	InputStream representing a CSV data stream
<i>nCols</i>	number of columns to initially create

##### *imrcp.system.CsvReader.CsvReader (InputStream oInputStream)*

Constructs a **CsvReader** for the given InputStream with the default number of columns

#### Parameters

<i>oInputStream</i>	InputStream representing a CSV data stream
---------------------	--

## Member Function Documentation

*void imrcp.system.CsvReader.getLine (StringBuilder sLine) throws IOException*

Reconstructs the current line held in **m\_sBuf** with the original commas and places the contents in the given StringBuilder

### Parameters

<i>sLine</i>	buffer to place the contents of the line in
--------------	---

### Exceptions

<i>IOException</i>
--------------------

*boolean imrcp.system.CsvReader.isNull (int nCol) throws IndexOutOfBoundsException*

Determines if the given column of the current line held in **m\_sBuf** contains nothing.

### Parameters

<i>nCol</i>	column index to check
-------------	-----------------------

### Returns

true if the column has no contents, otherwise false

### Exceptions

<i>IndexOutOfBoundsException</i>
----------------------------------

*double imrcp.system.CsvReader.parseDouble (int nCol) throws*

*IndexOutOfBoundsException, NumberFormatException*

Parses the given column of the current line held in **m\_sBuf** as a double

### Parameters

<i>nCol</i>	column index to parse
-------------	-----------------------

### Returns

the characters of the column parsed as a double

### Exceptions

<i>IndexOutOfBoundsException</i>
----------------------------------

<i>NumberFormatException</i>
------------------------------

*float imrcp.system.CsvReader.parseFloat (int nCol) throws*

*IndexOutOfBoundsException, NumberFormatException*

Parses the given column of the current line held in **m\_sBuf** as a float

### Parameters

<i>nCol</i>	column index to parse
-------------	-----------------------

*Returns*

the characters of the column parsed as a float

*Exceptions*

<i>IndexOutOfBoundsException</i>	
<i>NumberFormatException</i>	

*int imrcp.system.CsvReader.parseInt (int nCol) throws IndexOutOfBoundsException, NumberFormatException*

Parses the given column of the the current line held in **m\_sBuf** as a integer

*Parameters*

<i>nCol</i>	column index to parse
-------------	-----------------------

*Returns*

the characters of the column parsed as a integer

*Exceptions*

<i>IndexOutOfBoundsException</i>	
<i>NumberFormatException</i>	

*long imrcp.system.CsvReader.parseLong (int nCol) throws IndexOutOfBoundsException, NumberFormatException*

Parses the given column of the the current line held in **m\_sBuf** as a long

*Parameters*

<i>nCol</i>	column index to parse
-------------	-----------------------

*Returns*

the characters of the column parsed as a long

*Exceptions*

<i>IndexOutOfBoundsException</i>	
<i>NumberFormatException</i>	

*String imrcp.system.CsvReader.parseString (int nCol) throws IndexOutOfBoundsException*

Parses the given column of the the current line held in **m\_sBuf** as a String

*Parameters*

<i>nCol</i>	column index to parse
-------------	-----------------------

*Returns*

the characters of the column parsed as a String

*Exceptions*

<i>IndexOutOfBoundsException</i>
<i>NullPointerException</i>

*int imrcp.system.CsvReader.parseString (StringBuilder sBuf, int nCol) throws  
IndexOutOfBoundsException, NullPointerException*

Parses the given column of the the current line held in **m\_sBuf** as a String by placing it in the given StringBuilder

*Parameters*

<i>nCol</i>	column index to parse
<i>sBuf</i>	buffer to place the contents of the column in

*Returns*

the number of characters in the column

*Exceptions*

<i>IndexOutOfBoundsException</i>
<i>NullPointerException</i>
<i>IOException</i>

*int imrcp.system.CsvReader.readLine () throws IOException*

Reads a line of the CSV data stream into **m\_sBuf**

*Returns*

the number of columns read for the current line

*Exceptions*

<i>IOException</i>
--------------------

---

## Member Data Documentation

*final int imrcp.system.CsvReader.DEFAULT\_COLS = 80 [static], [protected]*

Default number of columns to allocate memory for

*final int imrcp.system.CsvReader.INTERNALQUOTE = 3 [static], [protected]*

State flag used to indicate a quote was found inside of a column

*int imrcp.system.CsvReader.m\_nCol [protected]*

Stores the column count for each line read

*int [] imrcp.system.CsvReader.m\_nColEnds [protected]*

Stores the index in **m\_sBuf** that each column ends at

*int imrcp.system.CsvReader.m\_nState = WHITESPACE [protected]*

Stores the current state of the reader

*StringBuilder imrcp.system.CsvReader.m\_sBuf = new*

*StringBuilder(BUFFER\_SIZE) [protected]*

Buffer used to store the characters of the current line being read

*final int imrcp.system.CsvReader.QUOTECOLUMN = 2 [static], [protected]*

State flag used to indicate a column started with a quote

*final int imrcp.system.CsvReader.STARTCOL = 1 [static], [protected]*

State flag used to indicate reading a column has started

*final int imrcp.system.CsvReader.WHITESPACE = 0 [static], [protected]*

State flag used to indicate the default state or that a control character was read

---

*The documentation for this class was generated from the following file:*

- system/CsvReader.java
- 

## imrcp.store.CsvStore Class Reference

### Public Member Functions

- void **getData** (**ImrcpResultSet** oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)
- void **getDataFromFile** (**ImrcpResultSet** oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime, **CsvWrapper** oFile)

### Protected Member Functions

- **FileWrapper** **getNewFileWrapper** ()

### Additional Inherited Members

---

### Detailed Description

**FileCache** that manages IMRCP CSV observation files.

### Author

Federal Highway Administration

---

### Member Function Documentation

*void imrcp.store.CsvStore.getData (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)*

Determines the files that match the query and then calls **getDataFromFile(imrcp.store.ImrcpResultSet, int, long, long,**

`int, int, int, int, long, imrcp.store.CsvWrapper)` on each of those files

Reimplemented from **imrcp.system.BaseBlock** (p.96).

Reimplemented in **imrcp.store.AlertsStore** (p.85), **imrcp.store.GeotabStore** (p.231), **imrcp.store.TrafficEventStore** (p.538), and **imrcp.store.WxDESubStore** (p.570).

**void imrcp.store.CsvStore.getDataFromFile (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime, CsvWrapper oFile)**

Iterates through the observations of the file and adds them to the **ImrcpResultSet** if they match the query. Observations in the files are written in the order they are received so if there are multiple observations that have the same temporal and spatial extents, observation type, and contributor only the most recent of those observations is added to the **ImrcpResultSet**

#### Parameters

<i>oReturn</i>	<b>ImrcpResultSet</b> that will be filled with obs
<i>nType</i>	obstype id
<i>lStartTime</i>	start time of the query in milliseconds since Epoch
<i>lEndTime</i>	end time of the query in milliseconds since Epoch
<i>nStartLat</i>	lower bound of latitude in decimal degrees scaled to 7 decimal places
<i>nEndLat</i>	upper bound of latitude in decimal degrees scaled to 7 decimal places
<i>nStartLon</i>	lower bound of longitude in decimal degrees scaled to 7 decimal places
<i>nEndLon</i>	upper bound of longitude in decimal degrees scaled to 7 decimal places
<i>lRefTime</i>	reference time (observations received after this time will not be included)
<i>oFile</i>	<b>CsvWrapper</b> to get observations from

Reimplemented in **imrcp.store.GeotabStore** (p.231), and **imrcp.store.WxDESubStore** (p.570).

**FileWrapper imrcp.store.CsvStore.getNewFileWrapper () [protected]**

#### Returns

a new **CsvWrapper** with the configured observation types

Reimplemented from **imrcp.store.FileCache** (p.212).

Reimplemented in **imrcp.store.AlertsStore** (p.85), **imrcp.store.GeotabStore** (p.231), **imrcp.store.MLPStore** (p.364), **imrcp.store.TrafficEventStore** (p.538), and **imrcp.store.WxDESubStore** (p.570).

---

*The documentation for this class was generated from the following file:*

- store/CsvStore.java
-

## imrcp.store.CsvWrapper Class Reference

### Public Member Functions

- **CsvWrapper** (int[] nObsTypes)
- void **load** (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId) throws Exception
- void **cleanup** (boolean bDelete)

### Additional Inherited Members

---

### Detailed Description

**FileWrapper** for parsing IMRCP CSV observation files.

#### Author

Federal Highway Administration

---

### Constructor & Destructor Documentation

**imrcp.store.CsvWrapper.CsvWrapper (int[] nObsTypes)**

Constructs a new **CsvWrapper** with the given observation types

#### Parameters

<i>nObsTypes</i>	array of observation types this file provides
------------------	---

---

### Member Function Documentation

**void imrcp.store.CsvWrapper.cleanup (boolean bDelete)**

Closes **m\_oCsvFile** if it is not null and then deletes the file if the delete flag is true and its length is less than 85 bytes since that would be a file that is just the header or an invalid file.

Reimplemented from **imrcp.store.FileWrapper** (p.223).

**void imrcp.store.CsvWrapper.load (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId) throws Exception**

IMRCP CSV observation files can be appended to while the file is already in memory. If the file is not in memory **m\_oCsvFile** gets initialized and skips the header. Observations are then created and added to **m\_oObs** for each line of the file.

Reimplemented from **imrcp.store.FileWrapper** (p.224).

Reimplemented in **imrcp.store.AlertsCsvWrapper** (p.85), **imrcp.store.GeotabWrapper** (p.232), **imrcp.store.MLPCsv** (p.352), **imrcp.store.TrafficEventCsv** (p.537), and **imrcp.store.WxDECsv** (p.568).

---

*The documentation for this class was generated from the following file:*

- store/CsvWrapper.java
-

## imrcp.comp.DataAssimilation Class Reference

### Public Member Functions

- void **reset** ()
- boolean **start** () throws Exception
- void **execute** ()

### Protected Member Functions

- void **processFile** (long lTimestamp)
- void **estimate** (double[][] dGrid)
- double[][] **createGrid** (ImrcpObsResultSet oData)
- void **writeFile** (String sFilename, double[][] dGrid) throws Exception

### Protected Attributes

- String **m\_sStore**
- **FilenameFormatter m\_oDest**
- int **m\_nSrcObsType**
- int **m\_nDestObsType**
- int **m\_nHz**
- int **m\_nVrt**
- double **m\_dStartHz**
- double **m\_dStartVrt**
- double **m\_dHzStep**
- double **m\_dVrtStep**
- double **m\_dEndHz**
- double **m\_dEndVrt**
- double **m\_dBias**
- double **m\_dMinVal**
- double **m\_dMaxVal**
- int **m\_nKernelSize**
- int[] **m\_nRowRanges**
- double[][] **m\_dKernDists**
- int **m\_nRange**
- int **m\_nPeriod**
- int **m\_nOffset**

### Additional Inherited Members

---

#### Detailed Description

Class that uses Data Assimilation techniques to combine a numerical model, namely a class of Kriging algorithms, with observations to have a more complete coverage of conditions.

#### Author

Federal Highway Administration

---

#### Member Function Documentation

***double[][] imrcp.comp.DataAssimilation.createGrid (ImrcpObsResultSet oData) [protected]***

Creates the grid of values from the given list of observation to run the kriging algorithm on. Values in the grid can be the average of the observations founds in that cell, Double.NEGATIVE\_INFINITY meaning that cell is a candidate for the kriging

algorithm since there are observed values within the configured kernel size, or Double.NaN meaning the cell does not have any observations in it nor within the kernel size so it can be skipped for processing in **DataAssimilation#estimate(double[][])**

#### Parameters

<i>oData</i>	List containing observations
--------------	------------------------------

#### Returns

grid filled in with values from the given observations

### **void imrcp.comp.DataAssimilation.estimate (double dGrid[][]) [protected]**

Applies a kriging algorithm to estimate values for cells in the grid that are near cells that have observed values in them. If a cell contains Double.NEGATIVE\_INFINITY that means it is a cell that has sufficient observed values around it to make an estimation for its value. Cells that contain {@Double.NaN} do not have sufficient observed values around it and will be skipped.

#### Parameters

<i>dGrid</i>	Grid of values where each cell is either the average of the observed values at that location, Double.NEGATIVE_INFINITY , or Double.NaN
--------------	--

### **void imrcp.comp.DataAssimilation.execute ()**

Wrapper for **DataAssimilation#processFile (long)** using the current time floored to the most recent period of execution.

Reimplemented from **imrcp.system.BaseBlock** (p.95).

### **void imrcp.comp.DataAssimilation.processFile (long lTimestamp) [protected]**

Creates a file that contains kriged estimations for the given timestamp

#### Parameters

<i>lTimestamp</i>	Time used for queries in milliseconds since Epoch
-------------------	---

### **void imrcp.comp.DataAssimilation.reset ()**

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.system.BaseBlock** (p.98).

### **boolean imrcp.comp.DataAssimilation.start () throws Exception**

Calculates the values for **m\_nRowRanges** and **m\_dKernDists** to be used for future runs of the algorithm to save processing time that can be done once. Then sets a schedule to execute on a fixed interval.

#### Returns

true if no exceptions are thrown

#### Exceptions

<i>Exception</i>	
------------------	--

Reimplemented from **imrcp.system.BaseBlock** (p.99).

`void imrcp.comp.DataAssimilation.writeFile (String sFilename, double dGrid[][])`  
`throws Exception [protected]`

Writes a file in the IMRCP gridded binary observation file format using half-floating point precision values

*Parameters*

<code>sFilename</code>	filename on disk
<code>dGrid</code>	grid of observed and estimated values

*Exceptions*

<code>Exception</code>	
------------------------	--

---

## Member Data Documentation

`double imrcp.comp.DataAssimilation.m_dBias [protected]`

Offset value used when storing values in memory

`double imrcp.comp.DataAssimilation.m_dEndHz [protected]`

End x value of the observations grid

`double imrcp.comp.DataAssimilation.m_dEndVrt [protected]`

End y value of the observations grid

`double imrcp.comp.DataAssimilation.m_dHzStep [protected]`

Step size used for consecutive columns

`double [][] imrcp.comp.DataAssimilation.m_dKernDists [protected]`

Stores the distance from the center of the kernel to each cell in the kernel

`double imrcp.comp.DataAssimilation.m_dMaxVal [protected]`

Maximum value allowed for observations

`double imrcp.comp.DataAssimilation.m_dMinVal [protected]`

Minimum value allowed for observations

`double imrcp.comp.DataAssimilation.m_dStartHz [protected]`

Starting x value of the observations grid

`double imrcp.comp.DataAssimilation.m_dStartVrt [protected]`

Starting y value of the observation grid

`double imrcp.comp.DataAssimilation.m_dVrtStep [protected]`

Step size used for consecutive rows

`int imrcp.comp.DataAssimilation.m_nDestObsType [protected]`

Observation type of the observations create by the algorithms

`int imrcp.comp.DataAssimilation.m_nHzr [protected]`

Number of columns for the observations grid

*int imrcp.comp.DataAssimilation.m\_nKernelSize [protected]*

Number of cells from the current cell to include in kriging calculations

*int imrcp.comp.DataAssimilation.m\_nOffset [protected]*

Schedule offset from midnight in seconds

*int imrcp.comp.DataAssimilation.m\_nPeriod [protected]*

Period of execution in seconds

*int imrcp.comp.DataAssimilation.m\_nRange [protected]*

How long in milliseconds from Epoch the kriged values are valid

*int [] imrcp.comp.DataAssimilation.m\_nRowRanges [protected]*

Number of cells to use in each row of the kernel

*int imrcp.comp.DataAssimilation.m\_nSrcObsType [protected]*

Observation type of the source observations used to create new observations

*int imrcp.comp.DataAssimilation.m\_nVrt [protected]*

Number of rows for the observations grid

*FilenameFormatter imrcp.comp.DataAssimilation.m\_oDest [protected]*

Object used to create time dependent file names to save on disk

*String imrcp.comp.DataAssimilation.m\_sStore [protected]*

Name of the store to query for observations

---

*The documentation for this class was generated from the following file:*

- comp/DataAssimilation.java
- 

## imrcp.web.tiles.DataObsTileCache Class Reference

### Protected Member Functions

- **GriddedFileWrapper getDataWrapper (int nFormatIndex)**

### Additional Inherited Members

---

#### Detailed Description

Tile Cache implementation for caches that use **imrcp.store.DataObsWrapper**s

#### Author

Federal Highway Administration

---

## Member Function Documentation

*GriddedFileWrapper imrcp.web.tiles.DataObsTileCache.getDataWrapper (int nFormatIndex) [protected]*

### Returns

a new **imrcp.store.DataObsWrapper** with the configured observations types  
Reimplemented from **imrcp.web.tiles.TileCache** (p.521).

---

*The documentation for this class was generated from the following file:*

- web/tiles/DataObsTileCache.java

---

## imrcp.store.DataObsWrapper Class Reference

### Public Member Functions

- **DataObsWrapper** (int[] nObs)
- void **load** (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId)  
throws Exception
- ArrayList< **Obs** > **getData** (int nObsTypeId, long lTimestamp, int nLat1, int nLon1, int nLat2, int nLon2)
- void **cleanup** (boolean dDelete)
- double **getReading** (int nObsType, long lTimestamp, int nLat, int nLon, Date oTimeRecv)
- void **getIndices** (int nLon, int nLat, int[] nIndices)
- double **getReading** (int nObsType, long lTimestamp, int[] nIndices)

### Additional Inherited Members

---

#### Detailed Description

**FileWrapper** for parsing IMRCP gridded binary observation files.

#### Author

Federal Highway Administration

---

### Constructor & Destructor Documentation

*imrcp.store.DataObsWrapper.DataObsWrapper (int[] nObs)*

Constructs a new **DataObsWrapper** with the given observation types and sets **m\_oEntryMap** to an empty **ArrayList**

#### Parameters

---

<i>nObs</i>	array of observation types this file provides
-------------	---

## Member Function Documentation

`void imrcp.store.DataObsWrapper.cleanup (boolean dDelete)`

Does nothing for this type of file.

Reimplemented from `imrcp.store.FileWrapper` (p.223).

`ArrayList< Obs > imrcp.store.DataObsWrapper.getData (int nObsTypeId, long lTimestamp, int nLat1, int nLon1, int nLat2, int nLon2)`

Determines the grids that match the spatial extents of the request and creates **Obs** objects for each of the values.

### Parameters

<code>nObsTypeId</code>	observation type of the query
<code>lTimestamp</code>	timestamp of the query in milliseconds since Epoch
<code>nLat1</code>	minimum latitude of the query in decimal degrees scaled to 7 decimal places
<code>nLon1</code>	minimum longitude of the query in decimal degrees scaled to 7 decimal places
<code>nLat2</code>	maximum latitude of the query in decimal degrees scaled to 7 decimal places
<code>nLon2</code>	maximum longitude of the query in decimal degrees scaled to 7 decimal places

### Returns

ArrayList filled with **Obs** that match the query

`void imrcp.store.DataObsWrapper.getIndices (int nLon, int nLat, int[] nIndices)`

Fills the given int array with the x and y coordinates/indices of the grid that correspond to the given longitude and latitude.

### Parameters

<code>nLon</code>	longitude in decimal degrees scaled to 7 decimal places
<code>nLat</code>	latitude in decimal degrees scaled to 7 decimal places
<code>nIndices</code>	array to fill with x and y coordinates of the grid that correspond to the longitude and latitude [x,y]

Reimplemented from `imrcp.store.GriddedFileWrapper` (p.258).

`double imrcp.store.DataObsWrapper.getReading (int nObsType, long lTimestamp, int nLat, int nLon, Date oTimeRecv)`

Gets the value of the cell of the grid for the given observation type at the given lon/lat and time.

### Parameters

<code>nObsType</code>	desired observation type
<code>lTimestamp</code>	query time in milliseconds since Epoch
<code>nLat</code>	latitude of query in decimal degrees scaled to 7 decimal places
<code>nLon</code>	longitude of query in decimal degrees scaled to 7 decimal places
<code>oTimeRecv</code>	Date object to have its time set to the time the observation was received for some implementation

### Returns

the value of the cell of the grid for the given observation type at the given lon/lat and time,  
Double.NaN if a valid value does not exists for the query

Reimplemented from **imrcp.store.GriddedFileWrapper** (p.258).

*double imrcp.store.DataObsWrapper.getReading (int nObsType, long lTimestamp,  
int[] nIndices)*

Get the value of the cell of the grid for the given observation type at the given indices/coordinates

### Parameters

<i>nObsType</i>	desired observation type
<i>lTimestamp</i>	query time in milliseconds since Epoch
<i>nIndices</i>	[x coordinate, y coordinate]

### Returns

Reimplemented from **imrcp.store.GriddedFileWrapper** (p.259).

*void imrcp.store.DataObsWrapper.load (long lStartTime, long lEndTime, long  
lValidTime, String sFilename, int nContribId) throws Exception*

Creates the **EntryData** by parsing the file.

Reimplemented from **imrcp.store.FileWrapper** (p.224).

---

*The documentation for this class was generated from the following file:*

- store/DataObsWrapper.java
- 

## imrcp.store.grib.DataRep Class Reference

### Public Member Functions

- abstract void **read** (FilterInputStream oIn, int nSecLen, **Projection** oProj, float[][] fRows) throws IOException, InterruptedException
- float **decompressSimple** (byte yX1, byte yX2)

### Static Public Member Functions

- static **DataRep newDataRep** (DataInputStream oIn, int nSecLen) throws IOException

### Public Attributes

- double **m\_dR**
- double **m\_dEE**
- double **m\_dDD**
- int **m\_nBits**
- int **m\_nFieldCode**

---

### Detailed Description

Base class used to encapsulate fields from Section 5 (Data Representation Section) of GRIB2 files. See [https://www.nco.ncep.noaa.gov/pmb/docs/grib2/grib2\\_doc/](https://www.nco.ncep.noaa.gov/pmb/docs/grib2/grib2_doc/)

*Author*

Federal Highway Administration

---

**Member Function Documentation**

*float imrcp.store.grib.DataRep.decompressSimple (byte yX1, byte yX2)*

Decompresses the given bytes using the simple packing formula defined in section 2.3 of the GRIB2 specification

*Parameters*

yX1	0
yX2	Scaled (encoded) value

*Returns*

Original data value

*static DataRep imrcp.store.grib.DataRep.newDataRep (DataInputStream oIn, int nSecLen) throws IOException [static]*

Creates and returns a new **DataRep** from the given DataInputStream of a GRIB2 file.

*Parameters*

<i>oIn</i>	DataInputStream of the GRIB2 file. It should be ready to read the 6th byte of Section 5 (meaning the length (4 bytes) of the section and section number (1 byte) has been read)
<i>nSecLen</i>	length of the section in bytes

*Returns*

**DataRep** object defined by Section 5 or null if the template in the section has not been implemented

*Exceptions*

<i>IOException</i>
--------------------

*abstract void imrcp.store.grib.DataRep.read (FilterInputStream oIn, int nSecLen, Projection oProj, float fRows[][])) throws IOException, InterruptedException [abstract]*

Child classes must implement this function which specifies the logic to parse Section 7 depending on the template type defined in Section 5.

*Parameters*

<i>oIn</i>	DataInputStream of the GRIB2 file. It should be ready to read the 6th byte of Section 7 (meaning the length (4 bytes) of the section and section number (1 byte) has been read)
<i>nSecLen</i>	length of Section 7 in bytes
<i>oProj</i>	<b>Projection</b> object that contains the data from Section 3
<i>fRows</i>	The grid that gets filled with the data

### Exceptions

<i>IOException</i>	
<i>InterruptedException</i>	

Reimplemented in **imrcp.store.grib.DataRepPng** (p.135).

---

### Member Data Documentation

*double imrcp.store.grib.DataRep.m\_dDD*

DD = 10^D where D = Decimal scale factor for simple packing

*double imrcp.store.grib.DataRep.m\_dEE*

EE = 2^E where E = Binary scale factor for simple packing

*double imrcp.store.grib.DataRep.m\_dR*

Reference value for simple packing. IEEE 32-bit floating-point value

*int imrcp.store.grib.DataRep.m\_nBits*

Number of bits required to hold the resulting scaled and reference data values

*int imrcp.store.grib.DataRep.m\_nFieldCode*

Type of original field values from Code Table 5.1

---

*The documentation for this class was generated from the following file:*

- store/grib/DataRep.java
- 

## imrcp.store.grib.DataRepPng Class Reference

### Public Member Functions

- **DataRepPng** (DataInputStream oIn, int nSecLen) throws IOException
- void **read** (FilterInputStream oIn, int nSecLen, **Projection** oProj, float[][] fRows) throws IOException, InterruptedException
- void **run** ()

### Additional Inherited Members

---

### Detailed Description

Data Representation object used to parse Section 7 of GRIB2 files that have template 41 (**Grid point data - Portable Network Graphics (PNG)**) defined in Section 5. See <https://www.w3.org/TR/PNG-Compression.html> for the PNG specification

### Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

*imrcp.store.grib.DataRepPng.DataRepPng (DataInputStream oIn, int nSecLen)  
throws IOException*

Constructs a **DataRepPng** from the given input stream of a GRIB2 file.

### Parameters

<i>oIn</i>	DataInputStream of the GRIB2 file. It should be ready to read the 12th byte of Section 5 (meaning the length (4 bytes) of the section, section number (1 byte), total data points (4 bytes), and template number (2 bytes) has been read)
<i>nSecLen</i>	Length of the section in bytes

### Exceptions

<i>IOException</i>
--------------------

---

## Member Function Documentation

*void imrcp.store.grib.DataRepPng.read (FilterInputStream oIn, int nSecLen,  
Projection oProj, float fRows[][][]) throws IOException, InterruptedException*

Reads the data in Section 7 of the GRIB2 which represents a PNG file for this Data Representation. The data is filled into the grid represented by fRows. Parallel processing is used to read the next row of the png file into **m\_yLoadRow** while the current row is being decompressed and unfiltered.

Reimplemented from **imrcp.store.grib.DataRep** (p.133).

*void imrcp.store.grib.DataRepPng.run ()*

Reads a row of the PNG file into **m\_yLoadRow**

---

*The documentation for this class was generated from the following file:*

- store/grib/DataRepPng.java

---

## imrcp.system.dbf.DbfDouble Class Reference

### Public Member Functions

- **DbfDouble** (byte[] yChars, int nLength)
- void **parseRecord** (DataInputStream oDataInputStream) throws Exception
- int **getInt** ()
- long **getLong** ()
- float **getFloat** ()
- double **getDouble** ()
- String **toString** ()

### Additional Inherited Members

---

## Detailed Description

Holds data base file field natively as a double precision number.

### Author

Federal Highway Administration

### Version

1.0

---

## Constructor & Destructor Documentation

*imrcp.system.dbf.DbfDouble.DbfDouble (byte[] yChars, int nLength)*

Creates a new instance of **DbfDouble** with field name and data length defined.

### Parameters

<i>yChars</i>	array of ASCII byte characters
<i>nLength</i>	length of byte character buffer

---

## Member Function Documentation

*double imrcp.system.dbf.DbfDouble.getDouble ()*

Returns field data as a double precision number.

### Returns

double precision number value of field data

Reimplemented from **imrcp.system.dbf.DbfField** (p.138).

*float imrcp.system.dbf.DbfDouble.getFloat ()*

Returns field data as a floating point number.

### Returns

floating point number value of field data

Reimplemented from **imrcp.system.dbf.DbfField** (p.139).

*int imrcp.system.dbf.DbfDouble.getInt ()*

Returns field data as an integer.

### Returns

integer value of field data

Reimplemented from **imrcp.system.dbf.DbfField** (p.139).

*long imrcp.system.dbf.DbfDouble.getLong ()*

Returns field data as an long integer.

### Returns

long integer value of field data

Reimplemented from **imrcp.system.dbf.DbfField** (p.139).

`void imrcp.system.dbf.DbfDouble.parseRecord (DataInputStream oDataInputStream)  
throws Exception`

Reads the field and stores its contents as a double precision number.

*Parameters*

<code>oDataInputStream am</code>	data file input stream
--------------------------------------	------------------------

*Exceptions*

<code>java.lang.Exce ption</code>
---------------------------------------

Reimplemented from **imrcp.system.dbf.DbfField** (p.139).

`String imrcp.system.dbf.DbfDouble.toString ()`

Returns field data as a string.

*Returns*

string value of field data

---

*The documentation for this class was generated from the following file:*

- system/dbf/DbfDouble.java
- 

## imrcp.system.dbf.DbfField Class Reference

### Public Member Functions

- `void parseRecord (DataInputStream oDataInputStream) throws Exception`
- `byte[] getBytes ()`
- `abstract int getInt ()`
- `abstract long getLong ()`
- `abstract float getFloat ()`
- `abstract double getDouble ()`
- `boolean wasNull ()`
- `int compareTo (String sFieldName)`

### Static Public Member Functions

- `static int getstartIndex (byte[] yChars)`
- `static int getLength (byte[] yChars, int nstartIndex)`

### Static Public Attributes

- `static final int METADATA_LENGTH = 32`

### Protected Member Functions

- `DbfField ()`
- `DbfField (byte[] yChars, int nDataLength)`

### Protected Attributes

- `int m_nstartIndex`
- `int m_nDataLength`
- `String m_sName`

- byte[] **m\_yBuffer**
- 

#### Detailed Description

Holds data base file field

#### Author

Federal Highway Administration

#### Version

1.0

---

#### Constructor & Destructor Documentation

*imrcp.system.dbf.DbfField.DbfField () [protected]*

Creates a new instance of **DbfField**.

*imrcp.system.dbf.DbfField.DbfField (byte[] yChars, int nDataLength) [protected]*

Creates a new instance of **DbfField** with field name and data length defined.

#### Parameters

<i>yChars</i>	array of ASCII byte characters
<i>nDataLength</i>	length of byte character buffer

---

#### Member Function Documentation

*int imrcp.system.dbf.DbfField.compareTo (String sFieldName)*

Determines if the parameter string is equal to the field name.

#### Parameters

<i>sFieldName</i>	field name to compare
-------------------	-----------------------

#### Returns

0 if field names are equal

*byte[] imrcp.system.dbf.DbfField.getBytes ()*

Returns the array of ASCII byte characters contained in the field.

#### Returns

array of ASCII byte characters

*abstract double imrcp.system.dbf.DbfField.getDouble () [abstract]*

Abstract method to retrieve field data as a double precision number.

#### Returns

double precision number value of field data

Reimplemented in **imrcp.system.dbf.DbfDouble** (p.136), **imrcp.system.dbf.DbfLong** (p.141), and **imrcp.system.dbf.DbfString** (p.189).

*abstract float imrcp.system.dbf.DbfField.getFloat () [abstract]*

Abstract method to retrieve field data as a floating point number.

*Returns*

floating point number value of field data

Reimplemented in **imrcp.system.dbf.DbfDouble** (p.136), **imrcp.system.dbf.DbfLong** (p.141), and **imrcp.system.dbf.DbfString** (p.189).

*abstract int imrcp.system.dbf.DbfField.getInt () [abstract]*

Abstract method to retrieve field data as an integer.

*Returns*

integer value of field data

Reimplemented in **imrcp.system.dbf.DbfDouble** (p.136), **imrcp.system.dbf.DbfLong** (p.141), and **imrcp.system.dbf.DbfString** (p.189).

*static int imrcp.system.dbf.DbfField.getLength (byte[] yChars, int nstartIndex) [static]*

Returns the length of the ASCII character array without leading or trailing whitespace

*Parameters*

<i>yChars</i>	array of ASCII character bytes
<i>nstartIndex</i>	index of the first valid ASCII character

*Returns*

integer length of the valid ASCII character array

*abstract long imrcp.system.dbf.DbfField.getLong () [abstract]*

Abstract method to retrieve field data as a long integer.

*Returns*

long integer value of field data

Reimplemented in **imrcp.system.dbf.DbfDouble** (p.136), **imrcp.system.dbf.DbfLong** (p.141), and **imrcp.system.dbf.DbfString** (p.189).

*static int imrcp.system.dbf.DbfField.getStartIndex (byte[] yChars) [static]*

Returns the location of the first ASCII character that is not null or whitespace

*Parameters*

<i>yChars</i>	array of ASCII character bytes
---------------	--------------------------------

*Returns*

integer index of the first valid ASCII character

*void imrcp.system.dbf.DbfField.parseRecord (DataInputStream oDataInputStream) throws Exception*

Reads the field and determines the starting index and length of the data by stripping leading and trailing whitespace.

*Parameters*

<i>oDataInputStre am</i>	data file input stream
------------------------------	------------------------

*Exceptions*

<i>java.lang.Exce ption</i>
---------------------------------

Reimplemented in **imrcp.system.dbf.DbfDouble** (p.137), **imrcp.system.dbf.DbfLong** (p.142), and **imrcp.system.dbf.DbfString** (p.189).

*boolean imrcp.system.dbf.DbfField.wasNull ()*

Determines if the field data was null

*Returns*

true if null field data, false otherwise

---

**Member Data Documentation**

*int imrcp.system.dbf.DbfField.m\_nDataLength [protected]*

Data field length.

*int imrcp.system.dbf.DbfField.m\_nstartIndex [protected]*

Data field starting position.

*String imrcp.system.dbf.DbfField.m\_sName [protected]*

Data field name.

*byte [] imrcp.system.dbf.DbfField.m\_yBuffer [protected]*

Data field byte stream buffer.

*final int imrcp.system.dbf.DbfField.METADATA\_LENGTH = 32 [static]*

Maximum metadata field length.

---

*The documentation for this class was generated from the following file:*

- system/dbf/DbfField.java

---

**imrcp.system.dbf.DbfLong Class Reference**

**Public Member Functions**

- **DbfLong** (byte[] yChars, int nLength)
- void **parseRecord** (DataInputStream oDataInputStream) throws Exception
- int **getInt** ()
- long **getLong** ()
- float **getFloat** ()
- double **getDouble** ()
- String **toString** ()

## Additional Inherited Members

---

### Detailed Description

Holds data base file field natively as a long integer.

#### Author

Federal Highway Administration

#### Version

1.0

---

### Constructor & Destructor Documentation

`imrcp.system.dbf.DbfLong.DbfLong (byte[] yChars, int nLength)`

Creates a new instance of **DbfLong** with field name and data length defined.

#### Parameters

<code>yChars</code>	array of ASCII byte characters
<code>nLength</code>	length of byte character buffer

---

### Member Function Documentation

`double imrcp.system.dbf.DbfLong.getDouble ()`

Returns field data as a double precision number.

#### Returns

double precision number value of field data

Reimplemented from **imrcp.system.dbf.DbfField** (p.138).

`float imrcp.system.dbf.DbfLong.getFloat ()`

Returns field data as a floating point number.

#### Returns

floating point number value of field data

Reimplemented from **imrcp.system.dbf.DbfField** (p.139).

`int imrcp.system.dbf.DbfLong.getInt ()`

Returns field data as an integer.

#### Returns

integer value of field data

Reimplemented from **imrcp.system.dbf.DbfField** (p.139).

`long imrcp.system.dbf.DbfLong.getLong ()`

Returns field data as an long integer.

*Returns*

long integer value of field data

Reimplemented from **imrcp.system.dbf.DbfField** (p.139).

**void imrcp.system.dbf.DbfLong.parseRecord (DataInputStream oDataInputStream)**  
**throws Exception**

Reads the field and stores its contents as a long integer.

*Parameters*

<i>oDataInputStream</i>	data file input stream
-------------------------	------------------------

*Exceptions*

<i>java.lang.Exception</i>
----------------------------

Reimplemented from **imrcp.system.dbf.DbfField** (p.139).

**String imrcp.system.dbf.DbfLong.toString ()**

Returns field data as a string.

*Returns*

string value of field data

---

*The documentation for this class was generated from the following file:*

- system/dbf/DbfLong.java
- 

## imrcp.system.dbf.DbfResultSet Class Reference

### Public Member Functions

- **DbfResultSet** (InputStream oInputStream) throws Exception
- **DbfResultSet** (DataInputStream oIn) throws Exception
- **DbfResultSet** (String sFilename) throws Exception
- **String[] getColumnNames ()**
- **boolean next ()** throws SQLException
- **void close ()** throws SQLException
- **boolean wasNull ()** throws SQLException
- **String getString (int columnIndex)** throws SQLException
- **boolean getBoolean (int columnIndex)** throws SQLException
- **byte getByte (int columnIndex)** throws SQLException
- **short getShort (int columnIndex)** throws SQLException
- **int getInt (int columnIndex)** throws SQLException
- **long getLong (int columnIndex)** throws SQLException
- **float getFloat (int columnIndex)** throws SQLException
- **double getDouble (int columnIndex)** throws SQLException
- **BigDecimal getBigDecimal (int columnIndex, int scale)** throws SQLException
- **byte[] getBytes (int columnIndex)** throws SQLException
- **Date getDate (int columnIndex)** throws SQLException
- **Time getTime (int columnIndex)** throws SQLException

- `Timestamp getTimestamp (int columnIndex) throws SQLException`
- `InputStream getAsciiStream (int columnIndex) throws SQLException`
- `InputStream getUnicodeStream (int columnIndex) throws SQLException`
- `InputStream getBinaryStream (int columnIndex) throws SQLException`
- `String getString (String columnLabel) throws SQLException`
- `boolean getBoolean (String columnLabel) throws SQLException`
- `byte getByte (String columnLabel) throws SQLException`
- `short getShort (String columnLabel) throws SQLException`
- `int getInt (String columnLabel) throws SQLException`
- `long getLong (String columnLabel) throws SQLException`
- `float getFloat (String columnLabel) throws SQLException`
- `double getDouble (String columnLabel) throws SQLException`
- `BigDecimal getBigDecimal (String columnLabel, int scale) throws SQLException`
- `byte[] getBytes (String columnLabel) throws SQLException`
- `Date getDate (String columnLabel) throws SQLException`
- `Time getTime (String columnLabel) throws SQLException`
- `Timestamp getTimestamp (String columnLabel) throws SQLException`
- `InputStream getAsciiStream (String columnLabel) throws SQLException`
- `InputStream getUnicodeStream (String columnLabel) throws SQLException`
- `InputStream getBinaryStream (String columnLabel) throws SQLException`
- `SQLWarning getWarnings () throws SQLException`
- `void clearWarnings () throws SQLException`
- `String getCursorName () throws SQLException`
- `ResultSetMetaData getMetaData () throws SQLException`
- `Object getObject (int columnIndex) throws SQLException`
- `Object getObject (String columnLabel) throws SQLException`
- `int findColumn (String columnLabel) throws SQLException`
- `Reader getCharacterStream (int columnIndex) throws SQLException`
- `Reader getCharacterStream (String columnLabel) throws SQLException`
- `BigDecimal getBigDecimal (int columnIndex) throws SQLException`
- `BigDecimal getBigDecimal (String columnLabel) throws SQLException`
- `boolean isBeforeFirst () throws SQLException`
- `boolean isAfterLast () throws SQLException`
- `boolean isFirst () throws SQLException`
- `boolean isLast () throws SQLException`
- `void beforeFirst () throws SQLException`
- `void afterLast () throws SQLException`
- `boolean first () throws SQLException`
- `boolean last () throws SQLException`
- `int getRow () throws SQLException`
- `boolean absolute (int row) throws SQLException`
- `boolean relative (int rows) throws SQLException`
- `boolean previous () throws SQLException`
- `void setFetchDirection (int direction) throws SQLException`
- `int getFetchDirection () throws SQLException`
- `void setFetchSize (int rows) throws SQLException`
- `int getFetchSize () throws SQLException`
- `int getType () throws SQLException`
- `int getConcurrency () throws SQLException`
- `boolean rowUpdated () throws SQLException`
- `boolean rowInserted () throws SQLException`
- `boolean rowDeleted () throws SQLException`
- `void updateNull (int columnIndex) throws SQLException`
- `void updateBoolean (int columnIndex, boolean x) throws SQLException`
- `void updateByte (int columnIndex, byte x) throws SQLException`
- `void updateShort (int columnIndex, short x) throws SQLException`
- `void updateInt (int columnIndex, int x) throws SQLException`

- void **updateLong** (int columnIndex, long x) throws SQLException
- void **updateFloat** (int columnIndex, float x) throws SQLException
- void **updateDouble** (int columnIndex, double x) throws SQLException
- void **updateBigDecimal** (int columnIndex, BigDecimal x) throws SQLException
- void **updateString** (int columnIndex, String x) throws SQLException
- void **updateBytes** (int columnIndex, byte[] x) throws SQLException
- void **updateDate** (int columnIndex, Date x) throws SQLException
- void **updateTime** (int columnIndex, Time x) throws SQLException
- void **updateTimestamp** (int columnIndex, Timestamp x) throws SQLException
- void **updateAsciiStream** (int columnIndex, InputStream x, int length) throws SQLException
  
- void **updateBinaryStream** (int columnIndex, InputStream x, int length) throws SQLException
  
- void **updateCharacterStream** (int columnIndex, Reader x, int length) throws SQLException
  
- void **updateObject** (int columnIndex, Object x, int scaleOrLength) throws SQLException
- void **updateObject** (int columnIndex, Object x) throws SQLException
- void **updateNull** (String columnLabel) throws SQLException
- void **updateBoolean** (String columnLabel, boolean x) throws SQLException
- void **updateByte** (String columnLabel, byte x) throws SQLException
- void **updateShort** (String columnLabel, short x) throws SQLException
- void **updateInt** (String columnLabel, int x) throws SQLException
- void **updateLong** (String columnLabel, long x) throws SQLException
- void **updateFloat** (String columnLabel, float x) throws SQLException
- void **updateDouble** (String columnLabel, double x) throws SQLException
- void **updateBigDecimal** (String columnLabel, BigDecimal x) throws SQLException
- void **updateString** (String columnLabel, String x) throws SQLException
- void **updateBytes** (String columnLabel, byte[] x) throws SQLException
- void **updateDate** (String columnLabel, Date x) throws SQLException
- void **updateTime** (String columnLabel, Time x) throws SQLException
- void **updateTimestamp** (String columnLabel, Timestamp x) throws SQLException
- void **updateAsciiStream** (String columnLabel, InputStream x, int length) throws SQLException
  
- void **updateBinaryStream** (String columnLabel, InputStream x, int length) throws SQLException
- void **updateCharacterStream** (String columnLabel, Reader x, int length) throws SQLException
  
- void **updateObject** (String columnLabel, Object x, int scaleOrLength) throws SQLException
  
- void **updateObject** (String columnLabel, Object x) throws SQLException
- void **insertRow** () throws SQLException
- void **updateRow** () throws SQLException
- void **deleteRow** () throws SQLException
- void **refreshRow** () throws SQLException
- void **cancelRowUpdates** () throws SQLException
- void **moveToInsertRow** () throws SQLException
- void **moveToCurrentRow** () throws SQLException
- Statement **getStatement** () throws SQLException
- Object **getObject** (int columnIndex, Map< String, Class<?> > map) throws SQLException
- Ref **getRef** (int columnIndex) throws SQLException
- Blob **getBlob** (int columnIndex) throws SQLException
- Clob **getClob** (int columnIndex) throws SQLException
- Array **getArray** (int columnIndex) throws SQLException
- Object **getObject** (String columnLabel, Map< String, Class<?> > map) throws SQLException
  
- Ref **getRef** (String columnLabel) throws SQLException
- Blob **getBlob** (String columnLabel) throws SQLException

- **Clob** **getBlob** (String columnLabel) throws SQLException
- **Array** **getArray** (String columnLabel) throws SQLException
- **Date** **getDate** (int columnIndex, Calendar cal) throws SQLException
- **Date** **getDate** (String columnLabel, Calendar cal) throws SQLException
- **Time** **getTime** (int columnIndex, Calendar cal) throws SQLException
- **Time** **getTime** (String columnLabel, Calendar cal) throws SQLException
- **Timestamp** **getTimestamp** (int columnIndex, Calendar cal) throws SQLException
- **Timestamp** **getTimestamp** (String columnLabel, Calendar cal) throws SQLException
- **URL** **getURL** (int columnIndex) throws SQLException
- **URL** **getURL** (String columnLabel) throws SQLException
- void **updateRef** (int columnIndex, Ref x) throws SQLException
- void **updateRef** (String columnLabel, Ref x) throws SQLException
- void **updateBlob** (int columnIndex, Blob x) throws SQLException
- void **updateBlob** (String columnLabel, Blob x) throws SQLException
- void **updateClob** (int columnIndex, Clob x) throws SQLException
- void **updateClob** (String columnLabel, Clob x) throws SQLException
- void **updateArray** (int columnIndex, Array x) throws SQLException
- void **updateArray** (String columnLabel, Array x) throws SQLException
- RowId **getRowId** (int columnIndex) throws SQLException
- RowId **getRowId** (String columnLabel) throws SQLException
- void **updateRowId** (int columnIndex, RowId x) throws SQLException
- void **updateRowId** (String columnLabel, RowId x) throws SQLException
- int **getHoldability** () throws SQLException
- boolean **isClosed** () throws SQLException
- void **updateNString** (int columnIndex, String nString) throws SQLException
- void **updateNString** (String columnLabel, String nString) throws SQLException
- void **updateNClob** (int columnIndex, NClob nClob) throws SQLException
- void **updateNClob** (String columnLabel, NClob nClob) throws SQLException
- NClob **getNClob** (int columnIndex) throws SQLException
- NClob **getNClob** (String columnLabel) throws SQLException
- SQLXML **getSQLXML** (int columnIndex) throws SQLException
- SQLXML **getSQLXML** (String columnLabel) throws SQLException
- void **updateSQLXML** (int columnIndex, SQLXML xmlObject) throws SQLException
- void **updateSQLXML** (String columnLabel, SQLXML xmlObject) throws SQLException
- String **getNString** (int columnIndex) throws SQLException
- String **getNString** (String columnLabel) throws SQLException
- Reader **getNCharacterStream** (int columnIndex) throws SQLException
- Reader **getNCharacterStream** (String columnLabel) throws SQLException
- void **updateNCharacterStream** (int columnIndex, Reader x, long length) throws SQLException
  
- void **updateNCharacterStream** (String columnLabel, Reader reader, long length) throws SQLException
- void **updateAsciiStream** (int columnIndex, InputStream x, long length) throws SQLException
  
- void **updateBinaryStream** (int columnIndex, InputStream x, long length) throws SQLException
  
- void **updateCharacterStream** (int columnIndex, Reader x, long length) throws SQLException
  
- void **updateAsciiStream** (String columnLabel, InputStream x, long length) throws SQLException
- void **updateBinaryStream** (String columnLabel, InputStream x, long length) throws SQLException
- void **updateCharacterStream** (String columnLabel, Reader reader, long length) throws SQLException
- void **updateBlob** (int columnIndex, InputStream inputStream, long length) throws SQLException

- void **updateBlob** (String columnLabel, InputStream inputStream, long length) throws SQLException
  - void **updateClob** (int columnIndex, Reader reader, long length) throws SQLException
  - void **updateClob** (String columnLabel, Reader reader, long length) throws SQLException
  - void **updateNClob** (int columnIndex, Reader reader, long length) throws SQLException
  - void **updateNClob** (String columnLabel, Reader reader, long length) throws SQLException
  - void **updateNCharacterStream** (int columnIndex, Reader x) throws SQLException
  - void **updateNCharacterStream** (String columnLabel, Reader reader) throws SQLException
  
  - void **updateAsciiStream** (int columnIndex, InputStream x) throws SQLException
  - void **updateBinaryStream** (int columnIndex, InputStream x) throws SQLException
  - void **updateCharacterStream** (int columnIndex, Reader x) throws SQLException
  - void **updateAsciiStream** (String columnLabel, InputStream x) throws SQLException
  - void **updateBinaryStream** (String columnLabel, InputStream x) throws SQLException
  - void **updateCharacterStream** (String columnLabel, Reader reader) throws SQLException
  - void **updateBlob** (int columnIndex, InputStream inputStream) throws SQLException
  - void **updateBlob** (String columnLabel, InputStream inputStream) throws SQLException
  - void **updateClob** (int columnIndex, Reader reader) throws SQLException
  - void **updateClob** (String columnLabel, Reader reader) throws SQLException
  - void **updateNClob** (int columnIndex, Reader reader) throws SQLException
  - void **updateNClob** (String columnLabel, Reader reader) throws SQLException
  - boolean **isWrapperFor** (Class<?> iface) throws SQLException
- 

### Detailed Description

Holds database file (DBF) records in a SQL type result set.

#### Author

Federal Highway Administration

#### Version

1.0

---

### Constructor & Destructor Documentation

*imrcp.system.dbf.DbfResultSet.DbfResultSet (InputStream oInputStream) throws Exception*

#### Parameters

<i>oInputStream</i>
---------------------

#### Exceptions

<i>Exception</i>
------------------

*imrcp.system.dbf.DbfResultSet.DbfResultSet (String sFilename) throws Exception*

Creates a new instance of DbfResultsSet from the specified file.

#### Parameters

<i>sFilename</i>	absolute path file name of .dbf file to process
------------------	---

*Exceptions*

<i>java.lang.Exception</i>	
----------------------------	--

---

**Member Function Documentation**

***boolean imrcp.system.dbf.DbfResultSet.absolute (int row) throws SQLException***

(NOT IMPLEMENTED) Moves to the given row number in this **DbfResultSet** object.

If the row number is positive, the cursor moves to the given row number with respect to the beginning of the result set.

If the given row number is negative, the cursor moves to an absolute row position with respect to the end of the result set.

Note: Calling `absolute(1)` is the same as calling `first()`. Calling `absolute(-1)` is the same as calling `last()`.

*Parameters*

<i>row</i>	the row number which to move (1's based)
------------	--

*Returns*

true if moved to a position in this **DbfResultSet** object; false if before the first row or after the last row

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

***void imrcp.system.dbf.DbfResultSet.afterLast () throws SQLException***

(NOT IMPLEMENTED) Moves to the end of this **DbfResultSet** object, just after the last row.

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

***void imrcp.system.dbf.DbfResultSet.beforeFirst () throws SQLException***

(NOT IMPLEMENTED) Moves to the front of this **DbfResultSet** object, just before the first row.

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

***void imrcp.system.dbf.DbfResultSet.cancelRowUpdates () throws SQLException***

(NOT IMPLEMENTED) Cancels the updates made to the current row in this **DbfResultSet** object. This method may be called after calling an updater method(s) and before calling the

method updateRow to roll back the updates made to a row. If no updates have been made or updateRow has already been called, this method has no effect.

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

**void imrcp.system.dbf.DbfResultSet.clearWarnings () throws SQLException**

(NOT IMPLEMENTED) Clears all warnings reported on this DdbfResultSet object. After this method is called, the method getWarnings returns null until a new warning is reported for this **DbfResultSet** object.

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

**void imrcp.system.dbf.DbfResultSet.close () throws SQLException**

Close the database file stream.

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

**void imrcp.system.dbf.DbfResultSet.deleteRow () throws SQLException**

(NOT IMPLEMENTED) Deletes the current row from this **DbfResultSet** object and from the underlying database. This method cannot be called when on the insert row.

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

**int imrcp.system.dbf.DbfResultSet.findColumn (String columnLabel) throws SQLException**

Returns the data field index associated with the specified label.

*Parameters*

<i>columnLabel</i>	data field label
--------------------	------------------

*Returns*

integer index of the data filed (1's based)

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

**boolean imrcp.system.dbf.DbfResultSet.first () throws SQLException**

(NOT IMPLEMENTED) Moves to the first row in this **DbfResultSet** object.

*Returns*

true if on a valid row; false if there are no rows in the result set

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*Array imrcp.system.dbf.DbResultSet.getArray (int columnIndex) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as an Array object.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
--------------------	------------------------------

*Returns*

Array representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*Array imrcp.system.dbf.DbResultSet.getArray (String columnLabel) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as a Array object.

*Parameters*

<i>columnLabel</i>	data field label
--------------------	------------------

*Returns*

Array representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*InputStream imrcp.system.dbf.DbResultSet.getAsciiStream (int columnIndex) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as an ASCII InputStream object.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
--------------------	------------------------------

*Returns*

ASCII InputStream representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*InputStream imrcp.system.dbf.DbResultSet.getAsciiStream (String columnLabel) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as an ASCII InputStream object.

*Parameters*

<i>columnLabel</i>	data field label
--------------------	------------------

*Returns*

ASCII InputStream representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*BigDecimal imrcp.system.dbf.DbResultSet.getBigDecimal (int columnIndex) throws SQLException*

Returns the specified data field as a big decimal.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
--------------------	------------------------------

*Returns*

big decimal representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*BigDecimal imrcp.system.dbf.DbResultSet.getBigDecimal (int columnIndex, int scale) throws SQLException*

Returns the specified data field as a big decimal number.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
<i>scale</i>	scale of the big decimal value to be returned

*Returns*

big decimal number representation of field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*BigDecimal imrcp.system.dbf.DbResultSet.getBigDecimal (String columnLabel) throws SQLException*

Returns the specified data field as a big decimal.

*Parameters*

<i>columnLabel</i>	data field label
--------------------	------------------

*Returns*

big decimal representation of the field data

*Exceptions*

<code>java.sql.SQLException</code>	
------------------------------------	--

*BigDecimal imrcp.system.dbf.DbResultSet.getBigDecimal (String columnLabel, int scale) throws SQLException*

Returns the specified data field as a big decimal number.

*Parameters*

<code>columnLabel</code>	data field label
--------------------------	------------------

*Returns*

big decimal number representation of the field data

*Exceptions*

<code>java.sql.SQLException</code>	
------------------------------------	--

*InputStream imrcp.system.dbf.DbResultSet.getBinaryStream (int columnIndex) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as a binary InputStream object.

*Parameters*

<code>columnIndex</code>	data field index (1's based)
--------------------------	------------------------------

*Returns*

binary InputStream representation of the field data

*Exceptions*

<code>java.sql.SQLException</code>	
------------------------------------	--

*InputStream imrcp.system.dbf.DbResultSet.getBinaryStream (String columnLabel) throws SQLException*

Returns the specified data field as a binary InputStream object.

*Parameters*

<code>columnLabel</code>	data field label
--------------------------	------------------

*Returns*

binary InputStream representation of the field data

*Exceptions*

<code>java.sql.SQLException</code>	
------------------------------------	--

*Blob imrcp.system.dbf.DbResultSet.getBlob (int columnIndex) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as a Blob object.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
--------------------	------------------------------

*Returns*

Blob representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*Blob imrcp.system.dbf.DbResultSet.getBlob (String columnLabel) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as a Blob object.

*Parameters*

<i>columnLabel</i>	data field label
--------------------	------------------

*Returns*

Blob representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*boolean imrcp.system.dbf.DbResultSet.getBoolean (int columnIndex) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as a boolean.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
--------------------	------------------------------

*Returns*

boolean representation of field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*boolean imrcp.system.dbf.DbResultSet.getBoolean (String columnLabel) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as a boolean.

*Parameters*

<i>columnLabel</i>	data field label
--------------------	------------------

*Returns*

boolean representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*byte imrcp.system.dbf.DbResultSet.getByte (int columnIndex) throws SQLException*

Returns the specified data field as a byte integer.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
--------------------	------------------------------

*Returns*

byte integer representation of field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*byte imrcp.system.dbf.DbResultSet.getByte (String columnLabel) throws SQLException*

Returns the specified data field as a byte.

*Parameters*

<i>columnLabel</i>	data field label
--------------------	------------------

*Returns*

byte representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*byte[] imrcp.system.dbf.DbResultSet.getBytes (int columnIndex) throws SQLException*

Returns the specified data field as a byte array.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
--------------------	------------------------------

*Returns*

byte array representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*byte[] imrcp.system.dbf.DbResultSet.getBytes (String columnLabel) throws SQLException*

Returns the specified data field as a byte array.

*Parameters*

<i>columnLabel</i>	data field label
--------------------	------------------

*Returns*

byte array representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*Reader imrcp.system.dbf.DbfResultSet.getCharacterStream (int columnIndex) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as a character stream Reader.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
--------------------	------------------------------

*Returns*

character stream Reader representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*Reader imrcp.system.dbf.DbfResultSet.getCharacterStream (String columnLabel) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as a character stream Reader.

*Parameters*

<i>columnLabel</i>	data field label
--------------------	------------------

*Returns*

character stream Reader representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*Clob imrcp.system.dbf.DbfResultSet.getClob (int columnIndex) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as a Clob object.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
--------------------	------------------------------

*Returns*

Clob representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*Clob imrcp.system.dbf.Db.ResultSet.getClob (String columnLabel) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as a Clob object.

*Parameters*

<i>columnLabel</i>	data field label
--------------------	------------------

*Returns*

Clob representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*int imrcp.system.dbf.Db.ResultSet.getConcurrency () throws SQLException*

Retrieves the concurrency mode of this **DbfResultSet** object.

*Returns*

integer specifying the concurrency type, either ResultSet.CONCUR\_READ\_ONLY or ResultSet.CONCUR\_UPDATABLE

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*String imrcp.system.dbf.Db.ResultSet.getCursorName () throws SQLException*

(NOT IMPLEMENTED) Returns the name of the cursor used by this **DbfResultSet** object.

*Returns*

the name for this **DbfResultSet** object's cursor

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*Date imrcp.system.dbf.Db.ResultSet.getDate (int columnIndex) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as a Date object.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
--------------------	------------------------------

*Returns*

Date representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>
------------------------------

*Date imrcp.system.dbf.DbfResultSet.getDate (int columnIndex, Calendar cal) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as a Date object. This method uses the given calendar to construct an appropriate millisecond value for the date if the underlying database does not store timezone information.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
<i>cal</i>	the Calendar object to use in constructing the date

*Returns*

Date representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>
------------------------------

*Date imrcp.system.dbf.DbfResultSet.getDate (String columnLabel) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as a Date object.

*Parameters*

<i>columnLabel</i>	data field label
--------------------	------------------

*Returns*

Date representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>
------------------------------

*Date imrcp.system.dbf.DbfResultSet.getDate (String columnLabel, Calendar cal) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as a Date object. This method uses the given calendar to construct an appropriate millisecond value for the date if the underlying database does not store timezone information.

*Parameters*

<i>columnLabel</i>	data field label
<i>cal</i>	the Calendar object to use in constructing the date

*Returns*

Date representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*double imrcp.system.dbf.DbfResultSet.getDouble (int columnIndex) throws SQLException*

Returns the specified data field as a double precision number.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
--------------------	------------------------------

*Returns*

double precision number representation of field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*double imrcp.system.dbf.DbfResultSet.getDouble (String columnLabel) throws SQLException*

Returns the specified data field as a double precision number.

*Parameters*

<i>columnLabel</i>	data field label
--------------------	------------------

*Returns*

double precision number representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*int imrcp.system.dbf.DbfResultSet.getFetchDirection () throws SQLException*

Retrieves the fetch direction for this ResultSet object.

*Returns*

the current fetch direction for this **DbfResultSet** object; one of  
ResultSet.FETCH\_FORWARD, ResultSet.FETCH\_REVERSE, or  
ResultSet.FETCH\_UNKNOWN

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*int imrcp.system.dbf.DbfResultSet.getFetchSize () throws SQLException*

Retrieves the fetch size for this ResultSet object.

*Returns*

integer fetch size for this ResultSet object

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*float imrcp.system.dbf.DbResultSet.getFloat (int columnIndex) throws SQLException*

Returns the specified data field as a floating point number.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
--------------------	------------------------------

*Returns*

floating point number representation of field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*float imrcp.system.dbf.DbResultSet.getFloat (String columnLabel) throws SQLException*

Returns the specified data field as a floating point number.

*Parameters*

<i>columnLabel</i>	data field label
--------------------	------------------

*Returns*

floating point number representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*int imrcp.system.dbf.DbResultSet.getHoldability () throws SQLException*

(NOT IMPLEMENTED)

*int imrcp.system.dbf.DbResultSet.getInt (int columnIndex) throws SQLException*

Returns the specified data field as an integer.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
--------------------	------------------------------

*Returns*

integer representation of field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*int imrcp.system.dbf.DbResultSet.getInt (String columnLabel) throws SQLException*

Returns the specified data field as an integer.

*Parameters*

<i>columnLabel</i>	data field label
--------------------	------------------

*Returns*

integer representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*long imrcp.system.dbf.DbResultSet.getLong (int columnIndex) throws SQLException*

Returns the specified data field as a long integer.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
--------------------	------------------------------

*Returns*

long integer representation of field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*long imrcp.system.dbf.DbResultSet.getLong (String columnLabel) throws SQLException*

Returns the specified data field as a long integer.

*Parameters*

<i>columnLabel</i>	data field label
--------------------	------------------

*Returns*

long integer representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*ResultSetMetaData imrcp.system.dbf.DbResultSet.getMetaData () throws SQLException*

(NOT IMPLEMENTED) Returns the number, types and properties of this **DbfResultSet** object's columns.

*Returns*

the description of this **DbfResultSet** object's columns

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*Reader imrcp.system.dbf.Db.ResultSet.getNCharacterStream (int columnIndex)*  
*throws SQLException*

(NOT IMPLEMENTED)

*Reader imrcp.system.dbf.Db.ResultSet.getNCharacterStream (String columnLabel)*  
*throws SQLException*

(NOT IMPLEMENTED)

*NBlob imrcp.system.dbf.Db.ResultSet.getNBlob (int columnIndex) throws*  
*SQLException*

(NOT IMPLEMENTED)

*NBlob imrcp.system.dbf.Db.ResultSet.getNBlob (String columnLabel) throws*  
*SQLException*

(NOT IMPLEMENTED)

*String imrcp.system.dbf.Db.ResultSet.getString (int columnIndex) throws*  
*SQLException*

Returns the specified data field as a String.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
--------------------	------------------------------

*Returns*

String representation of field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*String imrcp.system.dbf.Db.ResultSet.getString (String columnLabel) throws*  
*SQLException*

Returns the specified data field as a String.

*Parameters*

<i>columnLabel</i>	data field label
--------------------	------------------

*Returns*

String representation of field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*Object imrcp.system.dbf.Db.ResultSet.getObject (int columnIndex) throws*  
*SQLException*

Returns the specified data field as an object.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
--------------------	------------------------------

*Returns*

object **DbfField** object

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*Object imrcp.system.dbf.DbfResultSet.getObject (int columnIndex, Map< String, Class<?>> map) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as an object. This method uses the given Map object for the custom mapping of the SQL structured or distinct type that is being retrieved.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
<i>map</i>	a Map object that contains the mapping from SQL type names to classes

*Returns*

object **DbfField** object

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*Object imrcp.system.dbf.DbfResultSet.getObject (String columnLabel) throws SQLException*

Returns the specified data field as an object.

*Parameters*

<i>columnLabel</i>	data field label
--------------------	------------------

*Returns*

object **DbfField** object

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*Object imrcp.system.dbf.DbfResultSet.getObject (String columnLabel, Map< String, Class<?>> map) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as an object. This method uses the given Map object for the custom mapping of the SQL structured or distinct type that is being retrieved.

*Parameters*

<i>columnLabel</i>	data field label
<i>map</i>	a Map object that contains the mapping from SQL type names to classes

*Returns*

object **DbfField** object

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*Ref imrcp.system.dbf.DbfResultSet.getRef (int columnIndex) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as Ref object.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
--------------------	------------------------------

*Returns*

reference object to SQL REF value

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*Ref imrcp.system.dbf.DbfResultSet.getRef (String columnLabel) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as an Ref object.

*Parameters*

<i>columnLabel</i>	data field label
--------------------	------------------

*Returns*

reference object to SQL REF value

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*int imrcp.system.dbf.DbfResultSet.getRow () throws SQLException*

Returns the current row number.

*Returns*

integer row number (1's based)

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*RowId imrcp.system.dbf.DbResultSet.getRowId (int columnIndex) throws SQLException*

(NOT IMPLEMENTED)

*RowId imrcp.system.dbf.DbResultSet.getRowId (String columnLabel) throws SQLException*

(NOT IMPLEMENTED)

*short imrcp.system.dbf.DbResultSet.getShort (int columnIndex) throws SQLException*

Returns the specified data field as a short integer.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
--------------------	------------------------------

*Returns*

short integer representation of field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*short imrcp.system.dbf.DbResultSet.getShort (String columnLabel) throws SQLException*

Returns the specified data field as a short integer.

*Parameters*

<i>columnLabel</i>	data field label
--------------------	------------------

*Returns*

short integer representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*SQLXML imrcp.system.dbf.DbResultSet.getSQLXML (int columnIndex) throws SQLException*

(NOT IMPLEMENTED)

*SQLXML imrcp.system.dbf.DbResultSet.getSQLXML (String columnLabel) throws SQLException*

(NOT IMPLEMENTED)

*Statement imrcp.system.dbf.DbResultSet.getStatement () throws SQLException*

(NOT IMPLEMENTED) Returns the Statement object that produced this ResultSet object. Since the result set is not produced from a SQL query, the statement is null.

*Returns*

the Statement object that produced this **DbfResultSet** object or null if the result set was produced some other way

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*String imrcp.system.dbf.DbResultSet.getString (int columnIndex) throws SQLException*

Returns the specified data field as a String.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
--------------------	------------------------------

*Returns*

String representation of field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*String imrcp.system.dbf.DbResultSet.getString (String columnLabel) throws SQLException*

Returns the specified data field as a String.

*Parameters*

<i>columnLabel</i>	data field label
--------------------	------------------

*Returns*

String representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*Time imrcp.system.dbf.DbResultSet.getTime (int columnIndex) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as a Time object.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
--------------------	------------------------------

*Returns*

Time representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*Time imrcp.system.dbf.DbResultSet.getTime (int columnIndex, Calendar cal) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as a Time object. This method uses the given calendar to construct an appropriate millisecond value for the time if the underlying database does not store timezone information.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
<i>cal</i>	the Calendar object to use in constructing the time

*Returns*

Time representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>

*Time imrcp.system.dbf.DbResultSet.getTime (String columnLabel) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as a Time object.

*Parameters*

<i>columnLabel</i>	data field label
--------------------	------------------

*Returns*

Time representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>

*Time imrcp.system.dbf.DbResultSet.getTime (String columnLabel, Calendar cal) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as a Time object. This method uses the given calendar to construct an appropriate millisecond value for the time if the underlying database does not store timezone information.

*Parameters*

<i>columnLabel</i>	data field label
<i>cal</i>	the Calendar object to use in constructing the time

*Returns*

Time representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>

*Timestamp imrcp.system.dbf.DbResultSet.getTimestamp (int columnIndex) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as a Timestamp object.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
--------------------	------------------------------

*Returns*

Timestamp representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*Timestamp imrcp.system.dbf.DbResultSet.getTimestamp (int columnIndex, Calendar cal) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as a Timestamp object. This method uses the given calendar to construct an appropriate millisecond value for the time if the underlying database does not store timezone information.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
<i>cal</i>	the Calendar object to use in constructing the time

*Returns*

Timestamp representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*Timestamp imrcp.system.dbf.DbResultSet.getTimestamp (String columnLabel) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as a Timestamp object.

*Parameters*

<i>columnLabel</i>	data field label
--------------------	------------------

*Returns*

Timestamp representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*Timestamp imrcp.system.dbf.DbfResultSet.getTimestamp (String columnLabel, Calendar cal) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as a Timestamp object. This method uses the given calendar to construct an appropriate millisecond value for the time if the underlying database does not store timezone information.

*Parameters*

<i>columnLabel</i>	data field label
<i>cal</i>	the Calendar object to use in constructing the time

*Returns*

Timestamp representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*int imrcp.system.dbf.DbfResultSet.getType () throws SQLException*

Returns the type of this **DbfResultSet** object.

*Returns*

integer specifying the **DbfResultSet** type; one of ResultSet.TYPE\_FORWARD\_ONLY, ResultSet.TYPE\_SCROLL\_INSENSITIVE, or ResultSet.TYPE\_SCROLL\_SENSITIVE

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*InputStream imrcp.system.dbf.DbfResultSet.getUnicodeStream (int columnIndex) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as an Unicode InputStream object.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
--------------------	------------------------------

*Returns*

Unicode InputStream representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*InputStream imrcp.system.dbf.DbfResultSet.getUnicodeStream (String columnLabel) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as an Unicode InputStream object.

*Parameters*

<i>columnLabel</i>	data field label
--------------------	------------------

*Returns*

Unicode InputStream representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*URL imrcp.system.dbf.DbfResultSet.getURL (int columnIndex) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as a URL object.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
--------------------	------------------------------

*Returns*

URL representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*URL imrcp.system.dbf.DbfResultSet.getURL (String columnLabel) throws SQLException*

(NOT IMPLEMENTED) Returns the specified data field as a URL object.

*Parameters*

<i>columnLabel</i>	data field label
--------------------	------------------

*Returns*

URL representation of the field data

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*SQLWarning imrcp.system.dbf.DbfResultSet.getWarnings () throws SQLException*

(NOT IMPLEMENTED) Returns the first warning reported by calls on this **DbfResultSet** object. Subsequent warnings on this ResultSet object will be chained to the SQLWarning object that this method returns.

*Returns*

the first SQLWarning object reported or null if there are none

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbfResultSet.insertRow () throws SQLException*

(NOT IMPLEMENTED) Inserts the contents of the insert row into this **DbfResultSet** object and into the database. Must be on the insert row when this method is called.

*Exceptions*

<i>java.sql.SQLException</i>	
<i>ception</i>	

*boolean imrcp.system.dbf.DbfResultSet.isAfterLast () throws SQLException*

Determines if after the last row in this **DbfResultSet** object.

*Returns*

true if after the last row; false if at any other position or the result set contains no rows

*Exceptions*

<i>java.sql.SQLException</i>	
<i>ception</i>	

*boolean imrcp.system.dbf.DbfResultSet.isBeforeFirst () throws SQLException*

Determines if before the first row in this **DbfResultSet** object.

*Returns*

true if before the first row; false if at any other position or the result set contains no rows

*Exceptions*

<i>java.sql.SQLException</i>	
<i>ception</i>	

*boolean imrcp.system.dbf.DbfResultSet.isClosed () throws SQLException*

(NOT IMPLEMENTED)

*boolean imrcp.system.dbf.DbfResultSet.isFirst () throws SQLException*

Determines if on the first row of this **DbfResultSet** object.

*Returns*

true if on the first row; false otherwise

*Exceptions*

<i>java.sql.SQLException</i>	
<i>ception</i>	

*boolean imrcp.system.dbf.DbfResultSet.isLast () throws SQLException*

Determines if on the last row of this **DbfResultSet** object.

*Returns*

true if on the last row; false otherwise

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*boolean imrcp.system.dbf.DbfResultSet.isWrapperFor (Class<?> iface) throws SQLException*

(NOT IMPLEMENTED)

*boolean imrcp.system.dbf.DbfResultSet.last () throws SQLException*

(NOT IMPLEMENTED) Moves to the last row in this **DbfResultSet** object.

*Returns*

true if on a valid row; false if there are no rows in the result set

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbfResultSet.moveToCurrentRow () throws SQLException*

(NOT IMPLEMENTED) Moves to the remembered position, usually the current row. This method has no effect if not on the insert row.

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbfResultSet.moveToInsertRow () throws SQLException*

(NOT IMPLEMENTED) Moves to the insert row. The current position is remembered while positioned on the insert row.

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*boolean imrcp.system.dbf.DbfResultSet.next () throws SQLException*

Move to the next row in the results set.

*Returns*

true if there are more records remaining

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*boolean imrcp.system.dbf.DbfResultSet.previous () throws SQLException*

(NOT IMPLEMENTED) Move to the previous row in this **DbfResultsSet** object.

*Returns*

true if positioned on a valid row; false if positioned before the first row

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbResultSet.refreshRow () throws SQLException*

(NOT IMPLEMENTED) Refreshes the current row with its most recent value in the database. This method cannot be called when on the insert row.

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*boolean imrcp.system.dbf.DbResultSet.relative (int rows) throws SQLException*

(NOT IMPLEMENTED) Moves a relative number of rows, either positive or negative.

Note: Calling the method **relative(1)** is identical to calling the method **next()** and calling the method **relative(-1)** is identical to calling the method **previous()**.

*Parameters*

<i>rows</i>	number of rows to move
-------------	------------------------

*Returns*

true if on a row; false otherwise

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*boolean imrcp.system.dbf.DbResultSet.rowDeleted () throws SQLException*

(NOT IMPLEMENTED) Determines if the current row has been deleted.

*Returns*

true if the current row has been deleted; false otherwise

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*boolean imrcp.system.dbf.DbResultSet.rowInserted () throws SQLException*

(NOT IMPLEMENTED) Determines if the current row has been inserted.

*Returns*

true if the current row has been inserted; false otherwise

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*boolean imrcp.system.dbf.DbResultSet.rowUpdated () throws SQLException*

(NOT IMPLEMENTED) Determines if the current row has been updated.

*Returns*

true if the current row has been updated; false otherwise

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbfResultSet.setFetchDirection (int direction) throws SQLException*

(NOT IMPLEMENTED) Gives a hint as to the direction in which the rows in this **DbfResultSet** object will be processed. The fetch direction may be changed at any time.

*Parameters*

<i>direction</i>	integer specifying the suggested fetch direction; one of ResultSet.FETCH_FORWARD, ResultSet.FETCH_REVERSE, or ResultSet.FETCH_UNKNOWN
------------------	---

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbfResultSet.setFetchSize (int rows) throws SQLException*

(NOT IMPLEMENTED) Defines the number of rows that should be fetched when more rows are needed for the DbfResultSet object.

*Parameters*

<i>rows</i>	the number of rows to fetch
-------------	-----------------------------

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbfResultSet.updateArray (int columnIndex, Array x) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbfResultSet.updateArray (String columnLabel, Array x) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbfResultSet.updateAsciiStream (int columnIndex, InputStream x) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbfResultSet.updateAsciiStream (int columnIndex, InputStream x, int length) throws SQLException*

(NOT IMPLEMENTED) Updates the specified column with an ASCII InputStream value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
<i>x</i>	ASCII InputStream field value
<i>length</i>	integer length of the data in the stream

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbResultSet.updateAsciiStream (int columnIndex, InputStream x, long length) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbResultSet.updateAsciiStream (String columnLabel, InputStream x) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbResultSet.updateAsciiStream (String columnLabel, InputStream x, int length) throws SQLException*

(NOT IMPLEMENTED) Updates the specified column with an ASCII InputStream value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnLabel</i>	data field label
<i>x</i>	ASCII InputStream field value
<i>length</i>	integer length of the data in the stream

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbResultSet.updateAsciiStream (String columnLabel, InputStream x, long length) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbResultSet.updateBigDecimal (int columnIndex, BigDecimal x) throws SQLException*

(NOT IMPLEMENTED) Updates the specified column with a BigDecimal value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
<i>x</i>	BigDecimal field value

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbResultSet.updateBigDecimal (String columnLabel, BigDecimal x) throws SQLException*

(NOT IMPLEMENTED) Updates the specified column with a BigDecimal value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnLabel</i>	data field label
<i>x</i>	BigDecimal field value

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbResultSet.updateBinaryStream (int columnIndex, InputStream x) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbResultSet.updateBinaryStream (int columnIndex, InputStream x, int length) throws SQLException*

(NOT IMPLEMENTED) Updates the specified column with a binary InputStream value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
<i>x</i>	binary InputStream field value
<i>length</i>	integer length of the data in the stream

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbResultSet.updateBinaryStream (int columnIndex, InputStream x, long length) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbResultSet.updateBinaryStream (String columnLabel, InputStream x) throws SQLException*

(NOT IMPLEMENTED)

`void imrcp.system.dbf.DbfResultSet.updateBinaryStream (String columnLabel, InputStream x, int length) throws SQLException`

(NOT IMPLEMENTED) Updates the specified column with a binary InputStream value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

**Parameters**

<i>columnLabel</i>	data field label
<i>x</i>	binary InputStream field value
<i>length</i>	integer length of the data in the stream

**Exceptions**

<code>java.sql.SQLException</code>	
------------------------------------	--

`void imrcp.system.dbf.DbfResultSet.updateBinaryStream (String columnLabel, InputStream x, long length) throws SQLException`

(NOT IMPLEMENTED)

`void imrcp.system.dbf.DbfResultSet.updateBlob (int columnIndex, Blob x) throws SQLException`

(NOT IMPLEMENTED)

`void imrcp.system.dbf.DbfResultSet.updateBlob (int columnIndex, InputStream inputStream) throws SQLException`

(NOT IMPLEMENTED)

`void imrcp.system.dbf.DbfResultSet.updateBlob (int columnIndex, InputStream inputStream, long length) throws SQLException`

(NOT IMPLEMENTED)

`void imrcp.system.dbf.DbfResultSet.updateBlob (String columnLabel, Blob x) throws SQLException`

(NOT IMPLEMENTED)

`void imrcp.system.dbf.DbfResultSet.updateBlob (String columnLabel, InputStream inputStream) throws SQLException`

(NOT IMPLEMENTED)

`void imrcp.system.dbf.DbfResultSet.updateBlob (String columnLabel, InputStream inputStream, long length) throws SQLException`

(NOT IMPLEMENTED)

`void imrcp.system.dbf.DbfResultSet.updateBoolean (int columnIndex, boolean x) throws SQLException`

(NOT IMPLEMENTED) Updates the specified column with a boolean value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
<i>x</i>	boolean field value

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbfResultSet.updateBoolean (String columnLabel, boolean x) throws SQLException*

(NOT IMPLEMENTED) Updates the specified column with a boolean value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnLabel</i>	data field label
<i>x</i>	boolean field value

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbfResultSet.updateByte (int columnIndex, byte x) throws SQLException*

(NOT IMPLEMENTED) Updates the specified column with a bbyte value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
<i>x</i>	byte field value

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbfResultSet.updateByte (String columnLabel, byte x) throws SQLException*

(NOT IMPLEMENTED) Updates the specified column with a byte value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnLabel</i>	data field label
<i>x</i>	byte field value

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbResultSet.updateBytes (int columnIndex, byte[] x) throws SQLException*

(NOT IMPLEMENTED) Updates the specified column with a byte array value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
<i>x</i>	byte array field value

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbResultSet.updateBytes (String columnLabel, byte[] x) throws SQLException*

(NOT IMPLEMENTED) Updates the specified column with a byte array value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnLabel</i>	data field label
<i>x</i>	byte array field value

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbResultSet.updateCharacterStream (int columnIndex, Reader x) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbResultSet.updateCharacterStream (int columnIndex, Reader x, int length) throws SQLException*

(NOT IMPLEMENTED) Updates the specified column with a character stream Reader value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
<i>x</i>	character stream Reader field value
<i>length</i>	integer length of the data in the reader

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbResultSet.updateCharacterStream (int columnIndex, Reader x, long length) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbResultSet.updateCharacterStream (String columnLabel, Reader reader) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbResultSet.updateCharacterStream (String columnLabel, Reader reader, long length) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbResultSet.updateCharacterStream (String columnLabel, Reader x, int length) throws SQLException*

(NOT IMPLEMENTED) Updates the specified column with a character stream Reader value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnLabel</i>	data field label
<i>x</i>	character stream Reader field value
<i>length</i>	integer length of the data in the reader

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbResultSet.updateClob (int columnIndex, Clob x) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbResultSet.updateClob (int columnIndex, Reader reader) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbResultSet.updateClob (int columnIndex, Reader reader, long length) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbResultSet.updateClob (String columnLabel, Clob x) throws SQLException*

(NOT IMPLEMENTED)

```
void imrcp.system.dbf.DbfResultSet.updateClob (String columnLabel, Reader reader)
throws SQLException
```

(NOT IMPLEMENTED)

```
void imrcp.system.dbf.DbfResultSet.updateClob (String columnLabel, Reader reader,
long length) throws SQLException
```

(NOT IMPLEMENTED)

```
void imrcp.system.dbf.DbfResultSet.updateDate (int columnIndex, Date x) throws
SQLException
```

(NOT IMPLEMENTED) Updates the specified column with a Date value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

#### Parameters

<i>columnIndex</i>	data field index (1's based)
<i>x</i>	Date field value

#### Exceptions

<i>java.sql.SQLException</i>	
------------------------------	--

```
void imrcp.system.dbf.DbfResultSet.updateDate (String columnLabel, Date x) throws
SQLException
```

(NOT IMPLEMENTED) Updates the specified column with a Date value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

#### Parameters

<i>columnLabel</i>	data field label
<i>x</i>	Date field value

#### Exceptions

<i>java.sql.SQLException</i>	
------------------------------	--

```
void imrcp.system.dbf.DbfResultSet.updateDouble (int columnIndex, double x)
throws SQLException
```

(NOT IMPLEMENTED) Updates the specified column with a double precision number value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

#### Parameters

<i>columnIndex</i>	data field index (1's based)
<i>x</i>	double precision number field value

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

***void imrcp.system.dbf.DbResultSet.updateDouble (String columnLabel, double x) throws SQLException***

(NOT IMPLEMENTED) Updates the specified column with a double precision number value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnLabel</i>	data field label
<i>x</i>	double precision number field value

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

***void imrcp.system.dbf.DbResultSet.updateFloat (int columnIndex, float x) throws SQLException***

(NOT IMPLEMENTED) Updates the specified column with a floating point number value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
<i>x</i>	floating point number field value

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

***void imrcp.system.dbf.DbResultSet.updateFloat (String columnLabel, float x) throws SQLException***

(NOT IMPLEMENTED) Updates the specified column with a floating point number value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnLabel</i>	data field label
<i>x</i>	floating point number field value

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

`void imrcp.system.dbf.DbfResultSet.updateInt (int columnIndex, int x) throws SQLException`

(NOT IMPLEMENTED) Updates the specified column with an integer value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<code>columnIndex</code>	data field index (1's based)
<code>x</code>	integer field value

*Exceptions*

<code>java.sql.SQLException</code>	
------------------------------------	--

`void imrcp.system.dbf.DbfResultSet.updateInt (String columnLabel, int x) throws SQLException`

(NOT IMPLEMENTED) Updates the specified column with an integer value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<code>columnLabel</code>	data field label
<code>x</code>	integer field value

*Exceptions*

<code>java.sql.SQLException</code>	
------------------------------------	--

`void imrcp.system.dbf.DbfResultSet.updateLong (int columnIndex, long x) throws SQLException`

(NOT IMPLEMENTED) Updates the specified column with a long integer value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<code>columnIndex</code>	data field index (1's based)
<code>x</code>	long integer field value

*Exceptions*

<code>java.sql.SQLException</code>	
------------------------------------	--

`void imrcp.system.dbf.DbfResultSet.updateLong (String columnLabel, long x) throws SQLException`

(NOT IMPLEMENTED) Updates the specified column with a long integer value. The updater methods are used to update column values in the current row or the insert row. The

update methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnLabel</i>	data field label
<i>x</i>	long integer field value

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbResultSet.updateNCharacterStream (int columnIndex, Reader x) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbResultSet.updateNCharacterStream (int columnIndex, Reader x, long length) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbResultSet.updateNCharacterStream (String columnLabel, Reader reader) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbResultSet.updateNCharacterStream (String columnLabel, Reader reader, long length) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbResultSet.updateNClob (int columnIndex, NClob nClob) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbResultSet.updateNClob (int columnIndex, Reader reader) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbResultSet.updateNClob (int columnIndex, Reader reader, long length) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbResultSet.updateNClob (String columnLabel, NClob nClob) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbResultSet.updateNClob (String columnLabel, Reader reader) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbResultSet.updateNClob (String columnLabel, Reader reader, long length) throws SQLException*

(NOT IMPLEMENTED)

`void imrcp.system.dbf.DbfResultSet.updateNString (int columnIndex, String nString)  
throws SQLException`

(NOT IMPLEMENTED)

`void imrcp.system.dbf.DbfResultSet.updateNString (String columnLabel, String  
nString) throws SQLException`

(NOT IMPLEMENTED)

`void imrcp.system.dbf.DbfResultSet.updateNull (int columnIndex) throws  
SQLException`

(NOT IMPLEMENTED) Updates the specified column with a null value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

**Parameters**

<code>columnIndex</code>	data field index (1's based)
--------------------------	------------------------------

**Exceptions**

<code>java.sql.SQLException</code>	
------------------------------------	--

`void imrcp.system.dbf.DbfResultSet.updateNull (String columnLabel) throws  
SQLException`

(NOT IMPLEMENTED) Updates the specified column with a null value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

**Parameters**

<code>columnLabel</code>	data field label
--------------------------	------------------

**Exceptions**

<code>java.sql.SQLException</code>	
------------------------------------	--

`void imrcp.system.dbf.DbfResultSet.updateObject (int columnIndex, Object x) throws  
SQLException`

(NOT IMPLEMENTED) Updates the specified column with an Object value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

**Parameters**

<code>columnIndex</code>	data field index (1's based)
<code>x</code>	Object field value

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbResultSet.updateObject (int columnIndex, Object x, int scaleOrLength) throws SQLException*

(NOT IMPLEMENTED) Updates the specified column with an Object value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
<i>x</i>	Object field value
<i>scaleOrLength</i>	integer for an object of java.math.BigDecimal , this is the number of digits after the decimal point. For Java Object types InputStream and Reader, this is the length of the data in the stream or reader. For all other types, this value will be ignored.

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbResultSet.updateObject (String columnLabel, Object x) throws SQLException*

(NOT IMPLEMENTED) Updates the specified column with an object value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnLabel</i>	data field label
<i>x</i>	Object field value

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbResultSet.updateObject (String columnLabel, Object x, int scaleOrLength) throws SQLException*

(NOT IMPLEMENTED) Updates the specified column with an Object value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnLabel</i>	data field label
<i>x</i>	Object field value

<i>scaleOrLength</i>	integer for an object of java.math.BigDecimal , this is the number of digits after the decimal point. For Java Object types InputStream and Reader, this is the length of the data in the stream or reader. For all other types, this value will be ignored.
----------------------	--

*Exceptions*

<i>java.sql.SQLException</i>
------------------------------

*void imrcp.system.dbf.DbResultSet.updateRef (int columnIndex, Ref x) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbResultSet.updateRef (String columnLabel, Ref x) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbResultSet.updateRow () throws SQLException*

(NOT IMPLEMENTED) Updates the underlying database with the new contents of the current row of this **DbfResultSet** object. This method cannot be called when on the insert row.

*Exceptions*

<i>java.sql.SQLException</i>
------------------------------

*void imrcp.system.dbf.DbResultSet.updateRowId (int columnIndex, RowId x) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbResultSet.updateRowId (String columnLabel, RowId x) throws SQLException*

(NOT IMPLEMENTED)

*void imrcp.system.dbf.DbResultSet.updateShort (int columnIndex, short x) throws SQLException*

(NOT IMPLEMENTED) Updates the specified column with a short integer value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
<i>x</i>	short integer field value

*Exceptions*

<i>java.sql.SQLException</i>
------------------------------

`void imrcp.system.dbf.DbfResultSet.updateShort (String columnLabel, short x) throws SQLException`

(NOT IMPLEMENTED) Updates the specified column with a short integer value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnLabel</i>	data field label
<i>x</i>	short integer field value

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

`void imrcp.system.dbf.DbfResultSet.updateSQLXML (int columnIndex, SQLXML xmlObject) throws SQLException`

(NOT IMPLEMENTED)

`void imrcp.system.dbf.DbfResultSet.updateSQLXML (String columnLabel, SQLXML xmlObject) throws SQLException`

(NOT IMPLEMENTED)

`void imrcp.system.dbf.DbfResultSet.updateString (int columnIndex, String x) throws SQLException`

(NOT IMPLEMENTED) Updates the specified column with a String value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
<i>x</i>	String field value

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

`void imrcp.system.dbf.DbfResultSet.updateString (String columnLabel, String x) throws SQLException`

(NOT IMPLEMENTED) Updates the specified column with a String value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnLabel</i>	data field label
<i>x</i>	String field value

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbResultSet.updateTime (int columnIndex, Time x) throws SQLException*

(NOT IMPLEMENTED) Updates the specified column with a Time value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
<i>x</i>	Time field value

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbResultSet.updateTime (String columnLabel, Time x) throws SQLException*

(NOT IMPLEMENTED) Updates the specified column with a Time value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnLabel</i>	data field label
<i>x</i>	Time field value

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

*void imrcp.system.dbf.DbResultSet.updateTimestamp (int columnIndex, Timestamp x) throws SQLException*

(NOT IMPLEMENTED) Updates the specified column with a TimeStamp value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<i>columnIndex</i>	data field index (1's based)
<i>x</i>	TimeStamp field value

*Exceptions*

<i>java.sql.SQLException</i>	
------------------------------	--

`void imrcp.system.dbf.DbfResultSet.updateTimestamp (String columnLabel, Timestamp x) throws SQLException`

(NOT IMPLEMENTED) Updates the specified column with a Timestamp value. The updater methods are used to update column values in the current row or the insert row. The updater methods do not update the underlying database; instead the updateRow or insertRow methods are called to update the database.

*Parameters*

<code>columnLabel</code>	data field label
<code>x</code>	Timestamp field value

*Exceptions*

<code>java.sql.SQLException</code>	
------------------------------------	--

`boolean imrcp.system.dbf.DbfResultSet.wasNull () throws SQLException`

Determines if the current field was null.

*Returns*

true if current field was null, false otherwise

*Exceptions*

<code>java.sql.SQLException</code>	
------------------------------------	--

---

*The documentation for this class was generated from the following file:*

- system/dbf/DbfResultSet.java
- 

## imrcp.system.dbf.DbfString Class Reference

### Public Member Functions

- `DbfString` (byte[] yChars, int nLength)
- `void parseRecord` (DataInputStream oDataInputStream) throws Exception
- `int getInt ()`
- `long getLong ()`
- `float getFloat ()`
- `double getDouble ()`
- `String toString ()`

### Additional Inherited Members

---

#### Detailed Description

Holds data base file field natively as a string.

#### Author

Federal Highway Administration

*Version*

1.0

---

**Constructor & Destructor Documentation**

*imrcp.system.dbf.DbfString.DbfString (byte[] yChars, int nLength)*

Creates a new instance of **DbfString** with field name and data length defined.

*Parameters*

<i>yChars</i>	array of ASCII byte characters
<i>nLength</i>	length of byte character buffer

---

**Member Function Documentation**

*double imrcp.system.dbf.DbfString.getDouble ()*

Returns field data as a double precision number.

*Returns*

double precision number value of field data

Reimplemented from **imrcp.system.dbf.DbfField** (p.138).

*float imrcp.system.dbf.DbfString.getFloat ()*

Returns field data as a floating point number.

*Returns*

floating point number value of field data

Reimplemented from **imrcp.system.dbf.DbfField** (p.139).

*int imrcp.system.dbf.DbfString.getInt ()*

Returns field data as an integer.

*Returns*

integer value of field data

Reimplemented from **imrcp.system.dbf.DbfField** (p.139).

*long imrcp.system.dbf.DbfString.getLong ()*

Returns field data as an long integer.

*Returns*

long integer value of field data

Reimplemented from **imrcp.system.dbf.DbfField** (p.139).

*void imrcp.system.dbf.DbfString.parseRecord (DataInputStream oDataInputStream) throws Exception*

Reads the field and stores its contents as a string with leading and trailing whitespace removed.

*Parameters*

<i>oDataInputStream</i>	data file input stream
-------------------------	------------------------

*Exceptions*

<i>java.lang.Exception</i>
----------------------------

Reimplemented from **imrcp.system.dbf.DbfField** (p.139).

*String imrcp.system.dbf.DbfString.toString ()*

Returns field data as a string.

*Returns*

string value of field data

---

*The documentation for this class was generated from the following file:*

- system/dbf/DbfString.java
- 

[imrcp.forecast.mlp.MLPHurricane.Delegate Class Reference](#)

**Public Member Functions**

- void **run ()**
  - void **check ()**
- 

**Detailed Description**

**Delegate** object that handles the multi-threaded execution of the MLP Hurricane model for all the roadway segments in the Louisiana network.

---

**Member Function Documentation**

*void imrcp.forecast.mlp.MLPHurricane.Delegate.check ()*

Called after a **HurWork** is done processing all of its work. Checks if all other threads have finished processing. If they have applies the predictions to downstream roadway segments that do not have a prediction, writes the outputs to disk, and schedules the **MLPHurricane** instance to run again in one second and an hour after the current file's start time so the online prediction can be reran for that file with new traffic data.

*void imrcp.forecast.mlp.MLPHurricane.Delegate.run ()*

Calls  
**imrcp.system.Scheduling#getInstance () #execute (java.lang.Runnable)** on each **HurWork** in this **Delegate**

---

*The documentation for this class was generated from the following file:*

- forecast/mlp/MLPHurricane.java
-

---

## imrcp.geosrv.DEM Class Reference

### Public Member Functions

- void **reset** ()
- double **getElev** (int nLon, int nLat)
- String **getFilename** (int nX, int nY)
- int **compare** (int[] o1, int[] o2)

### Additional Inherited Members

---

### Detailed Description

Manages downloading, caching, and decoding PNG tiles from the Mapbox Terrian-DEM (Digital Elevation Map)

#### Author

Federal Highway Administration

---

### Member Function Documentation

#### `int imrcp.geosrv.DEM.compare (int[] o1, int[] o2)`

Integer arrays are used to represent a tile. Tile indices are stored in [x tile index, y tile index] so this compares tiles by x index then y index.

##### See also

`java.util.Comparator::compare(java.lang.Object, java.lang.Object)`

#### `double imrcp.geosrv.DEM.getElev (int nLon, int nLat)`

Looks up elevation at the given location. First this method determines the file name based off the map tile that contains the location. If the file is not saved on disk or its last modified time is too old based on `m_fileTimeout`, the file is downloaded and written to disk. Then the method checks if the file is in `m_oCache`, and loads it into memory if it is not. Finally the location in the tile is calculated and the elevation is decoded from the values in the color channels at that pixel.

##### Parameters

<code>nLon</code>	longitude of location in decimal degrees scaled to 7 decimal places
<code>nLat</code>	latitude of location in decimal degrees scaled to 7 decimal places

##### Returns

Elevation at the location in meters

#### `String imrcp.geosrv.DEM.getFilename (int nX, int nY)`

Generates the file name for the given x and y tile indices

#### Parameters

<i>nX</i>	x tile index
<i>nY</i>	y tile index

#### Returns

File name used to save files on disk

### `void imrcp.geosrv.DEM.reset ()`

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.system.BaseBlock** (*p.98*).

---

*The documentation for this class was generated from the following file:*

- geosrv/DEM.java
- 

## [imrcp.system.Directory Class Reference](#)

#### Classes

- class **RegisteredBlock**

#### Public Member Functions

- **Directory ()**
- **void init ()**
- **void run ()**
- synchronized int **register (ImrcpBlock oBlock, String[] sDependencies)**
- **void notifyBlocks (String[] sMessage)**
- synchronized void **notifyStart (String[] sMessage)**
- **void destroy ()**
- int **getContribPreference (int nContribId)**
- int **getPrecipPreference (int nContribId)**
- synchronized **ImrcpBlock lookup (String sName)**
- synchronized List<**BaseBlock**> **getStoresByObs (int nType)**
- **String getImrcpCode ()**
- **String getRemoteCode ()**
- **void doPost (HttpServletRequest oReq, HttpServletResponse oRes) throws IOException**
- **int compare (ImrcpBlock o1, ImrcpBlock o2)**

#### Static Public Member Functions

- static **Directory getInstance ()**

#### Static Public Attributes

- static final SimpleTimeZone **m\_oUTC** = new SimpleTimeZone(0, "")
- static final TimeZone **m\_oCST6CDT** = TimeZone.getTimeZone("CST6CDT")

---

#### Detailed Description

The **Directory** is the main system component that initializes the system by identifying and managing BaseBlocks and services available within the system.

*Author*

Federal Highway Administration

---

**Constructor & Destructor Documentation**

*imrcp.system.Directory.Directory ()*

Default constructor. Sets the singleton instance to this.

---

**Member Function Documentation**

*int imrcp.system.Directory.compare (ImrcpBlock o1, ImrcpBlock o2)*

Compares ImrcpBlocks by instance name

*void imrcp.system.Directory.destroy ()*

Called when the system is being shut down. Calls **HttpServlet#destroy()** for each **RegisteredBlock**, clears **m\_oRegistered** and calls **Scheduling#stop()**

*void imrcp.system.Directory.doPost (HttpServletRequest oReq, HttpServletResponse oRes) throws IOException*

*Parameters*

<i>oReq</i>	
<i>oRes</i>	

*Exceptions*

<i>IOException</i>
--------------------

*int imrcp.system.Directory.getContribPreference (int nContribId)*

Gets the preference value for the given IMRCP contributor id.

*Parameters*

<i>nContribId</i>	IMRCP contributor id
-------------------	----------------------

*Returns*

Preference value associated with the contributor id. Lower values are more preferred.

*String imrcp.system.Directory.getImrcpCode ()*

Get this IMRCP instance's ImrcpCode

*Returns*

the ImrcpCode of this instance of IMRCP

*static Directory imrcp.system.Directory.getInstance () [static]*

Get the singleton **Directory** instance

*Returns*

singleton **Directory** instance

*int imrcp.system.Directory.getPrecipPreference (int nContribId)*

Gets the preference value for the given IMRCP contributor id for precipitation rates.

*Parameters*

<i>nContribId</i>	IMRCP contributor id
-------------------	----------------------

*Returns*

Preference value associated with the contributor id for precipitation rates. Lower values are more preferred.

*String imrcp.system.Directory.getRemoteCode ()*

Get the configured remote IMRCP instance's ImrcpCode

*Returns*

the ImrcpCode of the configured remote instance of IMRCP

*synchronized List< BaseBlock > imrcp.system.Directory.getStoresByObs (int nType)*

Gets a List of all the data stores that provide data for the given observation type.

*Parameters*

<i>nType</i>	IMRCP Observation type id to search for
--------------	---

*Returns*

A List of all the data stores that provide data for the observation type

*void imrcp.system.Directory.init ()*

Initializes the **Directory**. This should be the first thing that happens at system start up. It sets up all of the BaseBlocks and handles starting their services.

*synchronized ImrcpBlock imrcp.system.Directory.lookup (String sName)*

Gets the instance of the registered **ImrcpBlock** with the given instance name.

*Parameters*

<i>sName</i>	instance name to lookup in <b>m_oRegistered</b>
--------------	---

*Returns*

Instance of the registered **ImrcpBlock** with the instance name if it exists, otherwise null.

*void imrcp.system.Directory.notifyBlocks (String[] sMessage)*

Sends the given notification message to the BaseBlocks that are subscribed to the **BaseBlock** the message is from

*Parameters*

<i>sMessage</i>	notification message in the format [ <b>BaseBlock</b> message is from, message name, <resources>...]
-----------------	--

*synchronized void imrcp.system.Directory.notifyStart (String[] sMessage)*

Queues the subscribers of the **BaseBlock** the "service started" message is from to start their service

#### Parameters

<code>sMessage</code>	[ <b>BaseBlock</b> message is from, "service started"]
-----------------------	--

`synchronized int imrcp.system.Directory.register (ImrcpBlock oBlock, String[] sDependencies)`

Registers the given **ImrcpBlock** with the given instance name dependencies.

#### Parameters

<code>oBlock</code>	The <b>ImrcpBlock</b> to reegister
<code>sDependencies</code>	contains the instance names of the ImrcpBlocks <code>oBlock</code> is dependent on

#### Returns

the registration id assigned to the **ImrcpBlock**, -1 if a block with `oBlock`'s instance name is already registered

`void imrcp.system.Directory.run ()`

Executes the last **Startup** in `m_oStartups` if there are any Startups in the queue

---

## Member Data Documentation

`final TimeZone imrcp.system.Directory.m_oCST6CDT =  
TimeZone.getTimeZone("CST6CDT") [static]`

TimeZone object for CST6CDT

`final SimpleTimeZone imrcp.system.Directory.m_oUTC = new SimpleTimeZone(0,  
"") [static]`

TimeZone object for UTC

---

*The documentation for this class was generated from the following file:*

- system/Directory.java
- 

## imrcp.forecast.mdss.DoMetroWrapper Class Reference

### Public Member Functions

- **DoMetroWrapper** (int nObsHrs, int nForecastHrs) throws Exception
- void **run** ()
- boolean **fillArrays** (int nLon, int nLat, boolean bBridge, int nTmtType, long lStartTime, **MetroFileset** oFiles)
- void **saveRoadcast** (int nLon, int nLat, long lStartTime)
- int **convertRoadCondition** (int nCond)
- int **imrcpToMetroRoadCond** (int nCond)
- int **getPrecipType** (**GriddedFileWrapper** oRapFile, long lTimestamp, int[] nRapIndices)
- void **fillArrays** (**MetroProcess** oProcess, int nIndex, **OsmWay** oWay)
- StringBuilder **log** (long lTimestamp) throws Exception

## Public Attributes

- **RoadcastData m\_oOutput**

## Static Public Attributes

- static boolean **g\_bLibraryLoaded** = true
- 

## Detailed Description

This class contains methods to create input arrays for METRo and save the returned output arrays. It also contains the wrapper function for the C code to run METRo directly. By doing this we bypass using METRo's Python code, which improves the performance greatly.

### Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

*imrcp.forecast.mdss.DoMetroWrapper.DoMetroWrapper (int nObsHrs, int nForecastHrs) throws Exception*

Allocates memory for all the arrays based off the number of observation and forecast hours

### Parameters

<i>nObsHrs</i>	number of observation hours
<i>nForecastHrs</i>	number of forecast hours

### Exceptions

<i>Exception</i>
------------------

---

## Member Function Documentation

*int imrcp.forecast.mdss.DoMetroWrapper.convertRoadCondition (int nCond)*

Takes the given METRo road condition and returns the corresponding IMRCP **ObsType#STPVT** value

### Parameters

<i>nCond</i>	METRo road condition value
--------------	----------------------------

### Returns

Corresponding IMRCP **ObsType#STPVT** value

*boolean imrcp.forecast.mdss.DoMetroWrapper.fillArrays (int nLon, int nLat, boolean bBridge, int nTmtType, long lStartTime, MetroFileset oFiles)*

Fills the input arrays for the C and Fortran code

### Parameters

<i>nLon</i>	longitude of the segment being used for the METRo model in decimal degrees scaled to 7 decimal places
-------------	---

<i>nLat</i>	latitude of the segment being used for the METRo model in decimal degrees scaled to 7 decimal places
<i>bBridge</i>	true if the segment is a bridge
<i>nTmtType</i>	1 if the segment is chemically treated, otherwise 0
<i>lStartTime</i>	start time of the METRo run
<i>oFiles</i>	Data files need for the METRo run

#### Returns

true if no errors or invalid values are found

```
void imrcp.forecast.mdss.DoMetroWrapper.fillArrays (MetroProcess oProcess, int nIndex, OsmWay oWay)
```

Fills the input arrays for the C and Fortran code. This method is used by **imrcp.web.Scenarios**

#### Parameters

<i>oProcess</i>	contains the information needed to run METRo for a <b>imrcp.web.Scenario</b>
<i>nIndex</i>	hour index of the scenario
<i>oWay</i>	OsmWay being used for this run of METRo

```
int imrcp.forecast.mdss.DoMetroWrapper.getPrecipType (GriddedFileWrapper oRapFile, long lTimestamp, int[] nRapIndices)
```

Reads the given RAP file at the given indices and returns the METRo precipitation type at the given timestamp

#### Parameters

<i>oRapFile</i>	RAP file to read
<i>lTimestamp</i>	query time in milliseconds since Epoch
<i>nRapIndices</i>	[x, y] location to query inside the file's grid

#### Returns

0 = no precipitation, 1= rain, 2 = snow

```
int imrcp.forecast.mdss.DoMetroWrapper.imrcpToMetroRoadCond (int nCond)
```

Takes the given IMRPC **ObsType#STPVT** value and returns the corresponding METRo road condition

#### Parameters

<i>nCond</i>	IMRPC <b>ObsType#STPVT</b> value
--------------	----------------------------------

#### Returns

corresponding METRo road condition

```
StringBuilder imrcp.forecast.mdss.DoMetroWrapper.log (long lTimestamp) throws Exception
```

Creates a StringBuilder containing information about the METRo run

*Parameters*

<i>lTimestamp</i>	run time of METRo
-------------------	-------------------

*Returns*

StringBuilder with log messages

*Exceptions*

<i>Exception</i>
------------------

**void imrcp.forecast.mdss.DoMetroWrapper.run ()**

Wrapper for **DoMetroWrapper#doMetroWrapper(int, double, double, int, int, double[], long[], long[], double[], long[], double[], double[], double[], int, double, double, double, double)**

**void imrcp.forecast.mdss.DoMetroWrapper.saveRoadcast (int nLon, int nLat, long lStartTime)**

Takes the output arrays from the C and Fortran code for the METRo run and stores them in a **imrcp.forecast.mdss.RoadcastData** object to be able to be used by other locations that have the same inputs to eliminate duplicate METRo runs

*Parameters*

<i>nLon</i>	longitude in decimal degrees scaled to 7 decimal places
<i>nLat</i>	latitude in decimal degrees scaled to 7 decimal places
<i>lStartTime</i>	start time of the METRo run

## Member Data Documentation

**boolean imrcp.forecast.mdss.DoMetroWrapper.g\_bLibraryLoaded = true [static]**

Flag to tell if the **Metro** shared library was loaded correctly

**RoadcastData imrcp.forecast.mdss.DoMetroWrapper.m\_oOutput**

Stores all of the outputs from a run of METRo

*The documentation for this class was generated from the following file:*

- forecast/mdss/DoMetroWrapper.java

## imrcp.system.Email Class Reference

### Public Member Functions

- **Email** (ArrayList< String > oTo, String sSubject, String sBody)
- **Email** (String[] sTo, String sSubject, String sBody)

## Public Attributes

- ArrayList< String > **m\_oTo**
  - String **m\_sSubject**
  - String **m\_sBody**
- 

## Detailed Description

Contains the data for a basic email (recipients, subject, and body).

### Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

*imrcp.system.Email.Email (ArrayList< String > oTo, String sSubject, String sBody)*

Constructs an **Email** with the given parameters

### Parameters

<i>oTo</i>	list of recipients
<i>sSubject</i>	subject of email
<i>sBody</i>	body of email

*imrcp.system.Email.Email (String[] sTo, String sSubject, String sBody)*

Constructs an **Email** with the given parameters

### Parameters

<i>sTo</i>	array of recipients
<i>sSubject</i>	subject of email
<i>sBody</i>	body of email

---

## Member Data Documentation

*ArrayList<String> imrcp.system.Email.m\_oTo*

List of recipients

*String imrcp.system.Email.m\_sBody*

**Email** subject's body

*String imrcp.system.Email.m\_sSubject*

**Email** subject

---

*The documentation for this class was generated from the following file:*

- system/Email.java
-

## imrcp.system.Emails Class Reference

### Static Public Member Functions

- static **Emails getInstance ()**
  - static void **send (Email oEmail)** throws Exception
- 

### Detailed Description

Singleton class that allows basic emails to be sent

#### Author

Federal Highway Administration

---

### Member Function Documentation

#### *static Emails imrcp.system.Emails.getInstance () [static]*

Gets the singleton instance

#### Returns

Singleton instance

#### *static void imrcp.system.Emails.send (Email oEmail) throws Exception [static]*

Sends the given email

#### Parameters

<i>oEmail</i>	<b>Email</b> to send
---------------	----------------------

#### Exceptions

<i>Exception</i>
------------------

---

*The documentation for this class was generated from the following file:*

- system/Emails.java
- 

## imrcp.store.EntryData Class Reference

### Public Member Functions

- abstract double **getValue** (int nHz, int nVrt)
- abstract void **setTimeDim** (int nIndex)
- void **getIndices** (double dLon1, double dLat1, double dLon2, double dLat2, int[] nIndices)
- void **getPointIndices** (double dLon, double dLat, int[] nIndices)
- int **getVrt** ()
- int **getHz** ()
- double **getCell** (int nHzIndex, int nVrtIndex, double[] dCorners)

### Public Attributes

- **ProjProfile m\_oProjProfile**

## Protected Member Functions

- final void **setProjProfile** (double[] dX, double[] dY, ProjectionImpl oProj, int nContrib)

## Protected Attributes

- int **m\_nObsTypeId**

---

## Detailed Description

Base class for representing one entry of gridded data found in **imrcp.store.GriddedFileWrapper**s

### Author

Federal Highway Administration

---

## Member Function Documentation

**double imrcp.store.EntryData.getCell (int nHrzIndex, int nVrtIndex, double[] dCorners)**

Fills the given double array with the latitude and longitudes of the corners of the cell of the grid at the given horizontal and vertical indices/coordinates and returns the value of the cell. Wrapper for **ProjProfile#getCell** and **getValue(int, int)**

### Parameters

<i>nHrzIndex</i>	x coordinate
<i>nVrtIndex</i>	y coordinate
<i>dCorners</i>	array to store the latitude and longitudes of the cell [top left lon, top left lat, top right lon, top right lat, bottom right lon, bottom right lat, bottom left lon, bottom right lat]

### Returns

value of the grid at the given x and y coordinates, if the coordinates are out of range, Double.NaN is returned

**int imrcp.store.EntryData.getHzr ()**

Get the last valid index (length - 1) of the horizontal axis

### Returns

Last valid index (length - 1) of the horizontal axis of this **EntryData**'s projected coordinate system grid

**void imrcp.store.EntryData.getIndices (double dLon1, double dLat1, double dLon2, double dLat2, int[] nIndices)**

Fills the given int[] with the indices of the grid that correspond to the lon/lat bounding box created by the given lon/lat points. The lon and lats do not have to be in a specific order as there is logic to handle either case in function that gets called. Wrapper for **ProjProfile#getIndices(double, double, double, double, int[])**

### Parameters

<i>dLon1</i>	longitude 1 in decimal degrees
<i>dLat1</i>	latitude 1 in decimal degrees

<i>dLon2</i>	longitude 2 in decimal degrees
<i>dLat2</i>	latitude 2 in decimal degrees
<i>nIndices</i>	array to be filled with the indices of the grid corresponding to the lon/lat points. [min x index, max y index, max x index, min y index]

*void imrcp.store.EntryData.getPointIndices (double dLon, double dLat, int[] nIndices)*

Fills the given int[] with the indices of the grid that correspond to the lon/lat point. Wrapper for **ProjProfile#getPointIndices (double, double, int[])**

#### Parameters

<i>dLon</i>	longitude in decimal degrees
<i>dLat</i>	latitude in decimal degrees
<i>nIndices</i>	array to be filled with the indices of the grid corresponding to the lon/lat point.

*abstract double imrcp.store.EntryData.getValue (int nHz, int nVrt) [abstract]*

Gets the value of the grid at the given x and y coordinates

#### Parameters

<i>nHz</i>	x coordinate
<i>nVrt</i>	y coordinate

#### Returns

value of the grid at the given x and y coordinates, if the coordinates are out of range, Double.NaN is returned

Reimplemented in **imrcp.store.ByteObsEntryData** (p.106), **imrcp.store.FloatObsEntryData** (p.226), **imrcp.store.GribEntryData** (p.247), and **imrcp.store.NcfEntryData** (p.370).

*int imrcp.store.EntryData.getVrt ()*

Get the last valid index (length - 1) of the vertical axis

#### Returns

Last valid index (length - 1) of the vertical axis of this **EntryData**'s projected coordinate system grid

*final void imrcp.store.EntryData.setProjProfile (double[] dX, double[] dY, ProjectionImpl oProj, int nContrib) [protected]*

Sets **m\_oProjProfile** by calling **ProjProfiles#getInstance ()#newProfile (double[], double[], ucar.unidata.geoloc.ProjectionImpl, int)** with the given parameters

#### Parameters

<i>dX</i>	array containing the values of the x axis of the projected coordinate system
<i>dY</i>	array containing the values of the y axis of the projected coordinate system
<i>oProj</i>	object created with the parameters of the projected coordinate system

<i>nContrib</i>	IMRCP contributor Id for the source of the projected coordinate system
-----------------	--

*abstract void imrcp.store.EntryData.setTimeDim (int nIndex) [abstract]*

Sets the time dimension to the given index, if applicable.

#### Parameters

<i>nIndex</i>	
---------------	--

Reimplemented in **imrcp.store.ByteObsEntryData** (p.106), **imrcp.store.FloatObsEntryData** (p.226), **imrcp.store.GribEntryData** (p.247), and **imrcp.store.NcfEntryData** (p.371).

## Member Data Documentation

*int imrcp.store.EntryData.m\_nObsTypId [protected]*

Observation type of the entry

*ProjProfile imrcp.store.EntryData.m\_oProjProfile*

Object used to project coordinates from lon/lat to projected coordinate system and vice versa

*The documentation for this class was generated from the following file:*

- store/EntryData.java

## imrcp.comp.EventComp Class Reference

### Public Member Functions

- void **reset** ()
- void **process** (String[] sNotification)

### Protected Member Functions

- void **processFile** (String sFile, long lRunTime)
- abstract long **getEvents** (String sFile, long lFileTime) throws Exception

### Protected Attributes

- ArrayList< EventObs > **m\_oEvents** = new ArrayList()
- int **m\_nFileFrequency**
- FilenameFormatter **m\_oFilenameFormat**
- String **m\_sPreviousFile** = null
- String **m\_sDateFormat**

### Static Protected Attributes

- static final String **HEADER** = "extid,name,type,starttime,endtime,updatedtime,lon,lat,lanesaffected,direction\n"

### Additional Inherited Members

## Detailed Description

Generic class used to manage real time event data

### Author

Federal Highway Administration

---

## Member Function Documentation

*abstract long imrcp.comp.EventComp.getEvents (String sFile, long lFileTime) throws Exception [abstract], [protected]*

Child class override this method which parses event files and updates the currently active events in memory.

### Parameters

<i>sFile</i>	Event file to parse
<i>lFileTime</i>	Timestamp for the file in milliseconds since Epoch

### Returns

Timestamp the file was last updated in milliseconds since Epoch

### Exceptions

<i>Exception</i>	
------------------	--

Reimplemented in **imrcp.comp.LAc2cEventsComp** (*p.308*), **imrcp.comp.LADOTD511Line** (*p.310*), **imrcp.comp.LADOTD511Point** (*p.312*), **imrcp.comp.OhGoConstruction** (*p.423*), and **imrcp.comp.OhGoIncidents** (*p.423*).

*void imrcp.comp.EventComp.process (String[] sNotification)*

Called when a message from **imrcp.system.Directory** is received. If the message is "file download" **EventComp#processFile(java.lang.String, long)** is called.

### Parameters

<i>sNotification</i>	[BaseBlock message is from, message name, name of downloaded file, start time of the file in milliseconds since Epoch]
----------------------	--

Reimplemented from **imrcp.system.BaseBlock** (*p.97*).

*void imrcp.comp.EventComp.processFile (String sFile, long lRunTime) [protected]*

Processes the given file and updates the list of currently active events based on the contents of the file.

### Parameters

<i>sFile</i>	File to parse
<i>lRunTime</i>	Start time of the file

*void imrcp.comp.EventComp.reset ()*

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.system.BaseBlock** (*p.98*).

Reimplemented in [imrcp.comp.LAc2cEventsComp](#) (p.308), [imrcp.comp.LADOTD511Line](#) (p.310), and [imrcp.comp.LADOTD511Point](#) (p.312).

---

## Member Data Documentation

*final String imrcp.comp.EventComp.HEADER =  
"extid,name,type,starttime,endtime,updatedtime,lon,lat,lanesaffected,direction\n" [s  
tatic], [protected]*

Header for .csv files

*int imrcp.comp.EventComp.m\_nFileFrequency [protected]*

How often a file should be made to store event observations in milliseconds

*ArrayList<EventObs> imrcp.comp.EventComp.m\_oEvents = new  
ArrayList() [protected]*

Stores the currently active events

*FilenameFormatter imrcp.comp.EventComp.m\_oFilenameFormat [protected]*

Object used to create time dependent file names to save on disk

*String imrcp.comp.EventComp.m\_sDateFormat [protected]*

Format string to use in the creation of `java.text.SimpleDateFormat` objects to parse and format date strings

*String imrcp.comp.EventComp.m\_sPreviousFile = null [protected]*

Keeps track of the last file events were written to

---

*The documentation for this class was generated from the following file:*

- comp/EventComp.java
- 

## imrcp.store.EventObs Class Reference

### Public Member Functions

- `EventObs()`
- `EventObs(EventObs oEvent)`
- `EventObs(EventObs oOriginal, int nLon, int nLat, Id oObjId)`
- `EventObs(int nObsTypeId, int nContribId, Id oObjId, long lObsTime1, long lObsTime2, long lTimeRecv, int nLat1, int nLon1, int nLat2, int nLon2, short tElev, double dValue, int nLanes)`
- `void reset()`
- `void copyValues(EventObs oEvent)`
- `void writeToFile(Writer oOut, SimpleDateFormat oSdf) throws Exception`
- `void close(long lTime, SimpleDateFormat oSdf)`

### Public Attributes

- `String m_sExtId`
- `int m_nLanesAffected`
- `String m_sEventName`
- `String m_sType`

- String **m\_sDir**
- boolean **m\_bOpen**
- boolean **m\_bUpdated**
- ArrayList< int[]> **m\_nPoints** = null

#### Static Public Attributes

- static Comparator< **EventObs** > **EXTCOMP** = (**EventObs** o1, **EventObs** o2) ->  
o1.m\_sExtId.compareTo(o2.m\_sExtId)
  - static Comparator< **EventObs** > **EXTOBJCOMP**
- 

#### Detailed Description

An extension of **Obs** with additional fields specific to observations representing events affecting roadways like incidents, work zones, and flooded roads

#### Author

Federal Highway Administration

---

#### Constructor & Destructor Documentation

##### *imrcp.store.EventObs.EventObs ()*

Default constructor. Does nothing.

##### *imrcp.store.EventObs.EventObs (EventObs oEvent)*

Copy constructor. Wrapper for **copyValues (imrcp.store.EventObs)** with the given **EventObs**.

##### Parameters

<i>oEvent</i>	<b>EventObs</b> to copy
---------------	-------------------------

##### *imrcp.store.EventObs.EventObs (EventObs oOriginal, int nLon, int nLat, Id oObjId)*

Constructs a new **EventObs** for the given event at a new location. This is used when an event is associated with multiple directions of travel or roadway segments

##### Parameters

<i>oOriginal</i>	original <b>EventObs</b>
<i>nLon</i>	longitude of event in decimal degrees scaled to 7 decimal places
<i>nLat</i>	latitude of event in decimal degrees scaled to 7 decimal places
<i>oObjId</i>	Imrcp Id of the object associated with the event at the given location

##### *imrcp.store.EventObs.EventObs (int nObsTypeld, int nContribld, Id oObjId, long IObsTime1, long IObsTime2, long ITimeRecv, int nLat1, int nLon1, int nLat2, int nLon2, short tElev, double dValue, int nLanes)*

Wrapper for **Obs#Obs (int, int, imrcp.system.Id, long, long, long, int, int, int, short, double)** and then sets the given number of lanes affected

*Parameters*

<i>nObsTypeId</i>	IMRCP observation type id, should be <b>imrcp.system.ObsType#EVT</b> for events
<i>nContribId</i>	IMRCP contributor id
<i>oObjId</i>	IMRCP object Id of the object associated with <b>Obs</b>
<i>lObsTime1</i>	time the event started in milliseconds since Epoch
<i>lObsTime2</i>	time the event ends (can be an estimation) in milliseconds since Epoch. If the event is active and there is no estimated end time use -1
<i>lTimeRecv</i>	time the event was received/updated in milliseconds since Epoch
<i>nLat1</i>	minimum latitude of event in decimal degrees scaled to 7 decimal places
<i>nLon1</i>	minimum longitude of event in decimal degrees scaled to 7 decimal places
<i>nLat2</i>	maximum latitude of event in decimal degrees scaled to 7 decimal places
<i>nLon2</i>	maximum longitude of event in decimal degrees scaled to 7 decimal places
<i>tElev</i>	elevation of event in meters
<i>dValue</i>	event type. <b>imrcp.system.ObsType#LOOKUP</b> contains all possible values
<i>nLanes</i>	number of lanes affected by the event

---

**Member Function Documentation**

**void imrcp.store.EventObs.close (long lTime, SimpleDateFormat oSdf)**

Closes the event by setting the end time to the given timestamp. Some child classes use the `SimpleDateFormat` even though the base implementation does not need it.

*Parameters*

<i>lTime</i>	end time of the event in milliseconds since Epoch
<i>oSdf</i>	date parsing/formatting object

Reimplemented in `imrcp.store.LADOTD511Event` (p.309), and `imrcp.store.LaDOTDEvent` (p.313).

**void imrcp.store.EventObs.copyValues (EventObs oEvent)**

Copies values from the given `EventObs`

*Parameters*

<i>oEvent</i>	Event to copy values from
---------------	---------------------------

**void imrcp.store.EventObs.reset ()**

Resets member variables to default values

Reimplemented in `imrcp.store.LaDOTDEvent` (p.314).

**void imrcp.store.EventObs.writeToFile (Writer oOut, SimpleDateFormat oSdf) throws Exception**

Writes the parameters of the event to the given writer in IMRCP's CSV work zone and event file format

#### Parameters

<i>oOut</i>	Writer that writes the event
<i>oSdf</i>	date formatting object

#### Exceptions

<i>Exception</i>	
------------------	--

Reimplemented in **imrcp.store.LaDOTDEvent** (*p.314*).

---

#### Member Data Documentation

*Comparator<EventObs> imrcp.store.EventObs.EXTCOMP = (EventObs o1, EventObs o2) -> o1.m\_sExtId.compareTo(o2.m\_sExtId) [static]*

C.compares **EventObs** by their external system id

*Comparator<EventObs> imrcp.store.EventObs.EXTOBJCOMP [static]*

```
Initial value:= (EventObs o1, EventObs o2) ->
{
    int nRet = EXTCOMP.compare(o1, o2);
    if (nRet == 0)
        nRet = Id.COMPARATOR.compare(o1.m_oObjId, o2.m_oObjId);

    return nRet;
}
```

C.compares **EventObs** by their external system id and then their IMRCP Id

*boolean imrcp.store.EventObs.m\_bOpen*

Flag indicating if the event is active(open) or inactive(closed)

*boolean imrcp.store.EventObs.m\_bUpdated*

Flag indicating if the event was updated in the most recent data file

*int imrcp.store.EventObs.m\_nLanesAffected*

Number of lanes affected by the event

*ArrayList<int[]> imrcp.store.EventObs.m\_nPoints = null*

Array of points used for event associated with a line string instead of a single point

*String imrcp.store.EventObs.m\_sDir*

Direction of travel the event is affecting

*String imrcp.store.EventObs.m\_sEventName*

Name of the event

*String imrcp.store.EventObs.m\_sExtId*

Identifier used by the external system that provided the event information

---

*The documentation for this class was generated from the following file:*

- store/EventObs.java
- 
-

## imrcp.system.ExtMapping Class Reference

### Public Member Functions

- void **reset** ()
- boolean **start** () throws Exception
- **Id[] getMapping** (int nContribId, String sExtId)
- boolean **hasMapping** (**Id** oId)

### Additional Inherited Members

---

#### Detailed Description

Keeps track of and maps external system identifiers to their corresponding IMRCP IDs.

#### Author

Federal Highway Administration

---

#### Member Function Documentation

##### **Id[] imrcp.system.ExtMapping.getMapping (int nContribId, String sExtId)**

Retrieves the IMRCP IDs that correspond to the given contributor id and external system id

###### Parameters

<b>nContribId</b>	Contributor id
<b>sExtId</b>	external system id

###### Returns

**Id[]** of IMRCP IDs that map to the external system id of the contributor, null if no mapping exists

##### **boolean imrcp.system.ExtMapping.hasMapping (Id oId)**

Tells whether or not the given **Id** has a mapping to an external system identifier

###### Parameters

<b>oId</b>	IMRCP ID
------------	----------

###### Returns

true if a mapping exists for the **Id**, otherwise false

##### **void imrcp.system.ExtMapping.reset ()**

Sets configurable member variables from the **BlockConfig** object, should be overridden by child classes.

Reimplemented from **imrcp.system.BaseBlock** (p.98).

##### **boolean imrcp.system.ExtMapping.start () throws Exception**

###### Returns

### *Exceptions*

<i>Exception</i>
Reimplemented from <b>imrcp.system.BaseBlock</b> (p.99).

---

*The documentation for this class was generated from the following file:*

- system/ExtMapping.java
- 

## imrcp.store.FileCache Class Reference

### Public Member Functions

- void **reset** ()
- void **process** (String[] sMessage)
- boolean **start** () throws Exception
- **FileWrapper getFile** (long lTimestamp, long lRefTime)
- **FileWrapper getFile** (long lTimestamp, long lRefTime, int nLon, int nLat)
- void **addDirForTime** (long lTimestamp, ArrayList< String > oDirs, int nFormatIndex, int nTileX, int nTileY)
- boolean **loadFileToMemory** (String sFullPath, int nFormatIndex)
- void **lruClear** ()
- void **execute** ()
- boolean **matches** (**FileWrapper** oFile, long lTimestamp, long lRefTime)
- boolean **stop** ()
- void **loadAFile** ()

### Public Attributes

- int[] **m\_nSubObsTypes**

### Static Public Attributes

- static int **VALID** = 0
- static int **START** = 2
- static int **END** = 1
- static Comparator< String > **REFTIMECOMP**
- static Comparator< **FileWrapper** > **FILENAMECOMP**

### Protected Member Functions

- **FileCache** ()
- abstract **FileWrapper getNewFileWrapper** ()
- boolean **loadFileToCache** (long lTimestamp, long lRefTime)
- boolean **loadFileToCache** (long lTimestamp, long lRefTime, int nLon, int nLat)

### Protected Attributes

- ArrayList< **FileWrapper** > **m\_oCache**
- boolean **m\_bFilesAppendable**
- int **m\_nPeriod**
- int **m\_nOffset**
- int **m\_nMaxForecast**
- int **m\_nFileFrequency**
- long **m\_lLastFileMissingTime** = 0
- int **m\_nTimeout**
- **FilenameFormatter[] m\_oFormatters**
- int **m\_nLimit**

- `ArrayList< String > m_oDoNotLoad = new ArrayList()`
- `int m_nSlashesForBase`
- `ReentrantReadWriteLock m_oLock = new ReentrantReadWriteLock(true)`

#### **Static Protected Attributes**

- `static Comparator< FileWrapper > TEMPORALFILECOMP`
  - `static Comparator< FileWrapper > SPATIALFILECOMP`
- 

#### **Detailed Description**

Base class used for data stores. Contains methods for managing the caching of data files on disk into memory for quick access.

#### *Author*

Federal Highway Administration

---

#### **Constructor & Destructor Documentation**

##### *`imrcp.store.FileCache.FileCache () [protected]`*

Default constructor. Wrapper for super()

---

#### **Member Function Documentation**

##### *`void imrcp.store.FileCache.addDirForTime (long lTimestamp, ArrayList< String > oDirs, int nFormatIndex, int nTileX, int nTileY)`*

Adds a directory that is valid for the given parameters.

#### *Parameters*

<i>lTimestamp</i>	query time in milliseconds since Epoch
<i>oDirs</i>	list to store valid directories
<i>nFormatIndex</i>	index of <code>m_oFormatters</code> to use
<i>nTileX</i>	x coordinate of tile
<i>nTileY</i>	y coordinate of tile

##### *`void imrcp.store.FileCache.execute ()`*

Checks for and removes any file in the cache that has not been used for the configured timeout.

Reimplemented from `imrcp.system.BaseBlock` (p.95).

Reimplemented in `imrcp.store.SpatialFileCache` (p.498).

##### *`FileWrapper imrcp.store.FileCache.getFile (long lTimestamp, long lRefTime)`*

Wrapper for `getFile(long, long, int, int)` with `Integer.MIN_VALUE` passed as the longitude and latitude. Used for FileCaches that are not SpatialFileCaches

#### *Parameters*

<i>lTimestamp</i>	query time in milliseconds since Epoch
<i>lRefTime</i>	reference time in millisecond since Epoch

#### Returns

**FileWrapper** that is valid for the given query and reference time. If no valid file exists, null is returned

*FileWrapper imrcp.store.FileCache.getFile (long lTimestamp, long lRefTime, int nLon, int nLat)*

Retrieves the "best" file that is valid for the given query and reference time. If that file is not in the cache, it is loaded into the cache in this process.

#### Parameters

<i>lTimestamp</i>	query time in milliseconds since Epoch
<i>lRefTime</i>	reference time in milliseconds since Epoch
<i>nLon</i>	longitude of query in decimal degrees scaled to 7 decimal places. Used <code>Integer.MIN_VALUE</code> if not a <b>SpatialFileCache</b>
<i>nLat</i>	latitude of query in decimal degrees scaled to 7 decimal places. Used <code>Integer.MIN_VALUE</code> if not a <b>SpatialFileCache</b>

#### Returns

Reimplemented in **imrcp.store.SpatialFileCache** (p.499).

*abstract FileWrapper imrcp.store.FileCache.getNewFileWrapper () [abstract], [protected]*

Returns the correct type of **FileWrapper** for the implemented class

Reimplemented in **imrcp.store.AHPSStore** (p.80), **imrcp.store.AlertsStore** (p.85), **imrcp.store.BinObsStore** (p.101), **imrcp.store.CAPStore** (p.110), **imrcp.store.CsvStore** (p.124), **imrcp.store.GeotabStore** (p.231), **imrcp.store.GribStore** (p.249), **imrcp.store.MetroStore** (p.341), **imrcp.store.MLPStore** (p.364), **imrcp.store.NDFDQpfStore** (p.377), **imrcp.store.NHCStore** (p.392), **imrcp.store.RAPStore** (p.470), **imrcp.store.TrafficEventStore** (p.538), **imrcp.store.TrafficSpeedStore** (p.539), **imrcp.store.WeatherStore** (p.564), **imrcp.store.WxDESubStore** (p.570), and **imrcp.web.tiles.TileCache** (p.522).

*void imrcp.store.FileCache.loadAFile ()*

Attempts to load any file that can be found in the directories for this store. This is only called at start up when no recent files could be found to try and ensure that projection profiles and static files get initialized at start up.

*boolean imrcp.store.FileCache.loadFileToCache (long lTimestamp, long lRefTime) [protected]*

Wrapper for `loadFileToCache(long, long, int, int)` with `Integer.MIN_VALUE` passed as the longitude and latitude. Used for FileCaches that are not SpatialFileCaches

#### Parameters

<i>lTimestamp</i>	query time in milliseconds since Epoch
<i>lRefTime</i>	reference time in millisecond since Epoch

#### Returns

true if a file valid for the query and reference was already in the cache or was loaded into the cache, otherwise false

`boolean imrcp.store.FileCache.loadFileToCache (long lTimestamp, long lRefTime, int nLon, int nLat) [protected]`

Attempts to load a file into memory that is valid for the query time and reference time. The longitude and latitude are only used in the implementation of this function for SpatialFileCaches.

*Parameters*

<code>lTimestamp</code>	query time in milliseconds since Epoch
<code>lRefTime</code>	reference time in milliseconds since Epoch
<code>nLon</code>	longitude of query in decimal degrees scaled to 7 decimal places. Used <code>Integer.MIN_VALUE</code> if not a <b>SpatialFileCache</b>
<code>nLat</code>	latitude of query in decimal degrees scaled to 7 decimal places. Used <code>Integer.MIN_VALUE</code> if not a <b>SpatialFileCache</b>

*Returns*

true if a file valid for the query and reference was already in the cache or was loaded into the cache, otherwise false

Reimplemented in **imrcp.store.SpatialFileCache** (p.499).

`boolean imrcp.store.FileCache.loadFileToMemory (String sFullPath, int nFormatIndex)`

Attempts to load the given file path into memory and places it in the cache

*Parameters*

<code>sFullPath</code>	File path to load
<code>nFormatIndex</code>	index of <code>m_oFormatters</code> to use

*Returns*

true if the file is successfully loaded into the cache

Reimplemented in **imrcp.store.NHCStore** (p.392), and **imrcp.web.tiles.TileCache** (p.522).

`void imrcp.store.FileCache.lruClear ()`

Determines and removes the 2 least recently used files from the cache.

`boolean imrcp.store.FileCache.matches (FileWrapper oFile, long lTimestamp, long lRefTime)`

Determines if the given file is valid for the given query and reference time

*Parameters*

<code>oFile</code>	file to check
<code>lTimestamp</code>	query time in milliseconds since Epoch
<code>lRefTime</code>	reference time in milliseconds since Epoch

*Returns*

true if the reference time is the equal to or after the valid/received time of the file and the query time is inbetween the start and end time of the file, otherwise false

`void imrcp.store.FileCache.process (String[] sMessage)`

Called when a message from `imrcp.system.Directory` is received. Attempts to load newly downloaded files into memory.

*Parameters*

<code>sMessage</code>	[BaseBlock message is from, message name, file1, file2, ..., filen]
-----------------------	---

Reimplemented from `imrcp.system.BaseBlock` (p.97).

Reimplemented in `imrcp.store.BinObsStore` (p.101), and `imrcp.store.SpatialFileCache` (p.500).

`void imrcp.store.FileCache.reset ()`

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from `imrcp.web.SecureBaseBlock` (p.492).

Reimplemented in `imrcp.store.BinObsStore` (p.102), `imrcp.store.GribStore` (p.250), `imrcp.store.MetroStore` (p.341), `imrcp.store.WeatherStore` (p.564), `imrcp.web.tiles.GribTileCache` (p.250), `imrcp.web.tiles.NcfTileCache` (p.372), and `imrcp.web.tiles.TileCache` (p.522).

`boolean imrcp.store.FileCache.start () throws Exception`

Attempts to load the most recent files into memory to be ready for system use. Then sets a schedule to execute on a fixed interval.

*Returns*

true if no Exception are thrown

*Exceptions*

<code>Exception</code>
------------------------

Reimplemented from `imrcp.system.BaseBlock` (p.99).

Reimplemented in `imrcp.store.SpatialFileCache` (p.500), and `imrcp.web.tiles.TileCache` (p.522).

`boolean imrcp.store.FileCache.stop ()`

Removes and cleans up resources of the files currently in the cache

*Returns*

Reimplemented from `imrcp.system.BaseBlock` (p.100).

---

## Member Data Documentation

`int imrcp.store.FileCache.END = 1 [static]`

Index of the end time in long[] frequently used by stores

`Comparator<FileWrapper> imrcp.store.FileCache.FILENAMECOMP [static]`

```
Initial value:= (FileWrapper o1, FileWrapper o2) ->
{
    return o1.m_sFilename.compareTo(o2.m_sFilename);
}
```

Compare FileWrappers by file name

*boolean imrcp.store.FileCache.m\_bFilesAppendable [protected]*

Flag indicating if the files cached by this store can have contents appended to them after the file is initially created

*long imrcp.store.FileCache.m\_lLastFileMissingTime = 0 [protected]*

Time in milliseconds since Epoch that a missing file was written to the log

*int imrcp.store.FileCache.m\_nFileFrequency [protected]*

Expected time in milliseconds between the collection of two consecutive files

*int imrcp.store.FileCache.m\_nLimit [protected]*

Maximum number of files that can be in the cache

*int imrcp.store.FileCache.m\_nMaxForecast [protected]*

Maximum time in milliseconds that a forecast can be for files in this store

*int imrcp.store.FileCache.m\_nOffset [protected]*

Schedule offset from midnight in seconds

*int imrcp.store.FileCache.m\_nPeriod [protected]*

Period of execution in seconds

*int imrcp.store.FileCache.m\_nSlashesForBase [protected]*

The number of file separators starting from the end of a file name there are until the base directory for the store

*int [] imrcp.store.FileCache.m\_nSubObsTypes*

Observation type ids of the data this store provides

*int imrcp.store.FileCache.m\_nTimeout [protected]*

Time in milliseconds files can stay in the cache unused

*ArrayList<FileWrapper> imrcp.store.FileCache.m\_oCache [protected]*

Used to cache data files

*ArrayList<String> imrcp.store.FileCache.m\_oDoNotLoad = new*

*ArrayList() [protected]*

List of files that failed to load

*FilenameFormatter [] imrcp.store.FileCache.m\_oFormatters [protected]*

Format objects used to determine time dependent file names on disk

*ReentrantReadWriteLock imrcp.store.FileCache.m\_oLock = new*

*ReentrantReadWriteLock(true) [protected]*

Read write lock used to maintain data integrity as asynchronous requests are made

*Comparator<String> imrcp.store.FileCache.REFTIMECOMP [static]*

```
Initial value:= (String o1, String o2) ->
{
    int nIndex1 = o1.lastIndexOf("_") + 1;
    String s1 = o1.substring(nIndex1, o1.indexOf(".", nIndex1));
    int nIndex2 = o2.lastIndexOf("_") + 1;
    String s2 = o2.substring(nIndex2, o2.indexOf(".", nIndex2));
    int nReturn = s2.compareTo(s1);
    if (nReturn == 0)
```

```

        {
            nIndex1 = o1.lastIndexOf("_", nIndex1 - 2);
            nIndex1 = o1.lastIndexOf("_", nIndex1 - 1) + 1;
            s1 = o1.substring(nIndex1, o1.indexOf("_", nIndex1));
            nIndex2 = o2.lastIndexOf("_", nIndex2 - 2);
            nIndex2 = o2.lastIndexOf("_", nIndex2 - 1) + 1;
            s2 = o2.substring(nIndex2, o2.indexOf("_", nIndex2));
            nReturn = s1.compareTo(s2);
        }
        return nReturn;
    }
}

```

Compares FileWrappers by parsing their reference/valid time from their file names

*Comparator<FileWrapper> imrcp.store.FileCache.SPATIALFILECOMP [static],  
[protected]*

```

Initial value:= (FileWrapper o1, FileWrapper o2) ->
{
    SpatialFileWrapper oSfw1 = (SpatialFileWrapper)o1;
    SpatialFileWrapper oSfw2 = (SpatialFileWrapper)o2;
    int nRet = oSfw1.m_nTileX - oSfw2.m_nTileX;
    if (nRet == 0)
    {
        nRet = oSfw1.m_nTileY - oSfw2.m_nTileY;
        if (nRet == 0)
            nRet = TEMPORALFILECOMP.compare(o1, o2);
    }
    return nRet;
}

```

Compares SpatialFileWrappers by x tile index, y tile index, and then uses  
**FileCache#TEMPORALFILECOMP**

*int imrcp.store.FileCache.START = 2 [static]*

Index of the start time in long[] frequently used by stores

*Comparator<FileWrapper> imrcp.store.FileCache.TEMPORALFILECOMP [static],  
[protected]*

```

Initial value:= (FileWrapper o1, FileWrapper o2) ->
{
    int nReturn = nReturn = o1.m_nFormatIndex - o2.m_nFormatIndex;
    if (nReturn == 0)
    {
        nReturn = Long.compare(o2.m_lValidTime, o1.m_lValidTime);
        if (nReturn == 0)
        {
            nReturn = Long.compare(o1.m_lStartTime, o2.m_lStartTime);
            if (nReturn == 0)
                Long.compare(o1.m_lEndTime, o2.m_lEndTime);
        }
    }
    return nReturn;
}

```

Compares FileWrappers by format index, valid time(descending order), start time, then end time

*int imrcp.store.FileCache.VALID = 0 [static]*

Index of the valid time in long[] frequently used by stores

*The documentation for this class was generated from the following file:*

- store/FileCache.java

## imrcp.system.FilenameFormatter Class Reference

### Public Member Functions

- **FilenameFormatter** (String sPattern)
- synchronized String **format** (long lRcvd, long lStart, long lEnd, int nOffset, String... sStrings)
- synchronized String **format** (long lRcvd, long lStart, long lEnd, String... sStrings)
- synchronized int **parse** (String sFilename, long[] lTimes)
- synchronized int **parseRecv** (String sFilename, long[] lTimes)
- synchronized void **parseTile** (String sFilename, int[] nTile)
- String **getExtension** ()

### Protected Member Functions

- **FilenameFormatter** ()
- void **setyyyyMMdd** (int nPos)
- void **setHH** (int nPos)
- void **setHHmm** (int nPos)
- void **setHHmmss** (int nPos)
- void **yMdHm** (int nPos)
- void **yMdH** (int nPos)
- void **yMd** (int nPos)
- void **yM** (int nPos)
- void **y** (int nPos)
- void **HHmm** (int nPos)
- void **H** (int nPos)
- void **n** (int nPos, int nOffset)
- void **nnn** (int nPos, int nOffset)
- void **S** (int nPos, String sString)

---

### Detailed Description

This class is used to generate time dependent file names that contain some metadata about what is contained in the file. Data files collected and created by the collect and comp packages while use this class to create the file names. In general file names in the IMRCP system have this format: contrib\_extrainfo0\_extrainfo1\_...\_starttime\_endtime\_validtime.ext where contrib is the up to 6 character alphanumeric string representing the IMRCP contributor ID, extrainfos can be anything like the size of grid for gridded data files, starttime, endtime, and validtime are all date/time strings in the format yyyyMMddHHmm for times in UTC, and ext is the file extension. An example of a pattern string is: "/opt/imrcp-data-prod/rtma/%ryM/%ryMd/rtma\_030\_%syMdHm\_%eyMdHm\_%ryMdHm.grb2" The " indicates the start of a replacement pattern. The character immediately after the " indicates the type of pattern.

Possible types of pattern are:

r = received(valid) time

s = start time

e = end time

S = String

Implemented replacement patters are: S = String

yMdHm = yyyyMMddHHmm (year month day hour(0-23) minute)

yMdH = yyyyMMddHH (year month day hour(0-23))

yMd = yyyyMMdd (year month day)

yM = yyyyMM (year month)

y = yyyy (year)  
nnn = 3 digit file offset index  
n = 1 digit file offset index  
HHmm = hour(0-23) minute  
H = hour(0-23)

#### Author

Federal Highway Administration

---

### Constructor & Destructor Documentation

#### *imrcp.system.FilenameFormatter.FilenameFormatter () [protected]*

Default constructor. Does nothing.

#### *imrcp.system.FilenameFormatter.FilenameFormatter (String sPattern)*

Constructs a **FilenameFormatter** with the given pattern and sets its timezone to UTC.

##### Parameters

<i>sPattern</i>	file name pattern
-----------------	-------------------

---

### Member Function Documentation

#### *synchronized String imrcp.system.FilenameFormatter.format (long lRcvd, long lStart, long lEnd, int nOffset, String... sStrings)*

Gets the file name using the given parameters to format **m\_sPattern**

##### Parameters

<i>lRcvd</i>	received time in milliseconds since Epoch
<i>lStart</i>	start time in milliseconds since Epoch
<i>lEnd</i>	end time in milliseconds since Epoch
<i>nOffset</i>	file offset index
<i>sStrings</i>	string array used to replace 'S's in the pattern in the order of the array

##### Returns

The formatted filename

#### *synchronized String imrcp.system.FilenameFormatter.format (long lRcvd, long lStart, long lEnd, String... sStrings)*

Wrapper for **format(long, long, long, int, java.lang.String...)** passing a file offset index of zero.

##### Parameters

<i>lRcvd</i>	received time in milliseconds since Epoch
<i>lStart</i>	start time in milliseconds since Epoch
<i>lEnd</i>	end time in milliseconds since Epoch

<i>sStrings</i>	string array used to replace 'S's in the pattern in the order of the array
-----------------	--

*Returns*

The formatted filename

***String imrcp.system.FilenameFormatter.getExtension ()***

Gets the file extension of the pattern

*Returns*

the file extension of the pattern

***void imrcp.system.FilenameFormatter.H (int nPos) [protected]***

Adds the internal calendar's hour to **m\_sBuf** at the given position.

*Parameters*

<i>nPos</i>	position to insert the date/time string
-------------	---

***void imrcp.system.FilenameFormatter.HHmm (int nPos) [protected]***

Adds the internal calendar's hour and minute to **m\_sBuf** at the given position.

*Parameters*

<i>nPos</i>	position to insert the date/time string
-------------	---

***void imrcp.system.FilenameFormatter.n (int nPos, int nOffset) [protected]***

Adds the given file offset index to **m\_sBuf** at the given position.

*Parameters*

<i>nPos</i>	position to insert the file offset index
<i>nOffset</i>	file offset index

***void imrcp.system.FilenameFormatter.nnn (int nPos, int nOffset) [protected]***

Adds the given file offset index to **m\_sBuf** at the given position.

*Parameters*

<i>nPos</i>	position to insert the file offset index
<i>nOffset</i>	file offset index

***synchronized int imrcp.system.FilenameFormatter.parse (String sFilename, long[] lTimes)***

Parses the given file name and places the start, end, and valid times in the given array in the format [received, end, start]. Filename must be in the IMRCP time dependent file name format.

*Parameters*

<i>sFilename</i>	file name with the format: contrib_extrainfo0_extrainfo1_..._starttime_endtime_validtime.e xt (described above in the class javadoc)
<i>lTimes</i>	array to fill with the times

*Returns*

IMRCP contributor id for the file

**synchronized int imrcp.system.FilenameFormatter.parseRecv (String sFilename, long[] lTimes)**

Parses source file names from National Weather Service collectors, determining the received time of the file and returning its file offset index.

*Parameters*

<i>sFilename</i>	File name to parse
<i>lTimes</i>	array to fill with the received time in position zero.

*Returns*

file offset index of the forecast file.

**synchronized void imrcp.system.FilenameFormatter.parseTile (String sFilename, int[] nTile)**

Parses the given file name of a data file in a SpatialFileCache to determine the map tile indices of the file and places them in *nTile*

*Parameters*

<i>sFilename</i>	file name of a data file in a SpatialFileCache, the x tile index must be in the first "%S" position and the y tile index must be in the second "%S" position.
<i>nTile</i>	array to be filled in the format [x map tile index, y map tile index]

**void imrcp.system.FilenameFormatter.S (int nPos, String sString) [protected]**

Adds the given string to *m\_sBuf* at the given position.

*Parameters*

<i>nPos</i>	position to insert the string
<i>sString</i>	the string to add

**void imrcp.system.FilenameFormatter.setHH (int nPos) [protected]**

Sets the internal calendar's hour from the given position in *m\_sBuf*

*Parameters*

<i>nPos</i>	position to start parsing
-------------	---------------------------

**void imrcp.system.FilenameFormatter.setHHmm (int nPos) [protected]**

Sets the internal calendar's hour and minute from the given position in *m\_sBuf*

*Parameters*

<i>nPos</i>	position to start parsing
-------------	---------------------------

**void imrcp.system.FilenameFormatter.setHHmmss (int nPos) [protected]**

Sets the internal calendar's hour, minute, and second from the given position in *m\_sBuf*

*Parameters*

<i>nPos</i>	position to start parsing
-------------	---------------------------

***void imrcp.system.FilenameFormatter.setyyyyMMdd (int nPos) [protected]***

Sets the internal calendar's year month and day from the given position in **m\_sBuf**

*Parameters*

<i>nPos</i>	position to start parsing
-------------	---------------------------

***void imrcp.system.FilenameFormatter.y (int nPos) [protected]***

Adds the internal calendar's year to **m\_sBuf** at the given position.

*Parameters*

<i>nPos</i>	position to insert the date/time string
-------------	---

***void imrcp.system.FilenameFormatter.yM (int nPos) [protected]***

Adds the internal calendar's year and month to **m\_sBuf** at the given position.

*Parameters*

<i>nPos</i>	position to insert the date/time string
-------------	---

***void imrcp.system.FilenameFormatter.yMd (int nPos) [protected]***

Adds the internal calendar's year, month, and day to **m\_sBuf** at the given position.

*Parameters*

<i>nPos</i>	position to insert the date/time string
-------------	---

***void imrcp.system.FilenameFormatter.yMdH (int nPos) [protected]***

Adds the internal calendar's year, month, day, and hour to **m\_sBuf** at the given position.

*Parameters*

<i>nPos</i>	position to insert the date/time string
-------------	---

***void imrcp.system.FilenameFormatter.yMdHm (int nPos) [protected]***

Adds the internal calendar's year, month, day, hour, and minute to **m\_sBuf** at the given position.

*Parameters*

<i>nPos</i>	position to insert the date/time string
-------------	---

---

*The documentation for this class was generated from the following file:*

- system/FilenameFormatter.java
-

## imrcp.system.FileUtil Class Reference

### Static Public Attributes

- static FileAttribute[] **DIRPERS**
  - static FileAttribute[] **FILEPERS**
  - static Set< StandardOpenOption > **APPENDTO** = EnumSet.of(StandardOpenOption.WRITE, StandardOpenOption.CREATE, StandardOpenOption.APPEND)
  - static Set< StandardOpenOption > **WRITE** = EnumSet.of(StandardOpenOption.WRITE, StandardOpenOption.CREATE, StandardOpenOption.TRUNCATE\_EXISTING)
  - static OpenOption[] **WRITEOPTS** = new OpenOption[]{ StandardOpenOption.WRITE, StandardOpenOption.CREATE, StandardOpenOption.TRUNCATE\_EXISTING }
  - static OpenOption[] **APPENDOPTS** = new OpenOption[]{ StandardOpenOption.WRITE, StandardOpenOption.CREATE, StandardOpenOption.APPEND }
- 

### Detailed Description

Contains convenience variables for common operations using the java nio package.

### Author

Federal Highway Administration

---

### Member Data Documentation

*OpenOption [] imrcp.system.FileUtil.APPENDOPTS = new  
OpenOption[]{StandardOpenOption.WRITE, StandardOpenOption.CREATE,  
StandardOpenOption.APPEND} [static]*

Array of options used to open input streams that append to a new or existing file

*Set<StandardOpenOption> imrcp.system.FileUtil.APPENDTO =  
EnumSet.of(StandardOpenOption.WRITE, StandardOpenOption.CREATE,  
StandardOpenOption.APPEND) [static]*

Set of options used to open input streams that append to a new or existing file

*FileAttribute [] imrcp.system.FileUtil.DIRPERS [static]*

```
Initial value:= new
FileAttribute[] {PosixFilePermissions.asFileAttribute(EnumSet.of(PosixFilePermissions.OWNER_READ,
    PosixFilePermission.OWNER_EXECUTE, PosixFilePermission.OWNER_WRITE,
    PosixFilePermission.GROUP_EXECUTE, PosixFilePermission.GROUP_READ,
    PosixFilePermission.OTHERS_READ, PosixFilePermission.OTHERS_EXECUTE))}
```

File permissions used for creating directories

*FileAttribute [] imrcp.system.FileUtil.FILEPERS [static]*

```
Initial value:= new
FileAttribute[] {PosixFilePermissions.asFileAttribute(EnumSet.of(PosixFilePermissions.OWNER_READ,
    PosixFilePermission.OWNER_WRITE,
    PosixFilePermission.GROUP_READ, PosixFilePermission.OTHERS_READ))}
```

File permissions used for creating regular files

*Set<StandardOpenOption> imrcp.system.FileUtil.WRITE =  
EnumSet.of(StandardOpenOption.WRITE, StandardOpenOption.CREATE,  
StandardOpenOption.TRUNCATE\_EXISTING) [static]*

Set of options used to open input streams that write to a new file or replace an existing file

```
OpenOption [] imrcp.system.FileUtil.WRITEOPTS = new
OpenOption[]{StandardOpenOption.WRITE, StandardOpenOption.CREATE,
StandardOpenOption.TRUNCATE_EXISTING} [static]
```

Array of options used to open input streams that write to a new file or replace an existing file

---

*The documentation for this class was generated from the following file:*

- system/FileUtil.java
- 

## imrcp.store.FileWrapper Class Reference

### Public Member Functions

- abstract void **load** (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId) throws Exception
- abstract void **cleanup** (boolean bDelete)
- void **deleteFile** (File oFile)
- void **setTimes** (long lValid, long lStart, long lEnd)

### Public Attributes

- long **m\_lStartTime**
- long **m\_lEndTime**
- long **m\_lValidTime**
- long **m\_llLastUsed** = System.currentTimeMillis()
- String **m\_sFilename**
- int[] **m\_nObsTypes**
- int **m\_nContribId**
- int **m\_nFormatIndex** = 0

### Static Public Attributes

- static final HashMap< Integer, Integer > **FCSTMINMAP**

### Protected Attributes

- Logger **m\_oLogger** = LogManager.getLogger(getClass())
- 

### Detailed Description

Base class for files that are loaded into memory by **imrcp.store.FileCache** (stores).

### Author

Federal Highway Administration

---

### Member Function Documentation

**abstract void imrcp.store.FileWrapper.cleanup (boolean bDelete) [abstract]**

Called when a **FileWrapper** is removed from the cache in memory to clean up resources if necessary.

#### Parameters

<i>bDelete</i>	flag used to indicate if index files created when the file is loaded into memory or the file itself should be deleted or not
Reimplemented in <a href="#">imrcp.store.AHPSWrapper</a> (p.81), <a href="#">imrcp.store.CapWrapper</a> (p.111), <a href="#">imrcp.store.CsvWrapper</a> (p.125), <a href="#">imrcp.store.GribWrapper</a> (p.252), <a href="#">imrcp.store.MetroWrapper</a> (p.342), <a href="#">imrcp.store.NcfWrapper</a> (p.374), <a href="#">imrcp.store.NHCWrapper</a> (p.395), <a href="#">imrcp.store.SpatialFileWrapper</a> (p.501), <a href="#">imrcp.store.TrafficSpeedStoreWrapper</a> (p.539), <a href="#">imrcp.web.tiles.TileWrapper</a> (p.532), and <a href="#">imrcp.store.DataObsWrapper</a> (p.131).	

*void imrcp.store.FileWrapper.deleteFile (File oFile)*

Attempts to delete the given File. Right now the implementation is commented out so files are not accidentally deleted

#### Parameters

<i>oFile</i>	File to delete
--------------	----------------

Reimplemented in [imrcp.store.GribWrapper](#) (p.252), [imrcp.store.TrafficEventCsv](#) (p.537), [imrcp.store.TrafficSpeedStoreWrapper](#) (p.540), and [imrcp.store.WxDECsv](#) (p.568).

*abstract void imrcp.store.FileWrapper.load (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId) throws Exception [abstract]*

Parses and loads observations from the given file into memory.

#### Parameters

<i>lStartTime</i>	time in milliseconds since Epoch that the file starts having observations
<i>lEndTime</i>	time in milliseconds since Epoch that the file stops having observations
<i>lValidTime</i>	time in milliseconds since Epoch that the file starts being valid (usually the time it is received)
<i>sFilename</i>	path of the file being loaded
<i>nContribId</i>	IMRCP contributor Id which is a computed by converting an up to a 6 character alphanumeric string using base 36.

#### Exceptions

<i>Exception</i>	
Reimplemented in <a href="#">imrcp.store.AHPSWrapper</a> (p.81), <a href="#">imrcp.store.AlertsCsvWrapper</a> (p.85), <a href="#">imrcp.store.CapWrapper</a> (p.111), <a href="#">imrcp.store.CsvWrapper</a> (p.125), <a href="#">imrcp.store.DataObsWrapper</a> (p.132), <a href="#">imrcp.store.GeotabWrapper</a> (p.232), <a href="#">imrcp.store.GribWrapper</a> (p.254), <a href="#">imrcp.store.MLPCsv</a> (p.352), <a href="#">imrcp.store.NcfWrapper</a> (p.376), <a href="#">imrcp.store.NHCWrapper</a> (p.395), <a href="#">imrcp.store.SpatialFileWrapper</a> (p.501), <a href="#">imrcp.store.TrafficEventCsv</a> (p.537), <a href="#">imrcp.store.TrafficSpeedStoreWrapper</a> (p.540), <a href="#">imrcp.store.WxDECsv</a> (p.568), and <a href="#">imrcp.web.tiles.TileWrapper</a> (p.532).	

*void imrcp.store.FileWrapper.setTimes (long lValid, long lStart, long lEnd)*

Sets the start, end, and valid times of the **FileWrapper** to the given values

#### Parameters

<i>lValid</i>	Time in milliseconds since Epoch that the file starts being valid
<i>lStart</i>	Time in milliseconds since Epoch of the earliest observation or forecast contained in this file

<code>lEnd</code>	Time in milliseconds since Epoch of the latest observation or forecast contained in this file
-------------------	---

---

## Member Data Documentation

`final HashMap<Integer, Integer> imrcp.store.FileWrapper.FCSTMINMAP [static]`

Static map object that maps IMRCP contributor Ids to a default time in milliseconds of how long forecasts last for that contributor

`long imrcp.store.FileWrapper.m_lEndTime`

Time in milliseconds since Epoch of the latest observation or forecast contained in this file

`long imrcp.store.FileWrapper.m_lLastUsed = System.currentTimeMillis()`

Time in milliseconds since Epoch that the file was last accessed/used

`long imrcp.store.FileWrapper.m_lStartTime`

Time in milliseconds since Epoch of the earliest observation or forecast contained in this file

`long imrcp.store.FileWrapper.m_lValidTime`

Time in milliseconds since Epoch that the file starts being valid, usually when the file is downloaded/created

`int imrcp.store.FileWrapper.m_nContribId`

IMRCP contributor Id which is computed by converting an up to a 6 character alphanumeric string using base 36.

`int imrcp.store.FileWrapper.m_nFormatIndex = 0`

Index used for this file in a store's `imrcp.store.FileCache#m_oFormatters`

`int [] imrcp.store.FileWrapper.m_nObsTypes`

Contains the IMRCP observation type ids that this file provides

`Logger imrcp.store.FileWrapper.m_oLogger = LogManager.getLogger(getClass()) [protected]`

Logger object

`String imrcp.store.FileWrapper.m_sFilename`

Absolute path of the file

---

*The documentation for this class was generated from the following file:*

- `store/FileWrapper.java`

---



---

## imrcp.store.FloatObsEntryData Class Reference

### Public Member Functions

- `double getValue (int nHz, int nVrt)`
- `void setTimeDim (int nIndex)`

## Additional Inherited Members

---

### Detailed Description

An **EntryData** for IMRCP gridded binary observation files with values that can be stored as floats.

### Author

Federal Highway Administration

---

### Member Function Documentation

#### `double imrcp.store.FloatObsEntryData.getValue (int nHrz, int nVrt)`

Gets the value of the grid at the given x and y coordinates

##### Parameters

<code>nHrz</code>	x coordinate
<code>nVrt</code>	y coordinate

##### Returns

value of the grid at the given x and y coordinates, if the coordinates are out of range,  
`Double.NaN` is returned

Reimplemented from **imrcp.store.EntryData** (*p.202*).

#### `void imrcp.store.FloatObsEntryData.setTimeDim (int nIndex)`

Entry datas of this type do nothing have multiple time dimensions so does nothing.

Reimplemented from **imrcp.store.EntryData** (*p.203*).

---

*The documentation for this class was generated from the following file:*

- `store/FloatObsEntryData.java`
- 

## imrcp.store.FloodMapping Class Reference

### Static Public Attributes

- static final Comparator<**FloodMapping**> **AHPSCOMP** = (**FloodMapping** o1, **FloodMapping** o2) -> o1.m\_sAHPSId.compareTo(o2.m\_sAHPSId)
  - static final Comparator<**FloodMapping**> **USGSOMP** = (**FloodMapping** o1, **FloodMapping** o2) -> o1.m\_sUSGSId.compareTo(o2.m\_sUSGSId)
- 

### Detailed Description

Maps Advanced Hydrologic Prediction Service (AHPS) identifiers to United States Geological Survey (USGS) identifiers and contains information used to compute the flood stage at the given sensor.

*Author*

Federal Highway Administration

---

**Member Data Documentation**

*final Comparator<FloodMapping> imrcp.store.FloodMapping.AHPSCOMP =  
(FloodMapping o1, FloodMapping o2) ->  
o1.m\_sAHPSId.compareTo(o2.m\_sAHPSId) [static]*

Compares **FloodMapping**s by AHPS id

*final Comparator<FloodMapping> imrcp.store.FloodMapping.USGSCOMP =  
(FloodMapping o1, FloodMapping o2) ->  
o1.m\_sUSGSId.compareTo(o2.m\_sUSGSId) [static]*

Compares **FloodMapping**s by USGS id

---

*The documentation for this class was generated from the following file:*

- store/FloodMapping.java
- 

**imrcp.collect.FloodStageMetadata Class Reference**

**Public Member Functions**

- **FloodStageMetadata (DbfResultSet oDbf)** throws SQLException
- double **getStageValue** (double dStageLevel)

**Public Attributes**

- double **m\_dAction**
  - double **m\_dFlood**
  - double **m\_dModerate**
  - double **m\_dMajor**
- 

**Detailed Description**

This class stores metadata for **AHPS** (Advanced Hydrologic Prediction Services) flood stations

*Author*

Federal Highway Administration

---

**Constructor & Destructor Documentation**

*imrcp.collect.FloodStageMetadata.FloodStageMetadata (DbfResultSet oDbf) throws  
SQLException*

Parses a record of an **AHPS** .dbf file to obtain the different flood stage values

*Parameters*

<i>oDbf</i>	DbfResultSet object that has already read the desired line of the .dbf file
-------------	---

*Exceptions*

<i>SQLException</i>
---------------------

---

**Member Function Documentation**

*double imrcp.collect.FloodStageMetadata.getStageValue (double dStageLevel)*

Compares a value to the different flood stages and returns the corresponding flood stage enumeration

*Parameters*

<i>dStageLevel</i>	Stage level to compare
--------------------	------------------------

*Returns*

Flood stage enumeration from `ObsType#lookup(int, java.lang.String)`

---

**Member Data Documentation**

*double imrcp.collect.FloodStageMetadata.m\_dAction*

Stage level for action

*double imrcp.collect.FloodStageMetadata.m\_dFlood*

Stage level for flood

*double imrcp.collect.FloodStageMetadata.m\_dMajor*

Stage level for major flood

*double imrcp.collect.FloodStageMetadata.m\_dModerate*

Stage level for moderate flood

---

*The documentation for this class was generated from the following file:*

- collect/FloodStageMetadata.java

---

---

**imrcp.web.ReportSubscription.Format Enum Reference**

**Public Attributes**

- CSV

---

**Detailed Description**

Enumeration for output formats

---

## Member Data Documentation

### *imrcp.web.ReportSubscription.Format.CSV*

Comma Separated Values

---

*The documentation for this enum was generated from the following file:*

- web/ReportSubscription.java
- 

## imrcp.collect.Geotab Class Reference

### Public Member Functions

- void **reset** ()
- boolean **start** () throws Exception
- void **execute** ()
- void **login** ()
- void **getFeeds** ()

### Static Public Attributes

- static HashMap< String, String > **m\_oDeviceLookup** = new HashMap()
- static final Object **LOCK** = new Object()

### Additional Inherited Members

---

#### Detailed Description

Generic collector for the **Geotab** system

#### Author

Federal Highway Administration

---

## Member Function Documentation

### *void imrcp.collect.Geotab.execute ()*

Wrapper for **Geotab#getFeeds** ()

Reimplemented from **imrcp.system.BaseBlock** (p.95).

### *void imrcp.collect.Geotab.getFeeds ()*

Uses **Geotab**'s multicall api to collect data from the LogRecord and StatusData feeds

### *void imrcp.collect.Geotab.login ()*

Logs into the **Geotab** system using the configured database, user name, and password. The sessionId is store in **Geotab#m\_sSessionId** so multiple calls of the **Geotab#getFeeds** () can be made without having to login in everytime. If an error occurs **Geotab#m\_sSessionId** is set to null.

### `void imrcp.collect.Geotab.reset ()`

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.collect.Collector** (p.114).

### `boolean imrcp.collect.Geotab.start () throws Exception`

Attempts to log in the **Geotab** system, download the Device metadata, and sets a schedule to execute on a fixed interval.

#### *Returns*

true if no Exceptions are thrown

#### *Exceptions*

<i>Exception</i>
------------------

Reimplemented from **imrcp.system.BaseBlock** (p.99).

---

## Member Data Documentation

### `final Object imrcp.collect.Geotab.LOCK = new Object() [static]`

Object used to lock threads for async methods

### `HashMap<String, String> imrcp.collect.Geotab.m_oDeviceLookup = new HashMap() [static]`

Maps **Geotab** device Ids to names

---

*The documentation for this class was generated from the following file:*

- collect/Geotab.java
- 

## imrcp.store.GeotabStore Class Reference

### Public Member Functions

- `void getData (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)`
- `void getDataFromFile (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime, CsvWrapper oFile)`

### Protected Member Functions

- `FileWrapper getNewFileWrapper ()`

### Additional Inherited Members

---

#### Detailed Description

**FileCache** that manages CSV files generated from the Geotab data feed

#### *Author*

Federal Highway Administration

---

## Member Function Documentation

`void imrcp.store.GeotabStore.getData (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)`

Determines the files that match the query and then calls `getDataFromFile(imrcp.store.ImrcpResultSet, int, long, long, int, int, int, int, long, imrcp.store.CsvWrapper)` on each of those files

Reimplemented from `imrcp.store.CsvStore` (p.123).

`void imrcp.store.GeotabStore.getDataFromFile (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime, CsvWrapper oFile)`

Iterates through the observations of the file and adds them to the `ImrcpResultSet` if they match the query. If there are multiple observations associated with the same object Id (which is determined by lon/lat in the case of Geotab obs) only the most recent of those observations is added to the `ImrcpResultSet`

Reimplemented from `imrcp.store.CsvStore` (p.124).

`FileWrapper imrcp.store.GeotabStore.getNewFileWrapper () [protected]`

### Returns

a new `GeotabWrapper` with the configured observation types

Reimplemented from `imrcp.store.CsvStore` (p.124).

---

*The documentation for this class was generated from the following file:*

- store/GeotabStore.java

---

## imrcp.store.GeotabWrapper Class Reference

### Public Member Functions

- `GeotabWrapper (int[] nObsTypes)`
- `void load (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId)`  
throws Exception

### Additional Inherited Members

---

### Detailed Description

Contains the logic to parse and create observations for CSV files generated from the Geotab data feed

### Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

`imrcp.store.GeotabWrapper.GeotabWrapper (int[] nObsTypes)`

Wrapper for `CsvWrapper#CsvWrapper (int[])`

### Parameters

<code>nObsTypes</code>	observation types this file contains
------------------------	--------------------------------------

---

## Member Function Documentation

`void imrcp.store.GeotabWrapper.load (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId) throws Exception`

Parses and creates observations from the given file. Geotab Csv files can have lines appended to them while the file is already in memory so each time this method is called `m_ocsvFile` is reopened and skips any lines that have already been read.

Reimplemented from `imrcp.store.CsvWrapper` (*p.125*).

---

*The documentation for this class was generated from the following file:*

- `store/GeotabWrapper.java`

---

## imrcp.geosrv.GeoUtil Class Reference

### Static Public Member Functions

- static void **getIntersection** (double dPx, double dPy, double dEnd1x, double dEnd1y, double dQx, double dQy, double dEnd2x, double dEnd2y, double[] dInter)
- static double **cross** (double dVx, double dVy, double dWx, double dWy)
- static int **floor** (int nValue, int nPrecision)
- static boolean **isInside** (int nX, int nY, int nT, int nR, int nB, int nL, int nTol)
- static int **rightHand** (double dX, double dY, double dX1, double dY1, double dX2, double dY2)
- static boolean **isInside** (double dX, double dY, double dT, double dR, double dB, double dL, double dTol)
- static int **getPerpDist** (int nX, int nY, int nX1, int nY1, int nX2, int nY2, **WaySnapInfo** oSnap)
- static int **getPerpDist** (int nX, int nY, int nX1, int nY1, int nX2, int nY2)
- static double **getPerpDist** (double dX, double dY, double dX1, double dY1, double dX2, double dY2, **WaySnapInfo** oSnap)
- static double **getPerpDist** (double dX, double dY, double dX1, double dY1, double dX2, double dY2)
- static double **snap** (double dX, double dY, double dX1, double dY1, double dX2, double dY2, double[] dSnap)
- static int **toIntDeg** (double dValue)
- static double **fromIntDeg** (int nValue)
- static double **fromIntDeg** (int nValue, int nScale)
- static boolean **isInsidePolygon** (int[] nPolyPoints, double nX, double nY)
- static boolean **isInsideMultiPolygon** (ArrayList<int[]> nRings, int nRingStart, int nHole, int nBoundsStart, double nX, double nY)
- static boolean **isInsideMultiPolygon** (ArrayList<int[]> nRings, int nRingStart, int nHole, int nBoundsStart, int[] nPolyBounds, int[] nPolyLine, int nLineStart, int nLineBoundsStart)

- static boolean **isInsideMultiPolygon** (ArrayList< int[]> nRings, int nRingStart, int nHole, int nBoundsStart, int[] nPolyBounds, **OsmWay** oWay)
- static boolean **isInsidePolygon** (int[] nPoly, double nX, double nY, int nStart)
- static boolean **isInsidePolygon** (int[] nPoly, double nX, double nY, int nStart, int nBoundsStart)
- static boolean **boundingBoxesIntersect** (double dXmin1, double dYmin1, double dXmax1, double dYmax1, double dXmin2, double dYmin2, double dXmax2, double dYmax2)
- static boolean **boundingBoxesIntersect** (int nXmin1, int nYmin1, int nXmax1, int nYmax1, int nXmin2, int nYmin2, int nXmax2, int nYmax2)
- static int **compareTol** (double d1, double d2, double dTol)
- static double **adjustLat** (double dLat)
- static double **adjustLon** (double dLon)
- static double **distanceFromLatLon** (double dLat1, double dLon1, double dLat2, double dLon2)
- static double **angle** (double dX1, double dY1, double dX2, double dY2, double dX3, double dY3)
- static double **round** (double dVal, int nPlaces)
- static double **sqDist** (double dXi, double dYi, double dXj, double dYj)
- static double **distance** (double dXi, double dYi, double dXj, double dYj)
- static double **angle** (double dX1, double dY1, double dX2, double dY2)
- static double **heading** (double dX1, double dY1, double dX2, double dY2)
- static double **hdgDiff** (double dHdg1, double dHdg2)
- static int **quad** (double dAngle)
- static boolean **obsInside** (Area oArea, **Obs** oObs)

### Static Public Attributes

- static final double **EARTH\_RADIUS\_KM** = 6371
  - static final double **PIOVER180** = Math.PI / 180
- 

### Detailed Description

This class contains computational geometry and related methods.

#### Author

Federal Highway Administration

---

### Member Function Documentation

#### *static double imrcp.geosrv.GeoUtil.adjustLat (double dLat) [static]*

Adjusts the latitude to decimal degrees if it is outside of the range of -90 <= dLat <= 90

##### Parameters

<i>dLat</i>	latitude to adjust if needed
-------------	------------------------------

##### Returns

latitude in decimal degrees

#### *static double imrcp.geosrv.GeoUtil.adjustLon (double dLon) [static]*

Adjusts the longitude to decimal degrees if it is outside of the range of -180 < dLon <= 180

##### Parameters

<i>dLon</i>	longitude to adjust if needed
-------------	-------------------------------

##### Returns

longitude in decimal degrees

`static double imrcp.geosrv.GeoUtil.angle (double dX1, double dY1, double dX2, double dY2) [static]`

Determines the measure of the angle defined by using the first point as the vertex of the angle, the second point as a point on the terminal side of the angle and a point one unit to the right on the vertex as a point on the initial side of the angle.

#### Parameters

<i>dX1</i>	x coordinate of point 1 (vertex)
<i>dY1</i>	y coordinate of point 1 (vertex)
<i>dX2</i>	x coordinate of the point 2 (terminal side of angle)
<i>dY2</i>	y coordinate of the point 2 (terminal side of angle)

#### Returns

Measure of the angle defined using the first point as the vertex of the angle, the second point as a point on the terminal side of the angle and a point one unit to the right on the vertex as a point on the initial side of the angle.

`static double imrcp.geosrv.GeoUtil.angle (double dX1, double dY1, double dX2, double dY2, double dX3, double dY3) [static]`

Determines the measure of the angle defined by the 3 points in radians. The second point is the vertex of the angle.

#### Parameters

<i>dX1</i>	x coordinate of point 1
<i>dY1</i>	y coordinate of point 1
<i>dX2</i>	x coordinate of the point 2(vertex)
<i>dY2</i>	y coordinate of the point 2(vertex)
<i>dX3</i>	x coordinate of point 3
<i>dY3</i>	y coordinate of point 3

#### Returns

Measure of the angle defined by the 3 points in radians

`static boolean imrcp.geosrv.GeoUtil.boundingBoxesIntersect (double dXmin1, double dYmin1, double dXmax1, double dYmax1, double dXmin2, double dYmin2, double dXmax2, double dYmax2) [static]`

Determines if the two bounding boxes intersect.

#### Parameters

<i>dXmin1</i>	min x of the first bounding box
<i>dYmin1</i>	min y of the first bounding box
<i>dXmax1</i>	max x of the first bounding box
<i>dYmax1</i>	max y of the first bounding box
<i>dXmin2</i>	min x of the second bounding box
<i>dYmin2</i>	min y of the second bounding box
<i>dXmax2</i>	max x of the second bounding box
<i>dYmax2</i>	max y of the second bounding box

### Returns

true if the two bounding boxes intersect, otherwise false

`static boolean imrcp.geosrv.GeoUtil.boundingBoxesIntersect (int nXmin1, int nYmin1, int nXmax1, int nYmax1, int nXmin2, int nYmin2, int nXmax2, int nYmax2) [static]`

Determines if the two bounding boxes intersect.

### Parameters

<code>nXmin1</code>	min x of the first bounding box
<code>nYmin1</code>	min y of the first bounding box
<code>nXmax1</code>	max x of the first bounding box
<code>nYmax1</code>	max y of the first bounding box
<code>nXmin2</code>	min x of the second bounding box
<code>nYmin2</code>	min y of the second bounding box
<code>nXmax2</code>	max x of the second bounding box
<code>nYmax2</code>	max y of the second bounding box

### Returns

true if the two bounding boxes intersect, otherwise false

`static int imrcp.geosrv.GeoUtil.compareTol (double d1, double d2, double dTol) [static]`

Compares the two doubles with the given tolerance.

### Parameters

<code>d1</code>	first double
<code>d2</code>	second double
<code>dTol</code>	tolerance

### Returns

0 if the doubles are within the tolerance of get other, -1 if the second double is more than the tolerance greater than the first double, 1 if the first double is more than the tolerance greater than the second double.

`static double imrcp.geosrv.GeoUtil.cross (double dVx, double dVy, double dWx, double dWy) [static]`

Gets the magnitude of the vector that is the cross product between two vectors with their z component set to zero.

### Parameters

<code>dVx</code>	x component of vector 1
<code>dVy</code>	y component of vector 1
<code>dWx</code>	x component of vector 2
<code>dWy</code>	y component of vector 2

### Returns

the magnitude of the vector that is the cross product between two vectors with their z component set to zero.

*static double imrcp.geosrv.GeoUtil.distance (double dXi, double dYi, double dXj, double dYj) [static]*

Gets the distance between the given two points

*Parameters*

<i>dXi</i>	x coordinate of the first point
<i>dYi</i>	y coordinate of the first point
<i>dXj</i>	x coordinate of the second point
<i>dYj</i>	y coordinate of the second point

*Returns*

The distance between the points.

*static double imrcp.geosrv.GeoUtil.distanceFromLatLon (double dLat1, double dLon1, double dLat2, double dLon2) [static]*

Gets the distance between the 2 geo-coordinates in km using the Haversine formula.

*Parameters*

<i>dLat1</i>	latitude in decimal degrees of the first point
<i>dLon1</i>	longitude in decimal degrees of the first point
<i>dLat2</i>	latitude in decimal degrees of the second point
<i>dLon2</i>	longitude in decimal degrees of the second point

*Returns*

distance in km between the 2 geo-coordinates.

*static int imrcp.geosrv.GeoUtil.floor (int nValue, int nPrecision) [static]*

Floors the given integer to the given precision.

*Parameters*

<i>nValue</i>	the value to floor
<i>nPrecision</i>	the precision to used to floor

*Returns*

the floored value of the integer based on the precision

*static double imrcp.geosrv.GeoUtil.fromIntDeg (int nValue) [static]*

Converts the given integer decimal degree scaled to 7 decimal places to a double representation in decimal degrees.

*Parameters*

<i>nValue</i>	integer decimal degree scaled to 7 decimal places
---------------	---

*Returns*

Decimal degrees value of the scaled integer value

*static double imrcp.geosrv.GeoUtil.fromIntDeg (int nValue, int nScale) [static]*

Converts the given integer decimal degree scaled to the given power of 10 to a double representation in decimal degrees

#### Parameters

<i>nValue</i>	integer decimal degree scaled to the power of 10
<i>nScale</i>	the power of 10

#### Returns

Decimal degrees value of the scaled integer value

```
static void imrcp.geosrv.GeoUtil.getIntersection (double dPx, double dPy, double  
dEnd1x, double dEnd1y, double dQx, double dQy, double dEnd2x, double dEnd2y,  
double[] dInter) [static]
```

Fills the given array with the point the two line segments intersect. If the two line segments do not intersect, the array is filled with Double.NaN

#### Parameters

<i>dPx</i>	x coordinate of the initial point of the first segment
<i>dPy</i>	y coordinate of the initial point of the first segment
<i>dEnd1x</i>	x coordinate of the terminal point of the first segment
<i>dEnd1y</i>	y coordinate of the terminal point of the first segment
<i>dQx</i>	x coordinate of the initial point of the second segment
<i>dQy</i>	y coordinate of the initial point of the second segment
<i>dEnd2x</i>	x coordinate of the terminal point of the second segment
<i>dEnd2y</i>	y coordinate of the terminal point of the second segment
<i>dInter</i>	array to get filled with the coordinates of the intersection point

```
static double imrcp.geosrv.GeoUtil.getPerpDist (double dX, double dY, double dX1,  
double dY1, double dX2, double dY2) [static]
```

Gets the squared perpendicular distance from the given point to the given directed line segment by attempting to project the point onto the line segment.

#### Parameters

<i>dX</i>	x coordinate of the point
<i>dY</i>	y coordinate of the point
<i>dX1</i>	x coordinate of the initial point of the line segment
<i>dY1</i>	y coordinate of the initial point of the line segment
<i>dX2</i>	x coordinate of the terminal point of the line segment
<i>dY2</i>	y coordinate of the terminal point of the line segment.

#### Returns

The perpendicular distance from the point to the line segment. If the point cannot be projected on to the line segment Double.NaN is returned

```
static double imrcp.geosrv.GeoUtil.getPerpDist (double dX, double dY, double dX1,  
double dY1, double dX2, double dY2, WaySnapInfo oSnap) [static]
```

Gets the squared perpendicular distance from the given point to the given directed line segment by attempting to project the point onto the line segment. Useful intermediate parameters get stored in the given **WaySnapInfo**.

#### Parameters

<i>dX</i>	x coordinate of the point
-----------	---------------------------

<i>dY</i>	y coordinate of the point
<i>dX1</i>	x coordinate of the initial point of the line segment
<i>dY1</i>	y coordinate of the initial point of the line segment
<i>dX2</i>	x coordinate of the terminal point of the line segment
<i>dY2</i>	y coordinate of the terminal point of the line segment.
<i>oSnap</i>	object that stores some useful intermediate parameters of the perpendicular distance algorithm

#### Returns

The perpendicular distance from the point to the line segment. If the point cannot be projected on to the line segment `Double.NaN` is returned

`static int imrcp.geosrv.GeoUtil.getPerpDist (int nX, int nY, int nX1, int nY1, int nX2, int nY2) [static]`

Gets the squared perpendicular distance from the given point to the given directed line segment. Wrapper for the method that accepts doubles: `getPerpDist(double, double, double, double, double)`

#### Parameters

<i>nX</i>	x coordinate of the point
<i>nY</i>	y coordinate of the point
<i>nX1</i>	x coordinate of the initial point of the line segment
<i>nY1</i>	y coordinate of the initial point of the line segment
<i>nX2</i>	x coordinate of the terminal point of the line segment
<i>nY2</i>	y coordinate of the terminal point of the line segment.

#### Returns

The perpendicular distance from the point to the line segment. If the point cannot be projected on to the line segment `Integer.MIN_VALUE` is returned

`static int imrcp.geosrv.GeoUtil.getPerpDist (int nX, int nY, int nX1, int nY1, int nX2, int nY2, WaySnapInfo oSnap) [static]`

Gets the squared perpendicular distance from the given point to the given directed line segment. Wrapper for the method that accepts doubles: `getPerpDist(double, double, double, double, double, double, imrcp.geosrv.WaySnapInfo)`. Useful intermediate parameters get stored in the given `WaySnapInfo`

#### Parameters

<i>nX</i>	x coordinate of the point
<i>nY</i>	y coordinate of the point
<i>nX1</i>	x coordinate of the initial point of the line segment
<i>nY1</i>	y coordinate of the initial point of the line segment
<i>nX2</i>	x coordinate of the terminal point of the line segment
<i>nY2</i>	y coordinate of the terminal point of the line segment.
<i>oSnap</i>	object that stores some useful intermediate parameters of the perpendicular distance algorithm

### Returns

The perpendicular distance from the point to the line segment. If the point cannot be snapped to the line segment Integer.MIN\_VALUE is returned

*static double imrcp.geosrv.GeoUtil.hdgDiff (double dHdg1, double dHdg2) [static]*

Determines the magnitude of the difference between the two headings.

### Parameters

<i>dHdg1</i>	first heading in radians
<i>dHdg2</i>	second heading in radians

### Returns

the magnitude of the difference between the two headings in radians

*static double imrcp.geosrv.GeoUtil.heading (double dX1, double dY1, double dX2, double dY2) [static]*

Determines the heading of the directed line segment defined by the given 2 points.

### Parameters

<i>dX1</i>	x coordinate of the initial point
<i>dY1</i>	y coordinate of the initial point
<i>dX2</i>	x coordinate of the terminal point
<i>dY2</i>	y coordinate of the terminal point

### Returns

The heading of the directed line segment in radians. Range is  $0 \leq \text{rad} < 2\pi$

*static boolean imrcp.geosrv.GeoUtil.isInside (double dX, double dY, double dT, double dR, double dB, double dL, double dTol) [static]*

Determines if the given point is inside the given bounding box with the given tolerance.

### Parameters

<i>dX</i>	x coordinate of the point
<i>dY</i>	y coordinate of the point
<i>dT</i>	top bound of the bounding box
<i>dR</i>	right bound of the bounding box
<i>dB</i>	bottom bound of the bounding box
<i>dL</i>	left bound of the bounding box
<i>dTol</i>	tolerance to expand the bounding box by

### Returns

true if the point is inside (or on the edge) of the bounding box expanded by the tolerance, otherwise false.

*static boolean imrcp.geosrv.GeoUtil.isInside (int nX, int nY, int nT, int nR, int nB, int nL, int nTol) [static]*

Determines if the given point is inside the given bounding box with the given tolerance.

*Parameters*

<i>nX</i>	x coordinate of the point
<i>nY</i>	y coordinate of the point
<i>nT</i>	top bound of the bounding box
<i>nR</i>	right bound of the bounding box
<i>nB</i>	bottom bound of the bounding box
<i>nL</i>	left bound of the bounding box
<i>nTol</i>	tolerance to expand the bounding box by

*Returns*

true if the point is inside (or on the edge) of the bounding box expanded by the tolerance, otherwise false.

*static boolean imrcp.geosrv.GeoUtil.isInsideMultiPolygon (ArrayList<int[]> nRings, int nRingStart, int nHole, int nBoundsStart, double nX, double nY) [static]*

Determines if the given point is inside the given multipolygon defined by the ArrayList of polygon rings.

*Parameters*

<i>nRings</i>	list containing any outer and inner rings defining a multipolygon object. Rings are defined by a growable array which has a flexible format but should be something like [insertion point, hole flag, min x, min y, max x, max y, x0, y0, x1, y1, ... xn, yn, x0, y0]. In that case nHole would be 1 and nBoundsStart would be 2 and nRingStart would be 6.
<i>nRingStart</i>	the index in the ring arrays that the coordinates start at
<i>nHole</i>	the index in the ring arrays that the hole flag is located
<i>nBoundsStart</i>	the index in the ring arrays that the bounding box starts at
<i>nX</i>	x coordinate of the point
<i>nY</i>	y coordinate of the point

*Returns*

true if the point is inside the multipolygon, otherwise false.

*static boolean imrcp.geosrv.GeoUtil.isInsideMultiPolygon (ArrayList<int[]> nRings, int nRingStart, int nHole, int nBoundsStart, int[] nPolyBounds, int[] nPolyLine, int nLineStart, int nLineBoundsStart) [static]*

Determines if the given polyline is inside or intersects the given multipolygon.

*Parameters*

<i>nRings</i>	list containing any outer and inner rings defining a multipolygon object. Rings are defined by a growable array which has a flexible format but should be something like [insertion point, hole flag, min x, min y, max x, max y, x0, y0, x1, y1, ... xn, yn, x0, y0]. In that case nHole would be 1 and nBoundsStart would be 2 and nRingStart would be 6.
<i>nRingStart</i>	the index in the ring arrays that the coordinates start at
<i>nHole</i>	the index in the ring arrays that the hole flag is located
<i>nBoundsStart</i>	the index in the ring arrays that the bounding box starts at

<i>nPolyBounds</i>	bounding box of the entire multipolygon [min x, min y, max x, max y]
<i>nPolyLine</i>	growable array defining the polyline to test which has a flexible format but should be something like [insertion point, min x, min y, max x, max y, x0, y0, x1, y1, ... xn, yn]. In that case nLineStart would be 5 and nLineBoundsStart would be 1
<i>nLineStart</i>	the index in the polyline array that the coordinates start at
<i>nLineBoundsStart</i>	the index in the polyline array that the bounding box starts at

#### Returns

true if the polyline is inside or intersect the multipolygon

```
static boolean imrcp.geosrv.GeoUtil.isInsideMultiPolygon (ArrayList<int[]> nRings, int nRingStart, int nHole, int nBoundsStart, int[] nPolyBounds, OsmWay oWay) [static]
```

Determines if the given roadway segment is inside or intersects the given multipolygon.

#### Parameters

<i>nRings</i>	list containing any outer and inner rings defining a multipolygon object. Rings are defined by a growable array which has a flexible format but should be something like [insertion point, hole flag, min x, min y, max x, max y, x0, y0, x1, y1, ... xn, yn, x0, y0]. In that case nHole would be 1 and nBoundsStart would be 2 and nRingStart would be 6.
<i>nRingStart</i>	the index in the ring arrays that the coordinates start at
<i>nHole</i>	the index in the ring arrays that the hole flag is located
<i>nBoundsStart</i>	the index in the ring arrays that the bounding box starts at
<i>nPolyBounds</i>	bounding box of the entire multipolygon [min x, min y, max x, max y]
<i>oWay</i>	the roadway segment to test

#### Returns

true if the roadway segment is inside or intersect the multipolygon

```
static boolean imrcp.geosrv.GeoUtil.isInsidePolygon (int[] nPoly, double nX, double nY, int nStart) [static]
```

Wrapper for `isInsidePolygon(int[], double, double, int, int)` for a polygon array that does not have the bounding box include so 0 is passed for nBoundsStart.

#### Parameters

<i>nPoly</i>	growable array defining the closed polygon which has a flexible format but should be something like [insertion point, x0, y0, x1, y1, ... xn, yn, x0, y0]
<i>nX</i>	x coordinate of the point
<i>nY</i>	y coordinate of the point
<i>nStart</i>	index in the polygon array that the coordinates start at

#### Returns

true if the point is inside the polygon, otherwise false

**static boolean imrcp.geosrv.GeoUtil.isInsidePolygon (int[] nPoly, double nX, double nY, int nStart, int nBoundsStart) [static]**

Determines if the given point is inside the given closed polygon defined by the array. This is an implementation of the Ray Casting Algorithm.

#### Parameters

<i>nPoly</i>	growable array defining the closed polygon which has a flexible format but should be something like [insertion point, x0, y0, x1, y1, ... xn, yn, x0, y0]
<i>nX</i>	x coordinate of the point
<i>nY</i>	y coordinate of the point
<i>nStart</i>	index in the polygon array that the coordinates start at
<i>nBoundsStart</i>	if the growable array contains the bounding box this is the index in the array that the bounding box starts at, otherwise should be 0

#### Returns

true if the point is inside the polygon, otherwise false

**static boolean imrcp.geosrv.GeoUtil.isInsidePolygon (int[] nPolyPoints, double nX, double nY) [static]**

Determines if the given point is inside the given closed polygon defined by the array. This is an implementation of the Ray Casting Algorithm.

#### Parameters

<i>nPolyPoints</i>	closed polygon in format [y0, x0, y1, x1, ... yn, xn, y0, x0]
<i>nX</i>	x coordinate of the point
<i>nY</i>	y coordinate of the point

#### Returns

true if the point is inside the polygon, otherwise false

**static boolean imrcp.geosrv.GeoUtil.obsInside (Area oArea, Obs oObs) [static]**

Determines if the spatial extents of the given observation are inside or intersect the given Area

#### Parameters

<i>oArea</i>	Area to test
<i>oObs</i>	Obs to test

#### Returns

true if the Obs is inside or intersects the Area, otherwise false

**static int imrcp.geosrv.GeoUtil.quad (double dAngle) [static]**

Determines the quadrant the given angle, in radians, is in.

*Parameters*

<i>dAngle</i>	angle in radians, $0 \leq \text{angle} \leq \pi$
---------------	--

*Returns*

the quadrant the angle is in. If the angle is on the x or y axis other values are returned namely:  
 $3\pi/2 = -3$   $\pi = -2$   $\pi/2 = -1$   $0 = 0$

**static int imrcp.geosrv.GeoUtil.rightHand (double dX, double dY, double dX1, double dY1, double dX2, double dY2) [static]**

Calculates the right hand rule value of the point in regards to the given directed line segment.

*Parameters*

<i>dX</i>	x coordinate of the point
<i>dY</i>	y coordinate of the point
<i>dX1</i>	x coordinate of the initial point of the line segment
<i>dY1</i>	y coordinate of the initial point of the line segment
<i>dX2</i>	x coordinate of the terminal point of the line segment
<i>dY2</i>	y coordinate of the terminal point of the line segment.

*Returns*

-1, 0, or 1. -1 means the point is on the right side of the line segment, 0 means the point is on the line defined by the line segment, and 1 means the point is on the left side of the line segment

**static double imrcp.geosrv.GeoUtil.round (double dVal, int nPlaces) [static]**

Rounds the given value to the nearest given amount of decimal places.

*Parameters*

<i>dVal</i>	value to round
<i>nPlaces</i>	number of decimal places to round to

*Returns*

the rounded value

**static double imrcp.geosrv.GeoUtil.snap (double dX, double dY, double dX1, double dY1, double dX2, double dY2, double[] dSnap) [static]**

Gets the squared perpendicular distance from the given point to the given directed line segment by attempting to project the point onto the line segment. The projected point gets stored in dSnap.

*Parameters*

<i>dX</i>	x coordinate of the point
<i>dY</i>	y coordinate of the point
<i>dX1</i>	x coordinate of the initial point of the line segment
<i>dY1</i>	y coordinate of the initial point of the line segment
<i>dX2</i>	x coordinate of the terminal point of the line segment
<i>dY2</i>	y coordinate of the terminal point of the line segment.

<i>dSnap</i>	array that gets filled with the projected point on the line segment in the format [projected x coordinate, projected y coordinate]. If the point cannot be projected both coordinates are set to Double.NaN
--------------	---

#### Returns

The perpendicular distance from the point to the line segment. If the point cannot be projected on to the line segment Double.NaN is returned

**static double imrcp.geosrv.GeoUtil.sqDist (double dXi, double dYi, double dXj, double dYj) [static]**

Gets the squared distance between the given two points

#### Parameters

<i>dXi</i>	x coordinate of the first point
<i>dYi</i>	y coordinate of the first point
<i>dXj</i>	x coordinate of the second point
<i>dYj</i>	y coordinate of the second point

#### Returns

The squared distance between the points.

**static int imrcp.geosrv.GeoUtil.toIntDeg (double dValue) [static]**

Converts the given decimal degree coordinate to an integer scaled to 7 decimal places. This is used to store geo-coordinates as integers.

#### Parameters

<i>dValue</i>	Decimal degree coordinate to convert
---------------	--------------------------------------

#### Returns

the decimal degree coordinate as an integer scaled to 7 decimal places

## Member Data Documentation

**final double imrcp.geosrv.GeoUtil.EARTH\_RADIUS\_KM = 6371 [static]**

Approximate radius of the earth in km

**final double imrcp.geosrv.GeoUtil.PIOVER180 = Math.PI / 180 [static]**

Pi divided by 180

*The documentation for this class was generated from the following file:*

- geosrv/GeoUtil.java

## imrcp.collect.GFS Class Reference

### Public Member Functions

- void **reset** ()
- boolean **start** () throws Exception
- String **getDestFilename** (long lFileTime, int nPageIndex, String... sStrings)
- void **execute** ()
- void **collect** (ArrayList< String > oFiles)
- String **filter** (String sFilename)

### Additional Inherited Members

---

#### Detailed Description

Collects files from **GFS** (Global Forecast System) and saves a filtered version of the .grb2 containing only the configurable observation types.

#### Author

Federal Highway Administration

---

#### Member Function Documentation

##### `void imrcp.collect.GFS.collect (ArrayList< String > oFiles)`

Attempts to download the list of files from the **GFS** servers. Once a file is successfully downloaded `imrcp.collect.GFS#filter(java.lang.String)` is called on the file. Files that fail to download are saved and attempted to download later.

###### Parameters

<code>oFiles</code>	The list contains filenames that come in sets of two, the first being the source file name, the second the destination file name
---------------------	--

##### `void imrcp.collect.GFS.execute ()`

Calls the `imrcp.collect.GFS#collect(java.util.ArrayList)` after determining the files that need to be downloaded.

Reimplemented from `imrcp.system.BaseBlock` (p.95).

##### `String imrcp.collect.GFS.filter (String sFilename)`

Opens the grb2 file and filters out undesired observation types and saves the resulting file to disk.

###### Parameters

<code>sFilename</code>	the grb2 file to filter
------------------------	-------------------------

###### Returns

the filename of the filtered file

##### `String imrcp.collect.GFS.getDestFilename (long lFileTime, int nPageIndex, String... sStrings)`

#### Parameters

<i>lFileTime</i>	
<i>nPageIndex</i>	
<i>sStrings</i>	

#### Returns

Reimplemented from **imrcp.collect.Collector** (p.114).

#### `void imrcp.collect.GFS.reset ()`

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.collect.Collector** (p.114).

#### `boolean imrcp.collect.GFS.start () throws Exception`

Determines the last time a set of files should have been downloaded and attempts to download those files and then sets a schedule to execute on a fixed interval. If **GFS#m\_oCatchup** is not empty, the schedule is not set and only the files for the configured dates are attempted to be downloaded

#### Returns

true

#### Exceptions

<i>Exception</i>	
------------------	--

Reimplemented from **imrcp.system.BaseBlock** (p.99).

---

The documentation for this class was generated from the following file:

- collect/GFS.java
- 

## imrcp.store.GribEntryData Class Reference

### Public Member Functions

- **GribEntryData** (*Grid oGrid, int nContrib*) throws IOException
- double **getValue** (*int nHz, int nVrt*)
- void **setTimeDim** (*int nIndex*)

### Public Attributes

- **DataRep m\_oDataRep**

### Additional Inherited Members

---

#### Detailed Description

An **EntryData** for .grb2 files. As of now only the .png compression for .grb2 is implemented since this is used by the MRMS files.

## Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

*imrcp.store.GribEntryData.GribEntryData (Grid oGrid, int nContrib) throws IOException*

Constructs a new **GribEntryData** from the given Grid.

### Parameters

<i>oGrid</i>	grb2 Grid object
<i>nContrib</i>	IMRCP contributor Id

### Exceptions

<i>IOException</i>
--------------------

---

## Member Function Documentation

*double imrcp.store.GribEntryData.getValue (int nHz, int nVrt)*

Gets the value of the grid at the given x and y coordinates

### Parameters

<i>nHz</i>	x coordinate
<i>nVrt</i>	y coordinate

### Returns

value of the grid at the given x and y coordinates, if the coordinates are out of range,  
Double.NaN is returned

Reimplemented from **imrcp.store.EntryData** (p.202).

*void imrcp.store.GribEntryData.setTimeDim (int nIndex)*

Entry datas of this type do nothing have multiple time dimensions so does nothing.

Reimplemented from **imrcp.store.EntryData** (p.203).

---

## Member Data Documentation

*DataRep imrcp.store.GribEntryData.m\_oDataRep*

DataRep object that corresponds to Section 5 of a .grb2 file

---

*The documentation for this class was generated from the following file:*

- store/GribEntryData.java
-

## imrcp.store.grib.GribParameter Class Reference

### Public Member Functions

- **GribParameter** (int nImrcpObsType, int nDiscipline, int nCategory, int nNumber)
- int **compareTo** (**GribParameter** o)

### Public Attributes

- final int **m\_nDiscipline**
- final int **m\_nCategory**
- final int **m\_nNumber**
- final int **m\_nImrcpObsType**

---

### Detailed Description

GRIB2 observation types are defined by 3 values, their discipline, category, and number. Disciplines are defined in GRIB2 - TABLE 0.0, Categories are defined in GRIB2 - CODE TABLE 4.1, and numbers are defined in GRIB2 - TABLE 4.2-d-c where d is the discipline and c is the category. See [https://www.nco.ncep.noaa.gov/pmb/docs/grib2/grib2\\_doc/](https://www.nco.ncep.noaa.gov/pmb/docs/grib2/grib2_doc/)

### Author

Federal Highway Administration

---

### Constructor & Destructor Documentation

*imrcp.store.grib.GribParameter.GribParameter (int nImrcpObsType, int nDiscipline, int nCategory, int nNumber)*

Constructs a new **GribParameter** with the given parameters.

#### Parameters

<i>nImrcpObsType</i>	IMRCP observation type id this <b>GribParameter</b> represents
<i>nDiscipline</i>	GRIB2 discipline
<i>nCategory</i>	GRIB2 category
<i>nNumber</i>	GRIB2 number

---

### Member Function Documentation

*int imrcp.store.grib.GribParameter.compareTo (GribParameter o)*

Compares GribParameters by discipline, then category, then number

---

### Member Data Documentation

*final int imrcp.store.grib.GribParameter.m\_nCategory*

GRIB2 Parameter Category

*final int imrcp.store.grib.GribParameter.m\_nDiscipline*

GRIB2 Parameter Discipline

*final int imrcp.store.grib.GribParameter.m\_nImrcpObsType*

IMRCP observation type id this **GribParameter** represents

*final int imrcp.store.grib.GribParameter.m\_nNumber*

GRIB2 Parameter Number

---

*The documentation for this class was generated from the following file:*

- store/grib/GribParameter.java
- 

## imrcp.store.GribStore Class Reference

### Public Member Functions

- void **getData** (**ImrcpResultSet** oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)
- void **reset** ()

### Protected Member Functions

- **GriddedFileWrapper** **getNewFileWrapper** ()

### Additional Inherited Members

---

#### Detailed Description

**FileCache** that manages .grb2 files.

#### Author

Federal Highway Administration

---

#### Member Function Documentation

**void imrcp.store.GribStore.getData (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)**

Determines the files that match the query and then calls **GribWrapper#getData(int, long, int, int, int, int, imrcp.store.ImrcpResultSet)** for each of those files.

Reimplemented from **imrcp.system.BaseBlock** (p.96).

**GriddedFileWrapper imrcp.store.GribStore.getNewFileWrapper () [protected]**

#### Returns

a new **GribWrapper** with the configured observation types and GribParameters

Reimplemented from **imrcp.store.FileCache** (p.212).

`void imrcp.store.GribStore.reset ()`

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.store.FileCache** (*p.214*).

---

*The documentation for this class was generated from the following file:*

- `store/GribStore.java`
- 

## [imrcp.web.tiles.GribTileCache Class Reference](#)

### [Public Member Functions](#)

- `void reset ()`

### [Protected Member Functions](#)

- `GriddedFileWrapper getDataWrapper (int nFormatIndex)`

### [Additional Inherited Members](#)

---

### [Detailed Description](#)

**TileCache** implementation that uses **GribWrapper** s

#### *Author*

Federal Highway Administration

---

### [Member Function Documentation](#)

`GriddedFileWrapper imrcp.web.tiles.GribTileCache.getDataWrapper (int nFormatIndex) [protected]`

#### *Returns*

a new `imrcp.store.GribWrapper` with the configured observations types and grib parameters

Reimplemented from **imrcp.web.tiles.TileCache** (*p.521*).

`void imrcp.web.tiles.GribTileCache.reset ()`

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.web.tiles.TileCache** (*p.522*).

---

*The documentation for this class was generated from the following file:*

- `web/tiles/GribTileCache.java`

---

## [imrcp.store.GribWrapper Class Reference](#)

### Public Member Functions

- **GribWrapper** (int[] nObsTypes, **GribParameter**[] oParameters)
- void **deleteFile** (File oFile)
- void **getGrids** (ArrayList<**Grid**> oGrids) throws Exception
- void **load** (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId) throws Exception
- void **cleanup** (boolean bDelete)
- double **getReading** (int nObsType, long lTimestamp, int nLat, int nLon, Date oTimeRecv)
- void **getData** (int nObsTypeId, long lTimestamp, int nLat1, int nLon1, int nLat2, int nLon2, **ImrcpResultSet** oObs)
- void **getIndices** (int nLon, int nLat, int[] nIndices)
- double **getReading** (int nObsType, long lTimestamp, int[] nIndices)

### Static Public Member Functions

- static long **readHeader** (DataInputStream oIn, int[] nDiscipline) throws IOException
- static **Projection** **getProj** (DataInputStream oIn, int nSecLen) throws IOException
- static int **readSection** (DataInputStream oIn, byte[] ySection) throws Exception
- static **GribParameter** **getParameter** (DataInputStream oIn, int nSecLen, int[] nDiscipline, **GribParameter**[] oParameters) throws IOException
- static **DataRep** **getLayout** (DataInputStream oIn, int nSecLen) throws IOException

### Public Attributes

- int[] **m\_nDiscipline** = new int[]{-1}
- long **m\_lFileTime**

### Static Public Attributes

- static int **GRIBMARKER** = 1196575042
- static int **ENDMARKER** = 926365495

### Additional Inherited Members

---

#### Detailed Description

##### Author

Federal Highway Administration

---

#### Constructor & Destructor Documentation

[\*imrcp.store.GribWrapper.GribWrapper \(int\[\] nObsTypes, GribParameter\[\] oParameters\)\*](#)

Constructs a new **GribWrapper** with the given observation types and GRIB2 parameters

##### Parameters

<i>nObsTypes</i>	IMRCP obseravtion types this file provides
<i>oParameters</i>	GRIB2 parameters that correspond to the IMRCP observation types in the file

---

## Member Function Documentation

`void imrcp.store.GribWrapper.cleanup (boolean bDelete)`

Clears `m_oEntryMap`

### Parameters

<code>bDelete</code>	not used for this implementation
----------------------	----------------------------------

Reimplemented from `imrcp.store.FileWrapper` (p.223).

`void imrcp.store.GribWrapper.deleteFile (File oFile)`

Attempts to delete the given File. Right now the implementation is commented out so files are not accidentally deleted

### Parameters

<code>oFile</code>	File to delete
--------------------	----------------

Reimplemented from `imrcp.store.FileWrapper` (p.224).

`void imrcp.store.GribWrapper.getData (int nObsTypeId, long lTimestamp, int nLat1, int nLon1, int nLat2, int nLon2, ImrcpResultSet oObs)`

Determines the cells of the grid corresponding to the given lat/lon bounding box and fills the given `ImrcpResultSet` with `imrcp.store.Obs` represent the data in those grids

### Parameters

<code>nObsTypeId</code>	observation type id
<code>lTimestamp</code>	query time in milliseconds since Epoch
<code>nLat1</code>	lower bound of latitude in decimal degrees scaled to 7 decimal places
<code>nLon1</code>	lower bound of longitude in decimal degrees scaled to 7 decimal places
<code>nLat2</code>	upper bound of latitude in decimal degrees scaled to 7 decimal places
<code>nLon2</code>	upper bound of longitude in decimal degrees scaled to 7 decimal places
<code>oObs</code>	<code>ImrcpResultSet</code> that will be filled with obs

`void imrcp.store.GribWrapper.getGrids (ArrayList< Grid > oGrids) throws Exception`

This method contains the algorithm for parsing a file that represents a single GRIB2 file or multiple GRIB2 files concatenated together.

### Parameters

<code>oGrids</code>	list to add the Grids read from the file
---------------------	--

### Exceptions

<code>Exception</code>	
------------------------	--

`void imrcp.store.GribWrapper.getIndices (int nLon, int nLat, int[] nIndices)`

Fills the given int array with the x and y coordinates/indices of the grid that correspond to the given longitude and latitude.

*Parameters*

<i>nLon</i>	longitude in decimal degrees scaled to 7 decimal places
<i>nLat</i>	latitude in decimal degrees scaled to 7 decimal places
<i>nIndices</i>	array to fill with x and y coordinates of the grid that correspond to the longitude and latitude [x,y]

Reimplemented from **imrcp.store.GriddedFileWrapper** (*p.258*).

**static DataRep imrcp.store.GribWrapper.getLayout (DataInputStream oIn, int nSecLen) throws IOException [static]**

Wrapper for **DataRep#newDataRep (java.io.DataInputStream, int)**  
which reads Section 5 of a GRIB2 file.

*Parameters*

<i>oIn</i>	DataInputStream of the GRIB2 file. Its position should be at 5 bytes past the start of Section 5 (meaning the length (4 bytes) of the section and section number (1 byte) has been read)
<i>nSecLen</i>	length of the section in bytes

*Returns*

DataRep object defined by Section 5 or null if the template in the section has not been implemented

*Exceptions*

<i>IOException</i>
--------------------

**static GribParameter imrcp.store.GribWrapper.getParameter (DataInputStream oIn, int nSecLen, int[] nDiscipline, GribParameter[] oParameters) throws IOException [static]**

Parses Section 4 of a GRIB2 file to get the GribParameter defined in the section.

*Parameters*

<i>oIn</i>	DataInputStream of the GRIB2 file. Its position should be at 5 bytes past the start of Section 4 (meaning the length (4 bytes) of the section and section number (1 byte) has been read)
<i>nSecLen</i>	number of bytes in the section
<i>nDiscipline</i>	GRIB2 discipline
<i>oParameters</i>	array of GribParameters used by the system.

*Returns*

The GribParameter defined by the section or null if it does not match one of the given GribParameters

*Exceptions*

<i>IOException</i>
--------------------

**static Projection imrcp.store.GribWrapper.getProj (DataInputStream oIn, int nSecLen) throws IOException [static]**

Wrapper for **Projection#newProjection (java.io.DataInputStream, int)** which reads Section 3 of a GRIB2 file

*Parameters*

<i>oIn</i>	DataInputStream of the GRIB2 file. Its position should be at 5 bytes past the start of Section 3 (meaning the length (4 bytes) of the section and section number (1 byte) has been read)
<i>nSecLen</i>	length of the section in bytes

*Returns*

Projection object created by parsing Section 3 or null if an error occurred or the projection defined in the file has not been implemented yet.

*Exceptions*

<i>IOException</i>
--------------------

*double imrcp.store.GribWrapper.getReading (int nObsType, long lTimestamp, int nLat, int nLon, Date oTimeRecv)*

NOT IMPLEMENTED

Reimplemented from **imrcp.store.GriddedFileWrapper** (p.258).

*double imrcp.store.GribWrapper.getReading (int nObsType, long lTimestamp, int[] nIndices)*

Get the value of the cell of the grid for the given observation type at the given indices/coordinates

*Parameters*

<i>nObsType</i>	desired observation type
<i>lTimestamp</i>	query time in milliseconds since Epoch
<i>nIndices</i>	[x coordinate, y coordinate]

*Returns*

Reimplemented from **imrcp.store.GriddedFileWrapper** (p.259).

*void imrcp.store.GribWrapper.load (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId) throws Exception*

Sets all of the given parameters and then calls **getGrids** to load the file into memory and get the data needed to create the **GribEntryData**'s representing the data stored in the file.

Reimplemented from **imrcp.store.FileWrapper** (p.224).

*static long imrcp.store.GribWrapper.readHeader (DataInputStream oIn, int[] nDiscipline) throws IOException [static]*

Reads the header (Section 0) of a GRIB2 file

*Parameters*

<i>oIn</i>	DataInputStream of the GRIB2 file. Its position should be at the start of the file
<i>nDiscipline</i>	array used to store the GRIB2 discipline

*Returns*

The number of bytes left in the file

*Exceptions*

<i>IOException</i>
--------------------

**static int imrcp.store.GribWrapper.readSection (DataInputStream oIn, byte[] ySection) throws Exception [static]**

Reads the length of the section (4 bytes) and the section number (1 byte) of the current section

*Parameters*

<i>oIn</i>	DataInputStream of the GRIB2 file. Its position should be at the start of a GRIB2 section
<i>ySection</i>	stores the section number

*Returns*

Length of the section in bytes

*Exceptions*

<i>Exception</i>
------------------

---

## Member Data Documentation

**int imrcp.store.GribWrapper.ENDMARKER = 926365495 [static]**

This integer marks the end of a GRIB2 file

**int imrcp.store.GribWrapper.GRIBMARKER = 1196575042 [static]**

This integer marks the start of a GRIB2 file

**long imrcp.store.GribWrapper.m\_lFileTime**

Stores the last modified time (downloaded time) of the file

**int [] imrcp.store.GribWrapper.m\_nDiscipline = new int[]{-1}**

Stores the GRIB2 discipline. We use an array since this value is set in the static function **GribWrapper#readHeader(java.io.DataInputStream, int[])** and it already has a return value.

---

*The documentation for this class was generated from the following file:*

- store/GribWrapper.java
- 

## imrcp.store.grib.Grid Class Reference

### Public Member Functions

- **Grid** (DataInputStream oIn, int nSecLen, **Projection** oProj, **DataRep** oDataRep, **GribParameter** oParameter) throws IOException, InterruptedException

## Public Attributes

- **Projection m\_oProj**
  - **DataRep m\_oDataRep**
  - **float[][] m\_fData**
  - **GribParameter m\_oParameter**
- 

## Detailed Description

Represents the **Grid** of data in a GRIB2 file found in Section 7.

### Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

*imrcp.store.grib.Grid.Grid (DataInputStream oIn, int nSecLen, Projection oProj,  
DataRep oDataRep, GribParameter oParameter) throws IOException,  
InterruptedException*

Constructs a **Grid** with the given parameters.

### Parameters

<i>oIn</i>	DataInputStream of the GRIB2 file. It should be ready to read the 6th byte of Section 7 (meaning the length (4 bytes) of the section and section number (1 byte) has been read)
<i>nSecLen</i>	length of Section 7 in byte
<i>oProj</i>	<b>Projection</b> object that contains the data from Section 3
<i>oDataRep</i>	Data Representation object that defines how to interpret the data in Section 7
<i>oParameter</i>	GRIB2 observation type of the data

### Exceptions

<i>IOException</i>	
<i>InterruptedException</i>	

---

## Member Data Documentation

*float [][] imrcp.store.grib.Grid.m\_fData*

Arrays used to store the gridded data

*DataRep imrcp.store.grib.Grid.m\_oDataRep*

Data Representation object (from Section 5) that defines how to interpret the data in Section 7.

*GribParameter imrcp.store.grib.Grid.m\_oParameter*

Object that defines the observation type of the data

*Projection imrcp.store.grib.Grid.m\_oProj*

**Projection** object that contains the data from Section 3

---

*The documentation for this class was generated from the following file:*

- store/grib/Grid.java
- 

## imrcp.store.GriddedFileWrapper Class Reference

### Public Member Functions

- **GriddedFileWrapper ()**
- abstract double **getReading** (int nObsType, long lTimestamp, int nLat, int nLon, Date oTimeRecv)
- abstract void **getIndices** (int nLon, int nLat, int[] nIndices)
- abstract double **getReading** (int nObsType, long lTimestamp, int[] nIndices)
- **EntryData getEntryByObsId** (int nObsTypeId)
- int **getTimeIndex** (**EntryData** oData, long lTimestamp)

### Static Protected Member Functions

- static int **getIndex** (double[] dValues, Double oValue)

### Protected Attributes

- ArrayList< **EntryData** > **m\_oEntryMap**

### Additional Inherited Members

---

### Detailed Description

Base class for data files containing a grid(s) (usually based on geolocation) of values.

#### Author

Federal Highway Administration

---

### Constructor & Destructor Documentation

#### *imrcp.store.GriddedFileWrapper.GriddedFileWrapper ()*

Default constructor. Does nothing.

---

### Member Function Documentation

#### *EntryData imrcp.store.GriddedFileWrapper.getEntryByObsId (int nObsTypeId)*

Gets the **EntryData** that contains data of the given observation type id.

##### Parameters

<i>nObsTypeId</i>	IMRCP observation type id
-------------------	---------------------------

##### Returns

The **EntryData** that contains data of the request observation type or null if that does not exist.

*static int imrcp.store.GriddedFileWrapper.getIndex (double[] dValues, Double oValue) [static], [protected]*

Determines the index that the given value would be in the given array of values.

*Parameters*

<i>dValues</i>	array of values. The values should be sorted and the step from one value to the next should be close to equal for each value in the value
<i>oValue</i>	The value to test.

*Returns*

The index associated with the given value if the value is within the range of values, otherwise -1.

*abstract void imrcp.store.GriddedFileWrapper.getIndices (int nLon, int nLat, int[] nIndices) [abstract]*

Fills the given int array with the x and y coordinates/indices of the grid that correspond to the given longitude and latitude.

*Parameters*

<i>nLon</i>	longitude in decimal degrees scaled to 7 decimal places
<i>nLat</i>	latitude in decimal degrees scaled to 7 decimal places
<i>nIndices</i>	array to fill with x and y coordinates of the grid that correspond to the longitude and latitude [x,y]

Reimplemented in **imrcp.store.DataObsWrapper** (p.131), **imrcp.store.GribWrapper** (p.252), **imrcp.store.NcfWrapper** (p.375), and **imrep.web.tiles.TileWrapper** (p.532).

*abstract double imrcp.store.GriddedFileWrapper.getReading (int nObsType, long lTimestamp, int nLat, int nLon, Date oTimeRecv) [abstract]*

Gets the value of the cell of the grid for the given observation type at the given lon/lat and time.

*Parameters*

<i>nObsType</i>	desired observation type
<i>lTimestamp</i>	query time in milliseconds since Epoch
<i>nLat</i>	latitude of query in decimal degrees scaled to 7 decimal places
<i>nLon</i>	longitude of query in decimal degrees scaled to 7 decimal places
<i>oTimeRecv</i>	Date object to have its time set to the time the observation was received for some implementation

*Returns*

the value of the cell of the grid for the given observation type at the given lon/lat and time, Double.NaN if a valid value does not exists for the query

Reimplemented in **imrcp.store.DataObsWrapper** (p.131), **imrcp.store.GribWrapper** (p.254), **imrep.web.tiles.TileWrapper** (p.532), **imrcp.store.NcfWrapper** (p.375), and **imrcp.store.RapNcfWrapper** (p.469).

*abstract double imrcp.store.GriddedFileWrapper.getReading (int nObsType, long lTimestamp, int[] nIndices) [abstract]*

Get the value of the cell of the grid for the given observation type at the given indices/coordinates

*Parameters*

<i>nObsType</i>	desired observation type
<i>lTimestamp</i>	query time in milliseconds since Epoch
<i>nIndices</i>	[x coordinate, y coordinate]

*Returns*

Reimplemented in **imrcp.store.DataObsWrapper** (*p.132*), **imrcp.store.GribWrapper** (*p.254*), **imrcp.store.NcfWrapper** (*p.375*), **imrcp.store.RapNcfWrapper** (*p.469*), and **imrcp.web.tiles.TileWrapper** (*p.532*).

*int imrcp.store.GriddedFileWrapper.getTimeIndex (EntryData oData, long lTimestamp)*

Default is to return 0 as most files do not have multiple time indices. Child classes must implement this function if there are multiple times defined in the file.

*Parameters*

<i>oData</i>	The <b>EntryData</b> being queried
<i>lTimestamp</i>	query time in milliseconds since Epoch

*Returns*

Reimplemented in **imrcp.store.NcfWrapper** (*p.376*).

---

## Member Data Documentation

*ArrayList<EntryData> imrcp.store.GriddedFileWrapper.m\_oEntryMap [protected]*

Stores the **EntryData** per observation type contained in the file.

---

*The documentation for this class was generated from the following file:*

- store/GriddedFileWrapper.java

---

## imrcp.geosrv.osm.HashBucket Class Reference

### Public Member Functions

- **HashBucket ()**
- **HashBucket (int nHash)**
- **HashBucket (int nX, int nY)**
- **int compareTo (HashBucket o)**

## Static Public Member Functions

- static int **hashLonLat** (int nLon, int nLat)
- static int **hashBucketVals** (int nX, int nY)
- static void **unhash** (int nHash, int[] nVals)
- static int **getBucket** (int nValue)

## Public Attributes

- int **m\_nHash**
  - ArrayList<OsmNode> **m\_oNodes** = new ArrayList()
- 

## Detailed Description

Represents a cell(bucket) on a hashed grid of the world used to spatially index roadway segments for quick lookup. The hash indices are computed by computing 16-bit values from latitude and longitude coordinates based off of **BUCKET\_SPACING** and storing the longitude value in the first 2 bytes of an integer and the latitude value in the last 2 bytes of the same integer.

### Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

### *imrcp.geosrv.osm.HashBucket.HashBucket ()*

Default constructor. Does nothing.

### *imrcp.geosrv.osm.HashBucket.HashBucket (int nHash)*

Constructs a new **HashBucket** with the given hash index

#### Parameters

<i>nHash</i>	Hash index of the bucket
--------------	--------------------------

### *imrcp.geosrv.osm.HashBucket.HashBucket (int nX, int nY)*

Constructs a new **HashBucket** with the given longitude and latitude by calculating the hash index.

#### Parameters

<i>nX</i>	longitude in decimal degrees scaled to 7 decimal places
<i>nY</i>	latitude in decimal degrees scaled to 7 decimal places

---

## Member Function Documentation

### *int imrcp.geosrv.osm.HashBucket.compareTo (HashBucket o)*

Compares HashBuckets by hash value.

### *static int imrcp.geosrv.osm.HashBucket.getBucket (int nValue) [static]*

Floors the given latitude or longitude using **BUCKET\_SPACING** and divide that number by **BUCKET\_SPACING** to compute the bucket value.

*Parameters*

<i>nValue</i>	latitude or longitude in decimal degrees scaled to 7 decimal places
---------------	---

*Returns*

Bucket value of the given latitude or longitude.

**static int imrcp.geosrv.osm.HashBucket.hashBucketVals (int nX, int nY) [static]**

Computes the hash index of the given x and y bucket values.

*Parameters*

<i>nX</i>	value returned from <b>getBucket(int)</b> of a longitude in decimal degrees scaled to 7 decimal places
<i>nY</i>	value returned from <b>getBucket(int)</b> of a latitude in decimal degrees scaled to 7 decimal places

*Returns*

Hash index of the bucket with the x and y value

**static int imrcp.geosrv.osm.HashBucket.hashLonLat (int nLon, int nLat) [static]**

Computes the hash index of the given longitude and latitude

*Parameters*

<i>nLon</i>	longitude in decimal degrees scaled to 7 decimal places
<i>nLat</i>	latitude in decimal degrees scaled to 7 decimal places

*Returns*

Hash index of the bucket the point is inside of

**static void imrcp.geosrv.osm.HashBucket.unhash (int nHash, int[] nVals) [static]**

Fills the given array with the x and y bucket values for the given hash index

*Parameters*

<i>nHash</i>	hash index
<i>nVals</i>	array to be filled with [x bucket value, y bucket value]

---

## Member Data Documentation

**int imrcp.geosrv.osm.HashBucket.m\_nHash**

Hash index

**ArrayList<OsmNode> imrcp.geosrv.osm.HashBucket.m\_oNodes = new ArrayList()**

List of OsmNodes inside the bucket

---

*The documentation for this class was generated from the following file:*

- geosrv/osm/HashBucket.java

---

## [imrcp.system.shp.Header Class Reference](#)

### Public Member Functions

- **Header** (DataInputStream oDataInputStream) throws Exception
- 

### Detailed Description

Reads header of a file byte stream.

#### Author

Federal Highway Administration

#### Version

1.0 (April 27, 2007)

---

### Constructor & Destructor Documentation

*imrcp.system.shp.Header.Header (DataInputStream oDataInputStream) throws Exception*

Creates a new instance of **Header** with data input stream defined.

#### Parameters

<i>oDataInputStream</i>	the input stream for this header
-------------------------	----------------------------------

#### Exceptions

<i>java.lang.Exception</i>	
----------------------------	--

---

*The documentation for this class was generated from the following file:*

- system/shp/Header.java
- 

---

## [imrcp.forecast.mlp.MLPHurricane.HurData Class Reference](#)

### Public Attributes

- int **m\_nLandfallCat**
  - long **m\_llLandfallTime**
  - double[] **m\_dDistances**
- 

### Detailed Description

Contains data that gets mapped to an associated roadway segment used in the MLP Hurricane models

---

## Member Data Documentation

### `double [] imrcp.forecast.mlp.MLPHurricane.HurData.m_dDistances`

Distance in meters from the midpoint of the associated roadway segment to each of the cities in the MLP Hurricane model.

[distance to Lake Charles, distance to Lafayette, distance to Baton Rouge, distance to New Orleans, distance to Alexandria, distance to Shreveport, distance to Monroe]

### `long imrcp.forecast.mlp.MLPHurricane.HurData.m_lLandfallTime`

Time in milliseconds since Epoch the projected center of the storm is closest to the associated roadway segment.

### `int imrcp.forecast.mlp.MLPHurricane.HurData.m_nLandfallCat`

Predicted hurricane category when the projected center of the storm is closest to the associated roadway segment

---

*The documentation for this class was generated from the following file:*

- forecast/mlp/MLPHurricane.java
- 

## imrcp.forecast.mlp.HurHistDataRecord Class Reference

### Public Member Functions

- **HurHistDataRecord ()**
- **HurHistDataRecord (CsvReader oIn, WayNetworks oWays)** throws Exception
- void **write (Writer oOut)** throws IOException
- int **compareTo (HurHistDataRecord o)**

### Public Attributes

- **OsmWay m\_oWay**
- long **m\_lTimestamp**
- double **m\_dSpeed**
- double **m\_dSpeedStd**
- int **m\_nStatusHur**
- int **m\_nLatHur**
- int **m\_nLonHur**
- int **m\_nMaxSpeedHur**
- int **m\_nMinPressureHur**

### Static Public Attributes

- static String **HEADER** =  
"linkid,onlydate,t\_start,t\_period,Speed,speed\_std,DayOfWeek,direction,ref,lat,lon,length,StatusHu  
r,LatHur,LonHur,MaxSpeed,MinPressure,TimeStamp\n"

---

### Detailed Description

Contains the data needed for a single record in the hurricane data file for the MLP Hurricane model

### Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

*imrcp.forecast.mlp.HurHistDataRecord.HurHistDataRecord ()*

Default constructor. Does nothing.

*imrcp.forecast.mlp.HurHistDataRecord.HurHistDataRecord (CsvReader oIn,  
WayNetworks oWays) throws Exception*

Constructs a **HurHistDataRecord** by using the CsvReader to parse the contents of a line from the hurricane histdat file.

### Parameters

<i>oIn</i>	CsvReader that already has called <b>CsvReader#readLine ()</b> and is ready to parse the current line.
<i>oWays</i>	Object used to look up OsmWays by Id

### Exceptions

<i>Exception</i>	
------------------	--

---

## Member Function Documentation

*int imrcp.forecast.mlp.HurHistDataRecord.compareTo (HurHistDataRecord o)*

Compares HurHistDataRecords by the way id and then timestamp.

### See also

java.lang.Comparable::compareTo(java.lang.Object)

*void imrcp.forecast.mlp.HurHistDataRecord.write (Writer oOut) throws IOException*

Writes the **HurHistDataRecord** as a CSV line to the given Writer

### Parameters

<i>oOut</i>	Writer to write the record to
-------------	-------------------------------

### Exceptions

<i>IOException</i>	
--------------------	--

---

## Member Data Documentation

*String imrcp.forecast.mlp.HurHistDataRecord.HEADER =  
"linkid,onlydate,t\_start,t\_period,Speed,speed\_std,DayOfWeek,direction,ref,lat,lon,len  
gth,StatusHur,LatHur,LonHur,MaxSpeed,MinPressure,Timestamp\n" [static]*

Header of the CSV file

*double imrcp.forecast.mlp.HurHistDataRecord.m\_dSpeed*

Average of the speed observations that occurred in the hour this record represents

*double imrcp.forecast.mlp.HurHistDataRecord.m\_dSpeedStd*

The standard deviation of the speed observations that occurred in the hours this record represents

*long imrcp.forecast.mlp.HurHistDataRecord.m\_lTimestamp*

Timestamp in milliseconds since Epoch of the hour the speed observations occurred in

*int imrcp.forecast.mlp.HurHistDataRecord.m\_nLatHur*

Predicted latitude of the hurricane at the hour of this record in decimal degrees scaled to 7 decimal places

*int imrcp.forecast.mlp.HurHistDataRecord.m\_nLonHur*

Predicted longitude of the hurricane at the hour of this record in decimal degrees scaled to 7 decimal places

*int imrcp.forecast.mlp.HurHistDataRecord.m\_nMaxSpeedHur*

Maximum predicted wind speed of the hurricane

*int imrcp.forecast.mlp.HurHistDataRecord.m\_nMinPressureHur*

Minimum predicted pressure of the hurricane

*int imrcp.forecast.mlp.HurHistDataRecord.m\_nStatusHur*

1, 2, or 3 depending on the predicted storm type. 3 = HU or MH (Hurricane or Major Hurricane) 2 = TS (Tropical Storm) 1 = anything else

*OsmWay imrcp.forecast.mlp.HurHistDataRecord.m\_oWay*

Roadway segment associated with the record

---

*The documentation for this class was generated from the following file:*

- forecast/mlp/HurHistDataRecord.java
- 

## imrcp.store.HurricaneCenter Class Reference

### Public Member Functions

- **HurricaneCenter** (int nLon, int nLat, int nWindSpeed, long lTimestamp, double dDistance)
- int **compareTo** (**HurricaneCenter** o)

### Public Attributes

- long **m\_lTimestamp**
  - int **m\_nMaxSpeed**
  - int **m\_nLon**
  - int **m\_nLat**
  - double **m\_dDistance**
- 

### Detailed Description

Represents the time and location of a forecasted hurricane

*Author*

Federal Highway Administration

---

**Constructor & Destructor Documentation**

*imrcp.store.HurricaneCenter.HurricaneCenter (int nLon, int nLat, int nWindSpeed, long lTimestamp, double dDistance)*

Constructs a new **HurricaneCenter** with the given parameters.

*Parameters*

<i>nLon</i>	Forecasted longitude of hurricane center in decimal degrees scaled to 7 decimal places
<i>nLat</i>	Forecasted latitude of hurricane center in decimal degrees scaled to 7 decimal places
<i>nWindSpeed</i>	max wind speed in knots
<i>lTimestamp</i>	time in milliseconds since Epoch hurricane center is at the given location
<i>dDistance</i>	radius of the hurricane at the given time and location

---

**Member Function Documentation**

*int imrcp.store.HurricaneCenter.compareTo (HurricaneCenter o)*

Compares HurricaneCenters by timestamp

---

**Member Data Documentation**

*double imrcp.store.HurricaneCenter.m\_dDistance*

Forecasted radius of the hurricane (minimum distance to the edge of the hurricane cone of probability from NHC)

*long imrcp.store.HurricaneCenter.m\_lTimestamp*

Time in milliseconds since Epoch the hurricane is forecasted to be at this location

*int imrcp.store.HurricaneCenter.m\_nLat*

Forecasted latitude of hurricane center in decimal degrees scaled to 7 decimal places

*int imrcp.store.HurricaneCenter.m\_nLon*

Forecasted longitude of hurricane center in decimal degrees scaled to 7 decimal places

*int imrcp.store.HurricaneCenter.m\_nMaxSpeed*

Forecasted maximum wind speed in knots

---

*The documentation for this class was generated from the following file:*

- store/HurricaneCenter.java
-

## imrcp.forecast.mlp.MLPHurricane.HurWork Class Reference

### Public Member Functions

- **HurWork (NHCWrapper oNHC, ArrayList< Obs > oHurCats, ArrayList< OsmWay > oWays, double[] dLandfall, int nThread, Delegate oDelegate)**
  - **void run ()**
- 

### Detailed Description

Does the actual work of running the MLP Hurricane models by using the RConnection and Rserve interfaces.

---

### Constructor & Destructor Documentation

*imrcp.forecast.mlp.MLPHurricane.HurWork (NHCWrapper oNHC, ArrayList< Obs > oHurCats, ArrayList< OsmWay > oWays, double[] dLandfall, int nThread, Delegate oDelegate)*

Constructs a **HurWork** with the given parameters.

#### Parameters

<i>oNHC</i>	NHC hurricane forecast file to process
<i>oHurCats</i>	Hurricane category observations for the hurricane forecast
<i>oWays</i>	List of OsmWays to process
<i>dLandfall</i>	[lon of landfall, lat of landfall, sshws hurricane category]
<i>nThread</i>	thread number
<i>oDelegate</i>	<b>Delegate</b> executing this <b>HurWork</b>

---

### Member Function Documentation

*void imrcp.forecast.mlp.MLPHurricane.HurWork.run ()*

Creates the necessary input files and runs the 3 models (SVM, oneshot, and online prediction), if it is first run of this file, or just the online prediction if it isn't the first run.

---

*The documentation for this class was generated from the following file:*

- forecast/mlp/MLPHurricane.java
- 

## imrcp.system.Id Class Reference

### Public Member Functions

- **Id (int nType, int[] nGrowableArr) throws IOException**
- **Id (int nType, int[] nGrowableArr, int nStart) throws IOException**
- **Id (String sId)**
- **Id (DataInputStream oIn) throws IOException**
- **void write (DataOutputStream oOut) throws IOException**
- **int compareTo (Id o)**
- **String toString ()**

- long **getLowBytes** ()
  - long **getHighBytes** ()

## Static Public Member Functions

- static boolean **isNull** (**Id** oId)
  - static boolean **isSensor** (**Id** oId)
  - static boolean **isNode** (**Id** oId)
  - static boolean **isLink** (**Id** oId)
  - static boolean **isSegment** (**Id** oId)
  - static boolean **isRoute** (**Id** oId)

## Static Public Attributes

- static final byte **SENSOR** = 1
  - static final byte **NODE** = 2
  - static final byte **LINK** = 3
  - static final byte **SEGMENT** = 4
  - static final byte **ROUTE** = 5
  - static final Comparator< **Id** > **COMPARATOR** = (**Id** o1, **Id** o2) -> o1.compareTo(o2)
  - static **Id** **NULLID** = new **Id**(**NULLSTRING**)

## Protected Member Functions

- **Id** ()

## Protected Attributes

- long m\_l0
  - long m\_l1

## Static Protected Attributes

- static final int **SIZE** = 16
  - static final String **NULLSTRING** = "00000000000000000000000000000000"

## Detailed Description

This class is used to represent identifiers of objects used in the IMRCP system.

## *Author*

Federal Highway Administration

# Constructor & Destructor Documentation

*imrcp.system.Id.Id () [protected]*

Default constructor. Does nothing.

*imrcp.system.Id.Id (int nType, int[] nGrowableArr) throws IOException*

### Parameters

<i>nType</i>	the type of object this <b>Id</b> represents, use <b>SENSOR</b> , <b>NODE</b> , <b>LINK</b> , <b>SEGMENT</b> , or <b>ROUTE</b>
<i>nGrowableArr</i>	Growable array containing the unique values of the object to use in the Shake256 squeeze algorithm

*Exceptions*

<i>IOException</i>
--------------------

*imrcp.system.Id.Id (int nType, int[] nGrowableArr, int nStart) throws IOException*

*Parameters*

<i>nType</i>	the type of object this <b>Id</b> represents, use <b>SENSOR</b> , <b>NODE</b> , <b>LINK</b> , <b>SEGMENT</b> , or <b>ROUTE</b>
<i>nGrowableArr</i>	
<i>nStart</i>	

*Exceptions*

<i>IOException</i>
--------------------

*imrcp.system.Id.Id (String sId)*

Constructs an **Id** from the String representation of it

*Parameters*

<i>sId</i>	String representation of the <b>Id</b> , which is the 16 bytes written as a hex string
------------	--

*imrcp.system.Id.Id (DataInputStream oIn) throws IOException*

Constructs an **Id** from the given DataInputStream by reading the next two longs.

*Parameters*

<i>oIn</i>	DataInputStream to read from
------------	------------------------------

*Exceptions*

<i>IOException</i>
--------------------

## Member Function Documentation

*int imrcp.system.Id.compareTo (Id o)*

C.compares Ids by the first 8 bytes, **m\_10**, then the second 8 bytes, **m\_11**

*Parameters*

<i>o</i>
----------

*Returns*

*long imrcp.system.Id.getHighBytes ()*

Gets the high bytes of the **Id**

*Returns*

The high bytes of the **Id**, **m\_10**

*long imrcp.system.Id.getLowBytes ()*

Gets the low bytes of the **Id**

*Returns*

The low bytes of the **Id**, **m\_11**

*static boolean imrcp.system.Id.isLink (Id oid) [static]*

Indicates if the object the **Id** represents is a Link

*Parameters*

<i>oId</i>	<b>Id</b> to test
------------	-------------------

*Returns*

true if the the object the **Id** represents is a Link, otherwise false

*static boolean imrcp.system.Id.isNode (Id oid) [static]*

Indicates if the object the **Id** represents is a Node

*Parameters*

<i>oId</i>	<b>Id</b> to test
------------	-------------------

*Returns*

true if the the object the **Id** represents is a Node, otherwise false

*static boolean imrcp.system.Id.isNull (Id oid) [static]*

Indicates if the **Id** is the null **Id**.

*Parameters*

<i>oId</i>	<b>Id</b> to test
------------	-------------------

*Returns*

true if the **Id** is the null **Id**, otherwise false

*static boolean imrcp.system.Id.isRoute (Id oid) [static]*

Indicates if the object the **Id** represents is a Route

*Parameters*

<i>oId</i>	<b>Id</b> to test
------------	-------------------

*Returns*

true if the the object the **Id** represents is a Route, otherwise false

*static boolean imrcp.system.Id.isSegment (Id oid) [static]*

Indicates if the object the **Id** represents is a Segment

*Parameters*

<i>oId</i>	<b>Id</b> to test
------------	-------------------

*Returns*

true if the the object the **Id** represents is a Segment, otherwise false

**static boolean imrcp.system.Id.isSensor (Id old) [static]**

Incidates if the object the **Id** represents is a Sensor

*Parameters*

<i>old</i>	<b>Id</b> to test
------------	-------------------

*Returns*

true if the the object the **Id** represents is a Sensor, otherwise false

**String imrcp.system.Id.toString ()**

Converts **m\_10** and **m\_11** first to bytes and then those bytes to a hex string.

*Returns*

Hex string representation of the 16 bytes of the **Id**

**void imrcp.system.Id.write (DataOutputStream oOut) throws IOException**

Writes the **Id** as two longs to the given DataOutputStream

*Parameters*

<i>oOut</i>	DataOutputStream to write the <b>Id</b> to
-------------	--

*Exceptions*

<b>IOException</b>
--------------------

---

## Member Data Documentation

**final Comparator<Id> imrcp.system.Id.COMPARATOR = (Id o1, Id o2) ->**  
**o1.compareTo(o2) [static]**

Comparator that wraps **compareTo (imrcp.system.Id)**

**final byte imrcp.system.Id.LINK = 3 [static]**

Byte indicating if the **Id** represents a link

**long imrcp.system.Id.m\_10 [protected]**

Long that stores the first 8 bytes of the **Id**

**long imrcp.system.Id.m\_11 [protected]**

Long that stores the second 8 bytes of the **Id**

**final byte imrcp.system.Id.NODE = 2 [static]**

Byte indicating if the **Id** represents a node

**Id imrcp.system.Id.NULLID = new Id(NULLSTRING) [static]**

Single instance used for a "null" id

String used to create **NULLID**

*final byte imrcp.system.Id.ROUTE = 5 [static]*

Byte indicating if the **Id** represents a route

*final byte imrcp.system.Id.SEGMENT = 4 [static]*

Byte indicating if the **Id** represents a roadway segment

*final byte imrcp.system.Id.SENSOR = 1 [static]*

Byte indicating if the **Id** represents a sensor

*final int imrcp.system.Id.SIZE = 16 [static], [protected]*

### Size of the Ids in bytes

*The documentation for this class was generated from the following file:*

- system/Id.java

## imrcp.system.ImrcpBlock Interface Reference

## Public Member Functions

- void **notify** (String sMessageName, String... sResources)
  - void **receive** (String[] sMessage)
  - void **register** ()
  - void **init** ()
  - void **destroy** ()
  - long[] **status** ()
  - String **getName** ()
  - void **setName** (String sName)
  - **BlockConfig getConfig** ()

## Static Public Attributes

- static final int **FROM** = 0
  - static final int **MESSAGE** = 1
  - static final int **OVERRIDESTATUS** = -1
  - static final int **STARTING** = 0
  - static final int **RUNNING** = 1
  - static final int **IDLE** = 2
  - static final int **ERROR** = 3
  - static final int **STOPPING** = 4
  - static final int **STOPPED** = 5
  - static final int **INIT** = 6
  - static final String[] **STATUSES**

## Detailed Description

Interface used for components of the IMRCP system.

## Author

Federal Highway Administration

---

## Member Function Documentation

*void imrcp.system.ImrcpBlock.destroy ()*

Method called when the **ImrcpBlock** is finished being active in the system

Implemented in **imrcp.system.BaseBlock** (p.95).

*BlockConfig imrcp.system.ImrcpBlock.getConfig ()*

Method called to get the configuration object for this **ImrcpBlock**

### Returns

configuration object

Implemented in **imrcp.system.BaseBlock** (p.96).

*String imrcp.system.ImrcpBlock.getName ()*

Method called to get the instance name of the **ImrcpBlock**

### Returns

Instance name of the **ImrcpBlock**

Implemented in **imrcp.system.BaseBlock** (p.97).

*void imrcp.system.ImrcpBlock.init ()*

Method called to initialize the **ImrcpBlock**

Implemented in **imrcp.system.BaseBlock** (p.97).

*void imrcp.system.ImrcpBlock.notify (String sMessageName, String... sResources)*

Method called to notify other subscribing system components with the given message and resources

### Parameters

<i>sMessageName</i>	type of message being sent
<i>sResources</i>	array of strings that represent the resources and details of the message

Implemented in **imrcp.system.BaseBlock** (p.97).

*void imrcp.system.ImrcpBlock.receive (String[] sMessage)*

Method called to receive a message from other system components.

### Parameters

<i>sMessage</i>	the message to receive
-----------------	------------------------

Implemented in **imrcp.system.BaseBlock** (p.98).

*void imrcp.system.ImrcpBlock.register ()*

Method called to register the **ImrcpBlock** to the system **Directory**

Implemented in **imrcp.system.BaseBlock** (p.98).

*void imrcp.system.ImrcpBlock.setName (String sName)*

Method called to set the instance name of the **ImrcpBlock**

#### Parameters

<code>sName</code>	desired instance name of the <b>ImrcpBlock</b>
--------------------	--

Implemented in **imrcp.system.BaseBlock** (p.99).

`long[] imrcp.system.ImrcpBlock.status ()`

Method called to get the current status of the **ImrcpBlock**

#### Returns

[current status, time in milliseconds since Epoch the status last changed]

Implemented in **imrcp.system.BaseBlock** (p.100).

---

## Member Data Documentation

`final int imrcp.system.ImrcpBlock.ERROR = 3 [static]`

Status enumeration indicating that the **ImrcpBlock** has had an error occur and is not ready to execute its task

`final int imrcp.system.ImrcpBlock.FROM = 0 [static]`

Index in string array system messages that corresponds to the **ImrcpBlock** the message is from

`final int imrcp.system.ImrcpBlock.IDLE = 2 [static]`

Status enumeration indicating that the **ImrcpBlock** is idle and ready to execute its task

`final int imrcp.system.ImrcpBlock.INIT = 6 [static]`

Status enumeration indicating that the **ImrcpBlock** is currently initializing

`final int imrcp.system.ImrcpBlock.MESSAGE = 1 [static]`

Index in string array system messages that corresponds to the type of message

`final int imrcp.system.ImrcpBlock.OVERRIDESTATUS = -1 [static]`

Status enumeration used to override the current status regardless of what it is

`final int imrcp.system.ImrcpBlock.RUNNING = 1 [static]`

Status enumeration indicating that the **ImrcpBlock** actively executing its task

`final int imrcp.system.ImrcpBlock.STARTING = 0 [static]`

Status enumeration indicating that the **ImrcpBlock** is starting

`final String [] imrcp.system.ImrcpBlock.STATUSES [static]`

```
Initial value:= new String[] {  
    "STARTING", "RUNNING", "IDLE", "ERROR", "STOPPING", "STOPPED", "INIT"}  
}
```

String names for the different status enumerations

`final int imrcp.system.ImrcpBlock.STOPPED = 5 [static]`

Status enumeration indicating that the **ImrcpBlock** has finished stopping its service

`final int imrcp.system.ImrcpBlock.STOPPING = 4 [static]`

Status enumeration indicating that the **ImrcpBlock** is actively shutting down and stopping its service

---

*The documentation for this interface was generated from the following file:*

- system/ImrcpBlock.java
- 

## [imrcp.store.ImrcpCapResultSet Class Reference](#)

### Classes

- interface **IntDelegate**

### Public Member Functions

- **ImrcpCapResultSet ()**

### Protected Attributes

- SimpleDateFormat **m\_oFormat** = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss.SSS")

### [Additional Inherited Members](#)

---

### Detailed Description

The implementation for **ImrcpResultSet** s that contain **CAPObs**

#### Author

Federal Highway Administration

---

### Constructor & Destructor Documentation

#### [\*imrcp.store.ImrcpCapResultSet.ImrcpCapResultSet \(\)\*](#)

Default constructor. Defines the column names and necessary delegate objects

---

### Member Data Documentation

*SimpleDateFormat imrcp.store.ImrcpCapResultSet.m\_oFormat = new  
SimpleDateFormat("yyyy/MM/dd HH:mm:ss.SSS") [protected]*

Object used to format time stamps into date strings

---

*The documentation for this class was generated from the following file:*

- store/ImrcpCapResultSet.java
- 

## [imrcp.store.ImrcpEventResultSet Class Reference](#)

### Classes

- interface **IntDelegate**

## Public Member Functions

- **ImrcpEventResultSet ()**

## Protected Attributes

- `SimpleDateFormat m_oFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss.SSS")`

## Additional Inherited Members

---

### Detailed Description

The implementation for **ImrcpResultSet** s that contain **EventObs**

#### Author

Federal Highway Administration

---

### Constructor & Destructor Documentation

#### *imrcp.store.ImrcpEventResultSet.ImrcpEventResultSet ()*

Default constructor. Defines the column names and necessary delegate objects

---

### Member Data Documentation

*SimpleDateFormat imrcp.store.ImrcpEventResultSet.m\_oFormat = new  
SimpleDateFormat("yyyy/MM/dd HH:mm:ss.SSS") [protected]*

Object used to format time stamps into date strings

---

*The documentation for this class was generated from the following file:*

- [store/ImrcpEventResultSet.java](#)
- 

## imrcp.store.ImrcpObsResultSet Class Reference

### Classes

- interface **IntDelegate**

## Public Member Functions

- **ImrcpObsResultSet ()**

## Protected Attributes

- `SimpleDateFormat m_oFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss.SSS")`

## Additional Inherited Members

---

### Detailed Description

The implementation for **ImrcpResultSet** s that contain **Obs**

## Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

### *imrcp.store.ImrcpObsResultSet.ImrcpObsResultSet()*

Default constructor. Defines the column names and necessary delegate objects

---

## Member Data Documentation

*SimpleDateFormat imrcp.store.ImrcpObsResultSet.m\_oFormat = new  
SimpleDateFormat("yyyy/MM/dd HH:mm:ss.SSS") [protected]*

Object used to format time stamps into date strings

---

*The documentation for this class was generated from the following file:*

- store/ImrcpObsResultSet.java
- 

## imrcp.store.ImrcpResultSet< T > Class Template Reference

### Public Member Functions

- boolean **next** () throws SQLException
- void **close** () throws SQLException
- String **getString** (int columnIndex) throws SQLException
- short **getShort** (int columnIndex) throws SQLException
- int **getInt** (int columnIndex) throws SQLException
- long **getLong** (int columnIndex) throws SQLException
- double **getDouble** (int columnIndex) throws SQLException
- int[] **getIntArray** (int columnIndex) throws SQLException
- boolean **getBoolean** (int columnIndex) throws SQLException
- byte **getByte** (int columnIndex) throws SQLException
- float **getFloat** (int columnIndex) throws SQLException
- boolean **wasNull** () throws SQLException
- BigDecimal **getBigDecimal** (int columnIndex, int scale) throws SQLException
- byte[] **getBytes** (int columnIndex) throws SQLException
- Date **getDate** (int columnIndex) throws SQLException
- Time **getTime** (int columnIndex) throws SQLException
- Timestamp **getTimestamp** (int columnIndex) throws SQLException
- InputStream **getAsciiStream** (int columnIndex) throws SQLException
- InputStream **getUnicodeStream** (int columnIndex) throws SQLException
- InputStream **getBinaryStream** (int columnIndex) throws SQLException
- String **getString** (String columnLabel) throws SQLException
- boolean **getBoolean** (String columnLabel) throws SQLException
- byte **getByte** (String columnLabel) throws SQLException
- short **getShort** (String columnLabel) throws SQLException
- int **getInt** (String columnLabel) throws SQLException
- long **getLong** (String columnLabel) throws SQLException
- float **getFloat** (String columnLabel) throws SQLException
- double **getDouble** (String columnLabel) throws SQLException

- `BigDecimal getBigDecimal` (String columnLabel, int scale) throws SQLException
- `byte[] getBytes` (String columnLabel) throws SQLException
- `Date getDate` (String columnLabel) throws SQLException
- `Time getTime` (String columnLabel) throws SQLException
- `Timestamp getTimestamp` (String columnLabel) throws SQLException
- `InputStream getAsciiStream` (String columnLabel) throws SQLException
- `InputStream getUnicodeStream` (String columnLabel) throws SQLException
- `InputStream getBinaryStream` (String columnLabel) throws SQLException
- `SQLWarning getWarnings` () throws SQLException
- `void clearWarnings` () throws SQLException
- `String getCursorName` () throws SQLException
- `ResultSetMetaData getMetaData` () throws SQLException
- `Object getObject` (int columnIndex) throws SQLException
- `Object getObject` (String columnLabel) throws SQLException
- `int findColumn` (String columnLabel) throws SQLException
- `Reader getCharacterStream` (int columnIndex) throws SQLException
- `Reader getCharacterStream` (String columnLabel) throws SQLException
- `BigDecimal getBigDecimal` (int columnIndex) throws SQLException
- `BigDecimal getBigDecimal` (String columnLabel) throws SQLException
- `boolean isBeforeFirst` () throws SQLException
- `boolean isAfterLast` () throws SQLException
- `boolean isFirst` () throws SQLException
- `boolean isLast` () throws SQLException
- `void beforeFirst` () throws SQLException
- `void afterLast` () throws SQLException
- `boolean first` () throws SQLException
- `boolean last` () throws SQLException
- `int getRow` () throws SQLException
- `boolean absolute` (int row) throws SQLException
- `boolean relative` (int rows) throws SQLException
- `boolean previous` () throws SQLException
- `void setFetchDirection` (int direction) throws SQLException
- `int getFetchDirection` () throws SQLException
- `void setFetchSize` (int rows) throws SQLException
- `int getFetchSize` () throws SQLException
- `int getType` () throws SQLException
- `int getConcurrency` () throws SQLException
- `boolean rowUpdated` () throws SQLException
- `boolean rowInserted` () throws SQLException
- `boolean rowDeleted` () throws SQLException
- `void updateNull` (int columnIndex) throws SQLException
- `void updateBoolean` (int columnIndex, boolean x) throws SQLException
- `void updateByte` (int columnIndex, byte x) throws SQLException
- `void updateShort` (int columnIndex, short x) throws SQLException
- `void updateInt` (int columnIndex, int x) throws SQLException
- `void updateLong` (int columnIndex, long x) throws SQLException
- `void updateFloat` (int columnIndex, float x) throws SQLException
- `void updateDouble` (int columnIndex, double x) throws SQLException
- `void updateBigDecimal` (int columnIndex, BigDecimal x) throws SQLException
- `void updateString` (int columnIndex, String x) throws SQLException
- `void updateBytes` (int columnIndex, byte[] x) throws SQLException
- `void updateDate` (int columnIndex, Date x) throws SQLException
- `void updateTime` (int columnIndex, Time x) throws SQLException
- `void updateTimestamp` (int columnIndex, Timestamp x) throws SQLException
- `void updateAsciiStream` (int columnIndex, InputStream x, int length) throws SQLException

- void **updateBinaryStream** (int columnIndex, InputStream x, int length) throws SQLException
- void **updateCharacterStream** (int columnIndex, Reader x, int length) throws SQLException
- void **updateObject** (int columnIndex, Object x, int scaleOrLength) throws SQLException
- void **updateObject** (int columnIndex, Object x) throws SQLException
- void **updateNull** (String columnLabel) throws SQLException
- void **updateBoolean** (String columnLabel, boolean x) throws SQLException
- void **updateByte** (String columnLabel, byte x) throws SQLException
- void **updateShort** (String columnLabel, short x) throws SQLException
- void **updateInt** (String columnLabel, int x) throws SQLException
- void **updateLong** (String columnLabel, long x) throws SQLException
- void **updateFloat** (String columnLabel, float x) throws SQLException
- void **updateDouble** (String columnLabel, double x) throws SQLException
- void **updateBigDecimal** (String columnLabel, BigDecimal x) throws SQLException
- void **updateString** (String columnLabel, String x) throws SQLException
- void **updateBytes** (String columnLabel, byte[] x) throws SQLException
- void **updateDate** (String columnLabel, Date x) throws SQLException
- void **updateTime** (String columnLabel, Time x) throws SQLException
- void **updateTimestamp** (String columnLabel, Timestamp x) throws SQLException
- void **updateAsciiStream** (String columnLabel, InputStream x, int length) throws SQLException
  
- void **updateBinaryStream** (String columnLabel, InputStream x, int length) throws SQLException
- void **updateCharacterStream** (String columnLabel, Reader reader, int length) throws SQLException
- void **updateObject** (String columnLabel, Object x, int scaleOrLength) throws SQLException
  
- void **updateObject** (String columnLabel, Object x) throws SQLException
- void **insertRow** () throws SQLException
- void **updateRow** () throws SQLException
- void **deleteRow** () throws SQLException
- void **refreshRow** () throws SQLException
- void **cancelRowUpdates** () throws SQLException
- void **moveToInsertRow** () throws SQLException
- void **moveToCurrentRow** () throws SQLException
- Statement **getStatement** () throws SQLException
- Object **getObject** (int columnIndex, Map< String, Class<?> > map) throws SQLException
- Ref **getRef** (int columnIndex) throws SQLException
- Blob **getBlob** (int columnIndex) throws SQLException
- Clob **getClob** (int columnIndex) throws SQLException
- Array **getArray** (int columnIndex) throws SQLException
- Object **getObject** (String columnLabel, Map< String, Class<?> > map) throws SQLException
  
- Ref **getRef** (String columnLabel) throws SQLException
- Blob **getBlob** (String columnLabel) throws SQLException
- Clob **getClob** (String columnLabel) throws SQLException
- Array **getArray** (String columnLabel) throws SQLException
- Date **getDate** (int columnIndex, Calendar cal) throws SQLException
- Date **getDate** (String columnLabel, Calendar cal) throws SQLException
- Time **getTime** (int columnIndex, Calendar cal) throws SQLException
- Time **getTime** (String columnLabel, Calendar cal) throws SQLException
- Timestamp **getTimestamp** (int columnIndex, Calendar cal) throws SQLException
- Timestamp **getTimestamp** (String columnLabel, Calendar cal) throws SQLException
- URL **getURL** (int columnIndex) throws SQLException
- URL **getURL** (String columnLabel) throws SQLException
- void **updateRef** (int columnIndex, Ref x) throws SQLException

- void **updateRef** (String columnLabel, Ref x) throws SQLException
- void **updateBlob** (int columnIndex, Blob x) throws SQLException
- void **updateBlob** (String columnLabel, Blob x) throws SQLException
- void **updateClob** (int columnIndex, Clob x) throws SQLException
- void **updateClob** (String columnLabel, Clob x) throws SQLException
- void **updateArray** (int columnIndex, Array x) throws SQLException
- void **updateArray** (String columnLabel, Array x) throws SQLException
- RowId **getRowId** (int columnIndex) throws SQLException
- RowId **getRowId** (String columnLabel) throws SQLException
- void **updateRowId** (int columnIndex, RowId x) throws SQLException
- void **updateRowId** (String columnLabel, RowId x) throws SQLException
- int **getHoldability** () throws SQLException
- boolean **isClosed** () throws SQLException
- void **updateNString** (int columnIndex, String nString) throws SQLException
- void **updateNString** (String columnLabel, String nString) throws SQLException
- void **updateNClob** (int columnIndex, NClob nClob) throws SQLException
- void **updateNClob** (String columnLabel, NClob nClob) throws SQLException
- NClob **getNClob** (int columnIndex) throws SQLException
- NClob **getNClob** (String columnLabel) throws SQLException
- SQLXML **getSQLXML** (int columnIndex) throws SQLException
- SQLXML **getSQLXML** (String columnLabel) throws SQLException
- void **updateSQLXML** (int columnIndex, SQLXML xmlObject) throws SQLException
- void **updateSQLXML** (String columnLabel, SQLXML xmlObject) throws SQLException
- String **getNString** (int columnIndex) throws SQLException
- String **getNString** (String columnLabel) throws SQLException
- Reader **getNCharacterStream** (int columnIndex) throws SQLException
- Reader **getNCharacterStream** (String columnLabel) throws SQLException
- void **updateNCharacterStream** (int columnIndex, Reader x, long length) throws SQLException
  
- void **updateNCharacterStream** (String columnLabel, Reader reader, long length) throws SQLException
- void **updateAsciiStream** (int columnIndex, InputStream x, long length) throws SQLException
  
- void **updateBinaryStream** (int columnIndex, InputStream x, long length) throws SQLException
  
- void **updateCharacterStream** (int columnIndex, Reader x, long length) throws SQLException
  
- void **updateAsciiStream** (String columnLabel, InputStream x, long length) throws SQLException
- void **updateBinaryStream** (String columnLabel, InputStream x, long length) throws SQLException
- void **updateCharacterStream** (String columnLabel, Reader reader, long length) throws SQLException
- void **updateBlob** (int columnIndex, InputStream inputStream, long length) throws SQLException
  
- void **updateBlob** (String columnLabel, InputStream inputStream, long length) throws SQLException
- void **updateClob** (int columnIndex, Reader reader, long length) throws SQLException
- void **updateClob** (String columnLabel, Reader reader, long length) throws SQLException
- void **updateNClob** (int columnIndex, Reader reader, long length) throws SQLException
- void **updateNClob** (String columnLabel, Reader reader, long length) throws SQLException
- void **updateNCharacterStream** (int columnIndex, Reader x) throws SQLException
- void **updateNCharacterStream** (String columnLabel, Reader reader) throws SQLException
  
- void **updateAsciiStream** (int columnIndex, InputStream x) throws SQLException
- void **updateBinaryStream** (int columnIndex, InputStream x) throws SQLException
- void **updateCharacterStream** (int columnIndex, Reader x) throws SQLException

- void **updateAsciiStream** (String columnLabel, InputStream x) throws SQLException
- void **updateBinaryStream** (String columnLabel, InputStream x) throws SQLException
- void **updateCharacterStream** (String columnLabel, Reader reader) throws SQLException
- void **updateBlob** (int columnIndex, InputStream inputStream) throws SQLException
- void **updateBlob** (String columnLabel, InputStream inputStream) throws SQLException
- void **updateClob** (int columnIndex, Reader reader) throws SQLException
- void **updateClob** (String columnLabel, Reader reader) throws SQLException
- void **updateNClob** (int columnIndex, Reader reader) throws SQLException
- void **updateNClob** (String columnLabel, Reader reader) throws SQLException
- boolean **isWrapperFor** (Class<?> iface) throws SQLException

#### Protected Member Functions

- int **lookup** (String sCol)
- int **rangeTest** (int nColIndex) throws SQLException

#### Protected Attributes

- int **m\_nCursor**
- String[] **m\_sColNames**
- Function< T, Integer >[] **m\_oIntDelegates**
- Function< T, String >[] **m\_oStringDelegates**
- Function< T, Double >[] **m\_oDoubleDelegates**
- Function< T, Long >[] **m\_oLongDelegates**
- Function< T, Short >[] **m\_oShortDelegates**
- Function< T, Id >[] **m\_oIdDelegates**

#### Detailed Description

Template class that implements the ResultSet interface with an ArrayList. Many of the functions are not implemented since only a subset of the interface functions were needed.

#### Author

Federal Highway Administration

#### Parameters

<T>	type of object stored by the ResultSet
-----	--

#### Member Function Documentation

*boolean imrcp.store.ImrcpResultSet< T >.absolute (int row) throws SQLException*

NOT IMPLEMENTED

*void imrcp.store.ImrcpResultSet< T >.afterLast () throws SQLException*

NOT IMPLEMENTED

*void imrcp.store.ImrcpResultSet< T >.beforeFirst () throws SQLException*

NOT IMPLEMENTED

*void imrcp.store.ImrcpResultSet< T >.cancelRowUpdates () throws SQLException*

NOT IMPLEMENTED

*void imrcp.store.ImrcpResultSet< T >.clearWarnings () throws SQLException*

NOT IMPLEMENTED

*void imrcp.store.ImrcpResultSet< T >.close () throws SQLException*

Wrapper for **ArrayList#clear()**

*Exceptions*

<i>SQLException</i>
---------------------

*void imrcp.store.ImrcpResultSet< T >.deleteRow () throws SQLException*

NOT IMPLEMENTED

*int imrcp.store.ImrcpResultSet< T >.findColumn (String columnLabel) throws SQLException*

NOT IMPLEMENTED

*boolean imrcp.store.ImrcpResultSet< T >.first () throws SQLException*

NOT IMPLEMENTED

*Array imrcp.store.ImrcpResultSet< T >.getArray (int columnIndex) throws SQLException*

NOT IMPLEMENTED

*Array imrcp.store.ImrcpResultSet< T >.getArray (String columnLabel) throws SQLException*

NOT IMPLEMENTED

*InputStream imrcp.store.ImrcpResultSet< T >.getAsciiStream (int columnIndex) throws SQLException*

NOT IMPLEMENTED

*InputStream imrcp.store.ImrcpResultSet< T >.getAsciiStream (String columnLabel) throws SQLException*

NOT IMPLEMENTED

*BigDecimal imrcp.store.ImrcpResultSet< T >.getBigDecimal (int columnIndex) throws SQLException*

NOT IMPLEMENTED

*BigDecimal imrcp.store.ImrcpResultSet< T >.getBigDecimal (int columnIndex, int scale) throws SQLException*

NOT IMPLEMENTED

*BigDecimal imrcp.store.ImrcpResultSet< T >.getBigDecimal (String columnLabel) throws SQLException*

NOT IMPLEMENTED

*BigDecimal imrcp.store.ImrcpResultSet< T >.getBigDecimal (String columnLabel, int scale) throws SQLException*

NOT IMPLEMENTED

*InputStream imrcp.store.ImrcpResultSet< T >.getBinaryStream (int columnIndex) throws SQLException*

NOT IMPLEMENTED

*InputStream imrcp.store.ImrcpResultSet< T >.getBinaryStream (String columnLabel)  
throws SQLException*

NOT IMPLEMENTED

*Blob imrcp.store.ImrcpResultSet< T >.getBlob (int columnIndex) throws SQLException*  
NOT IMPLEMENTED

*Blob imrcp.store.ImrcpResultSet< T >.getBlob (String columnLabel) throws  
SQLException*

NOT IMPLEMENTED

*boolean imrcp.store.ImrcpResultSet< T >.getBoolean (int columnIndex) throws  
SQLException*

NOT IMPLEMENTED

*boolean imrcp.store.ImrcpResultSet< T >.getBoolean (String columnLabel) throws  
SQLException*

NOT IMPLEMENTED

*byte imrcp.store.ImrcpResultSet< T >.getByte (int columnIndex) throws SQLException*  
NOT IMPLEMENTED

*byte imrcp.store.ImrcpResultSet< T >.getByte (String columnLabel) throws  
SQLException*

NOT IMPLEMENTED

*byte[] imrcp.store.ImrcpResultSet< T >.getBytes (int columnIndex) throws  
SQLException*

NOT IMPLEMENTED

*byte[] imrcp.store.ImrcpResultSet< T >.getBytes (String columnLabel) throws  
SQLException*

NOT IMPLEMENTED

*Reader imrcp.store.ImrcpResultSet< T >.getCharacterStream (int columnIndex)  
throws SQLException*

NOT IMPLEMENTED

*Reader imrcp.store.ImrcpResultSet< T >.getCharacterStream (String columnLabel)  
throws SQLException*

NOT IMPLEMENTED

*Clob imrcp.store.ImrcpResultSet< T >.getClob (int columnIndex) throws SQLException*  
NOT IMPLEMENTED

*Clob imrcp.store.ImrcpResultSet< T >.getClob (String columnLabel) throws  
SQLException*

NOT IMPLEMENTED

*int imrcp.store.ImrcpResultSet< T >.getConcurrency () throws SQLException*  
NOT IMPLEMENTED

*String imrcp.store.ImrcpResultSet< T >.getCursorName () throws SQLException*  
NOT IMPLEMENTED

*Date imrcp.store.ImrcpResultSet< T >.getDate (int columnIndex) throws SQLException*  
NOT IMPLEMENTED

*Date imrcp.store.ImrcpResultSet< T >.getDate (int columnIndex, Calendar cal) throws SQLException*  
NOT IMPLEMENTED

*Date imrcp.store.ImrcpResultSet< T >.getDate (String columnLabel) throws SQLException*  
NOT IMPLEMENTED

*Date imrcp.store.ImrcpResultSet< T >.getDate (String columnLabel, Calendar cal) throws SQLException*  
NOT IMPLEMENTED

*double imrcp.store.ImrcpResultSet< T >.getDouble (int columnIndex) throws SQLException*

Gets the double value for the given column

*Parameters*

<i>columnIndex</i>	1 based column index
--------------------	----------------------

*Returns*

double value for the given column

*Exceptions*

<i>SQLException</i>
---------------------

*double imrcp.store.ImrcpResultSet< T >.getDouble (String columnLabel) throws SQLException*

NOT IMPLEMENTED

*int imrcp.store.ImrcpResultSet< T >.getFetchDirection () throws SQLException*  
NOT IMPLEMENTED

*int imrcp.store.ImrcpResultSet< T >.getFetchSize () throws SQLException*  
NOT IMPLEMENTED

*float imrcp.store.ImrcpResultSet< T >.getFloat (int columnIndex) throws SQLException*  
NOT IMPLEMENTED

*float imrcp.store.ImrcpResultSet< T >.getFloat (String columnLabel) throws SQLException*  
NOT IMPLEMENTED

*int imrcp.store.ImrcpResultSet< T >.getHoldability () throws SQLException*  
NOT IMPLEMENTED

*int imrcp.store.ImrcpResultSet< T >.getInt (int columnIndex) throws SQLException*

Gets the int value for the given column

*Parameters*

<i>columnIndex</i>	1 based column index
--------------------	----------------------

*Returns*

int value for the given column

*Exceptions*

<i>SQLException</i>
---------------------

*int imrcp.store.ImrcpResultSet< T >.getInt (String columnLabel) throws SQLException*

NOT IMPLEMENTED

*int[] imrcp.store.ImrcpResultSet< T >.getIntArray (int columnIndex) throws SQLException*

NOT IMPLEMENTED

*long imrcp.store.ImrcpResultSet< T >.getLong (int columnIndex) throws SQLException*

Gets the long value for the given column

*Parameters*

<i>columnIndex</i>	1 based column index
--------------------	----------------------

*Returns*

*Exceptions*

<i>SQLException</i>
---------------------

*long imrcp.store.ImrcpResultSet< T >.getLong (String columnLabel) throws SQLException*

NOT IMPLEMENTED

*ResultSetMetaData imrcp.store.ImrcpResultSet< T >.getMetaData () throws SQLException*

NOT IMPLEMENTED

*Reader imrcp.store.ImrcpResultSet< T >.getNCharacterStream (int columnIndex) throws SQLException*

NOT IMPLEMENTED

*Reader imrcp.store.ImrcpResultSet< T >.getNCharacterStream (String columnLabel) throws SQLException*

NOT IMPLEMENTED

*NClob imrcp.store.ImrcpResultSet< T >.getNClob (int columnIndex) throws SQLException*

NOT IMPLEMENTED

*NClob imrcp.store.ImrcpResultSet< T >.getNClob (String columnLabel) throws SQLException*

NOT IMPLEMENTED

*String imrcp.store.ImrcpResultSet< T >.getString (int columnIndex) throws SQLException*

NOT IMPLEMENTED

*String imrcp.store.ImrcpResultSet< T >.getString (String columnLabel) throws SQLException*

NOT IMPLEMENTED

*Object imrcp.store.ImrcpResultSet< T >.getObject (int columnIndex) throws SQLException*

NOT IMPLEMENTED

*Object imrcp.store.ImrcpResultSet< T >.getObject (int columnIndex, Map< String, Class<?> > map) throws SQLException*

NOT IMPLEMENTED

*Object imrcp.store.ImrcpResultSet< T >.getObject (String columnLabel) throws SQLException*

NOT IMPLEMENTED

*Object imrcp.store.ImrcpResultSet< T >.getObject (String columnLabel, Map< String, Class<?> > map) throws SQLException*

NOT IMPLEMENTED

*Ref imrcp.store.ImrcpResultSet< T >.getRef (int columnIndex) throws SQLException*

NOT IMPLEMENTED

*Ref imrcp.store.ImrcpResultSet< T >.getRef (String columnLabel) throws SQLException*

NOT IMPLEMENTED

*int imrcp.store.ImrcpResultSet< T >.getRow () throws SQLException*

NOT IMPLEMENTED

*RowId imrcp.store.ImrcpResultSet< T >.getRowId (int columnIndex) throws SQLException*

NOT IMPLEMENTED

*RowId imrcp.store.ImrcpResultSet< T >.getRowId (String columnLabel) throws SQLException*

NOT IMPLEMENTED

*short imrcp.store.ImrcpResultSet< T >.getShort (int columnIndex) throws SQLException*

Gets the short value for the given column

#### Parameters

<i>columnIndex</i>	1 based column index
--------------------	----------------------

*Returns*

short value for the given column

*Exceptions*

<i>SQLException</i>
---------------------

*short imrcp.store.ImrcpResultSet< T >.getShort (String columnLabel) throws SQLException*

NOT IMPLEMENTED

*SQLXML imrcp.store.ImrcpResultSet< T >.getSQLXML (int columnIndex) throws SQLException*

NOT IMPLEMENTED

*SQLXML imrcp.store.ImrcpResultSet< T >.getSQLXML (String columnLabel) throws SQLException*

NOT IMPLEMENTED

*Statement imrcp.store.ImrcpResultSet< T >.getStatement () throws SQLException*

NOT IMPLEMENTED

*String imrcp.store.ImrcpResultSet< T >.getString (int columnIndex) throws SQLException*

Gets the String value for the given column

*Parameters*

<i>columnIndex</i>	1 based column index
--------------------	----------------------

*Returns*

String value for the given column

*Exceptions*

<i>SQLException</i>
---------------------

*String imrcp.store.ImrcpResultSet< T >.getString (String columnLabel) throws SQLException*

NOT IMPLEMENTED

*Time imrcp.store.ImrcpResultSet< T >.getTime (int columnIndex) throws SQLException*

NOT IMPLEMENTED

*Time imrcp.store.ImrcpResultSet< T >.getTime (int columnIndex, Calendar cal) throws SQLException*

NOT IMPLEMENTED

*Time imrcp.store.ImrcpResultSet< T >.getTime (String columnLabel) throws SQLException*

NOT IMPLEMENTED

*Time imrcp.store.ImrcpResultSet< T >.getTime (String columnLabel, Calendar cal)  
throws SQLException*

NOT IMPLEMENTED

*Timestamp imrcp.store.ImrcpResultSet< T >.getTimestamp (int columnIndex) throws  
SQLException*

NOT IMPLEMENTED

*Timestamp imrcp.store.ImrcpResultSet< T >.getTimestamp (int columnIndex,  
Calendar cal) throws SQLException*

NOT IMPLEMENTED

*Timestamp imrcp.store.ImrcpResultSet< T >.getTimestamp (String columnLabel)  
throws SQLException*

NOT IMPLEMENTED

*Timestamp imrcp.store.ImrcpResultSet< T >.getTimestamp (String columnLabel,  
Calendar cal) throws SQLException*

NOT IMPLEMENTED

*int imrcp.store.ImrcpResultSet< T >.getType () throws SQLException*

NOT IMPLEMENTED

*InputStream imrcp.store.ImrcpResultSet< T >.getUnicodeStream (int columnIndex)  
throws SQLException*

NOT IMPLEMENTED

*InputStream imrcp.store.ImrcpResultSet< T >.getUnicodeStream (String columnLabel)  
throws SQLException*

NOT IMPLEMENTED

*URL imrcp.store.ImrcpResultSet< T >.getURL (int columnIndex) throws SQLException*

NOT IMPLEMENTED

*URL imrcp.store.ImrcpResultSet< T >.getURL (String columnLabel) throws  
SQLException*

NOT IMPLEMENTED

*SQLWarning imrcp.store.ImrcpResultSet< T >.getWarnings () throws SQLException*

NOT IMPLEMENTED

*void imrcp.store.ImrcpResultSet< T >.insertRow () throws SQLException*

NOT IMPLEMENTED

*boolean imrcp.store.ImrcpResultSet< T >.isAfterLast () throws SQLException*

NOT IMPLEMENTED

*boolean imrcp.store.ImrcpResultSet< T >.isBeforeFirst () throws SQLException*

NOT IMPLEMENTED

*boolean imrcp.store.ImrcpResultSet< T >.isClosed () throws SQLException*

NOT IMPLEMENTED

*boolean imrcp.store.ImrcpResultSet< T >.isFirst () throws SQLException*  
NOT IMPLEMENTED

*boolean imrcp.store.ImrcpResultSet< T >.isLast () throws SQLException*  
NOT IMPLEMENTED

*boolean imrcp.store.ImrcpResultSet< T >.isWrapperFor (Class<?> iface) throws SQLException*  
NOT IMPLEMENTED

*boolean imrcp.store.ImrcpResultSet< T >.last () throws SQLException*  
NOT IMPLEMENTED

*int imrcp.store.ImrcpResultSet< T >.lookup (String sCol) [protected]*  
Looks up the index for the given column name.

*Parameters*

<i>sCol</i>	Column to look up
-------------	-------------------

*Returns*

Index that corresponds to the column name. If the column name does not exist -1

*void imrcp.store.ImrcpResultSet< T >.moveToCurrentRow () throws SQLException*  
NOT IMPLEMENTED

*void imrcp.store.ImrcpResultSet< T >.moveToInsertRow () throws SQLException*  
NOT IMPLEMENTED

*boolean imrcp.store.ImrcpResultSet< T >.next () throws SQLException*  
Advances the cursor and checks if there are any more rows available

*Returns*

true if the cursor is not past the end of the list

*Exceptions*

<i>SQLException</i>
---------------------

*boolean imrcp.store.ImrcpResultSet< T >.previous () throws SQLException*  
NOT IMPLEMENTED

*int imrcp.store.ImrcpResultSet< T >.rangeTest (int nColIndex) throws SQLException [protected]*

Tests if the cursor and given column index are in range and returns the correct index for the column. Indices are 1 based for ResultSets, since the underlaying list is zero based the correct index is nColIndex - 1

*Parameters*

<i>nColIndex</i>	1 based column index
------------------	----------------------

*Returns*

0 based column index

*Exceptions*

<i>SQLException</i>	if the cursor or column index is out of range
---------------------	---

*void imrcp.store.ImrcpResultSet< T >.refreshRow () throws SQLException*

NOT IMPLEMENTED

*boolean imrcp.store.ImrcpResultSet< T >.relative (int rows) throws SQLException*

NOT IMPLEMENTED

*boolean imrcp.store.ImrcpResultSet< T >.rowDeleted () throws SQLException*

NOT IMPLEMENTED

*boolean imrcp.store.ImrcpResultSet< T >.rowInserted () throws SQLException*

NOT IMPLEMENTED

*boolean imrcp.store.ImrcpResultSet< T >.rowUpdated () throws SQLException*

NOT IMPLEMENTED

*void imrcp.store.ImrcpResultSet< T >.setFetchDirection (int direction) throws SQLException*

NOT IMPLEMENTED

*void imrcp.store.ImrcpResultSet< T >.setFetchSize (int rows) throws SQLException*

NOT IMPLEMENTED

*void imrcp.store.ImrcpResultSet< T >.updateArray (int columnIndex, Array x) throws SQLException*

NOT IMPLEMENTED

*void imrcp.store.ImrcpResultSet< T >.updateArray (String columnName, Array x) throws SQLException*

NOT IMPLEMENTED

*void imrcp.store.ImrcpResultSet< T >.updateAsciiStream (int columnIndex, InputStream x) throws SQLException*

NOT IMPLEMENTED

*void imrcp.store.ImrcpResultSet< T >.updateAsciiStream (int columnIndex, InputStream x, int length) throws SQLException*

NOT IMPLEMENTED

*void imrcp.store.ImrcpResultSet< T >.updateAsciiStream (int columnIndex, InputStream x, long length) throws SQLException*

NOT IMPLEMENTED

*void imrcp.store.ImrcpResultSet< T >.updateAsciiStream (String columnName, InputStream x) throws SQLException*

NOT IMPLEMENTED

*void imrcp.store.ImrcpResultSet< T >.updateAsciiStream (String columnName, InputStream x, int length) throws SQLException*

NOT IMPLEMENTED

```
void imrcp.store.ImrcpResultSet< T >.updateAsciiStream (String columnLabel,  
InputStream x, long length) throws SQLException  
    NOT IMPLEMENTED  
  
void imrcp.store.ImrcpResultSet< T >.updateBigDecimal (int columnIndex, BigDecimal  
x) throws SQLException  
    NOT IMPLEMENTED  
  
void imrcp.store.ImrcpResultSet< T >.updateBigDecimal (String columnLabel,  
BigDecimal x) throws SQLException  
    NOT IMPLEMENTED  
  
void imrcp.store.ImrcpResultSet< T >.updateBinaryStream (int columnIndex,  
InputStream x) throws SQLException  
    NOT IMPLEMENTED  
  
void imrcp.store.ImrcpResultSet< T >.updateBinaryStream (int columnIndex,  
InputStream x, int length) throws SQLException  
    NOT IMPLEMENTED  
  
void imrcp.store.ImrcpResultSet< T >.updateBinaryStream (int columnIndex,  
InputStream x, long length) throws SQLException  
    NOT IMPLEMENTED  
  
void imrcp.store.ImrcpResultSet< T >.updateBinaryStream (String columnLabel,  
InputStream x) throws SQLException  
    NOT IMPLEMENTED  
  
void imrcp.store.ImrcpResultSet< T >.updateBinaryStream (String columnLabel,  
InputStream x, int length) throws SQLException  
    NOT IMPLEMENTED  
  
void imrcp.store.ImrcpResultSet< T >.updateBinaryStream (String columnLabel,  
InputStream x, long length) throws SQLException  
    NOT IMPLEMENTED  
  
void imrcp.store.ImrcpResultSet< T >.updateBlob (int columnIndex, Blob x) throws  
SQLException  
    NOT IMPLEMENTED  
  
void imrcp.store.ImrcpResultSet< T >.updateBlob (int columnIndex, InputStream  
inputStream) throws SQLException  
    NOT IMPLEMENTED  
  
void imrcp.store.ImrcpResultSet< T >.updateBlob (int columnIndex, InputStream  
inputStream, long length) throws SQLException  
    NOT IMPLEMENTED  
  
void imrcp.store.ImrcpResultSet< T >.updateBlob (String columnLabel, Blob x) throws  
SQLException  
    NOT IMPLEMENTED
```

```
void imrcp.store.ImrcpResultSet< T >.updateBlob (String columnLabel, InputStream
InputStream) throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateBlob (String columnLabel, InputStream
InputStream, long length) throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateBoolean (int columnIndex, boolean x)
throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateBoolean (String columnLabel, boolean x)
throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateByte (int columnIndex, byte x) throws
SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateByte (String columnLabel, byte x) throws
SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateBytes (int columnIndex, byte[] x) throws
SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateBytes (String columnLabel, byte[] x)
throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateCharacterStream (int columnIndex,
Reader x) throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateCharacterStream (int columnIndex,
Reader x, int length) throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateCharacterStream (int columnIndex,
Reader x, long length) throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateCharacterStream (String columnLabel,
Reader reader) throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateCharacterStream (String columnLabel,
Reader reader, int length) throws SQLException
    NOT IMPLEMENTED
```

```
void imrcp.store.ImrcpResultSet< T >.updateCharacterStream (String columnLabel,  
Reader reader, long length) throws SQLException  
    NOT IMPLEMENTED  
  
void imrcp.store.ImrcpResultSet< T >.updateClob (int columnIndex, Clob x) throws  
SQLException  
    NOT IMPLEMENTED  
  
void imrcp.store.ImrcpResultSet< T >.updateClob (int columnIndex, Reader reader)  
throws SQLException  
    NOT IMPLEMENTED  
  
void imrcp.store.ImrcpResultSet< T >.updateClob (int columnIndex, Reader reader,  
long length) throws SQLException  
    NOT IMPLEMENTED  
  
void imrcp.store.ImrcpResultSet< T >.updateClob (String columnLabel, Clob x) throws  
SQLException  
    NOT IMPLEMENTED  
  
void imrcp.store.ImrcpResultSet< T >.updateClob (String columnLabel, Reader reader)  
throws SQLException  
    NOT IMPLEMENTED  
  
void imrcp.store.ImrcpResultSet< T >.updateClob (String columnLabel, Reader reader,  
long length) throws SQLException  
    NOT IMPLEMENTED  
  
void imrcp.store.ImrcpResultSet< T >.updateDate (int columnIndex, Date x) throws  
SQLException  
    NOT IMPLEMENTED  
  
void imrcp.store.ImrcpResultSet< T >.updateDate (String columnLabel, Date x)  
throws SQLException  
    NOT IMPLEMENTED  
  
void imrcp.store.ImrcpResultSet< T >.updateDouble (int columnIndex, double x)  
throws SQLException  
    NOT IMPLEMENTED  
  
void imrcp.store.ImrcpResultSet< T >.updateDouble (String columnLabel, double x)  
throws SQLException  
    NOT IMPLEMENTED  
  
void imrcp.store.ImrcpResultSet< T >.updateFloat (int columnIndex, float x) throws  
SQLException  
    NOT IMPLEMENTED  
  
void imrcp.store.ImrcpResultSet< T >.updateFloat (String columnLabel, float x)  
throws SQLException  
    NOT IMPLEMENTED
```

```
void imrcp.store.ImrcpResultSet< T >.updateInt (int columnIndex, int x) throws
SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateInt (String columnLabel, int x) throws
SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateLong (int columnIndex, long x) throws
SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateLong (String columnLabel, long x) throws
SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateNCharacterStream (int columnIndex,
Reader x) throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateNCharacterStream (int columnIndex,
Reader x, long length) throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateNCharacterStream (String columnLabel,
Reader reader) throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateNCharacterStream (String columnLabel,
Reader reader, long length) throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateNClob (int columnIndex, NClob nClob)
throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateNClob (int columnIndex, Reader reader)
throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateNClob (int columnIndex, Reader reader,
long length) throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateNClob (String columnLabel, NClob nClob)
throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateNClob (String columnLabel, Reader reader)
throws SQLException
    NOT IMPLEMENTED
```

```
void imrcp.store.ImrcpResultSet< T >.updateNClob (String columnLabel, Reader reader, long length) throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateNString (int columnIndex, String nString)
throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateNString (String columnLabel, String nString) throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateNull (int columnIndex) throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateNull (String columnLabel) throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateObject (int columnIndex, Object x)
throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateObject (int columnIndex, Object x, int scaleOrLength) throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateObject (String columnLabel, Object x)
throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateObject (String columnLabel, Object x, int scaleOrLength) throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateRef (int columnIndex, Ref x) throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateRef (String columnLabel, Ref x) throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateRow () throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateRowId (int columnIndex, RowId x) throws SQLException
    NOT IMPLEMENTED
```

```
void imrcp.store.ImrcpResultSet< T >.updateRowId (String columnLabel, RowId x)
throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateShort (int columnIndex, short x) throws
SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateShort (String columnLabel, short x)
throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateSQLXML (int columnIndex, SQLXML
xmlObject) throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateSQLXML (String columnLabel, SQLXML
xmlObject) throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateString (int columnIndex, String x) throws
SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateString (String columnLabel, String x)
throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateTime (int columnIndex, Time x) throws
SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateTime (String columnLabel, Time x)
throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateTimestamp (int columnIndex, Timestamp
x) throws SQLException
    NOT IMPLEMENTED

void imrcp.store.ImrcpResultSet< T >.updateTimestamp (String columnLabel,
Timestamp x) throws SQLException
    NOT IMPLEMENTED

boolean imrcp.store.ImrcpResultSet< T >.wasNull () throws SQLException
    NOT IMPLEMENTED
```

---

## Member Data Documentation

*int imrcp.store.ImrcpResultSet< T >.m\_nCursor [protected]*

Keeps track of the current row(index) of the ArrayList

*Function<T, Double> [] imrcp.store.ImrcpResultSet< T >.m\_oDoubleDelegates [protected]*

Child classes implement an interface for the specific T to get double values from the columns

*Function<T, Id> [] imrcp.store.ImrcpResultSet< T >.m\_oldDelegates [protected]*

Child classes implement an interface for the specific T to get Id values from the columns

*Function<T, Integer> [] imrcp.store.ImrcpResultSet< T >.m\_oIntDelegates [protected]*

Child classes implement an interface for the specific T to get int values from the columns

*Function<T, Long> [] imrcp.store.ImrcpResultSet< T >.m\_oLongDelegates [protected]*

Child classes implement an interface for the specific T to get long values from the columns

*Function<T, Short> [] imrcp.store.ImrcpResultSet< T >.m\_oShortDelegates [protected]*

Child classes implement an interface for the specific T to get short values from the columns

*Function<T, String> [] imrcp.store.ImrcpResultSet< T >.m\_oStringDelegates [protected]*

Child classes implement an interface for the specific T to get String values from the columns

*String [] imrcp.store.ImrcpResultSet< T >.m\_sColNames [protected]*

Contains the column names of the objects stored in the list

---

*The documentation for this class was generated from the following file:*

- store/ImrcpResultSet.java

---

## imrcp.collect.Inrix Class Reference

### Public Member Functions

- void **reset** ()
- boolean **start** () throws Exception
- void **execute** ()
- void **getToken** ()

### Additional Inherited Members

---

## Detailed Description

Generic collector for **Inrix** data

### Author

Federal Highway Administration

---

## Member Function Documentation

### `void imrcp.collect.Inrix.execute ()`

Queries the **Inrix** Segment Speed api by generating a URL for each of the configured map tiles at the configured zoom level and saves the resulting JSON objects in a single JSON array on disk

Reimplemented from **imrcp.system.BaseBlock** (p.95).

### `void imrcp.collect.Inrix.getToken ()`

Queries for and stores in memory a security token from the **Inrix** api that is used for authenticating data queries. The response also tells when the token will expire so **imrcp.system.Scheduling** schedules this function to run 5 minutes before the token expires.

### `void imrcp.collect.Inrix.reset ()`

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.collect.Collector** (p.114).

### `boolean imrcp.collect.Inrix.start () throws Exception`

Checks if **Inrix#m\_sQueryString** has an even length (throws an Exception if the length is odd), calls **Inrix#getToken()** and sets a schedule to execute on a fixed interval.

#### Returns

true if no exceptions are thrown

#### Exceptions

<code>Exception</code>
------------------------

Reimplemented from **imrcp.system.BaseBlock** (p.99).

---

*The documentation for this class was generated from the following file:*

- `collect/Inrix.java`

---

## imrcp.comp.InrixComp Class Reference

### Public Member Functions

- `void reset ()`
- `void process (String[] sMessage)`

## Protected Member Functions

- void **parseFile** (String sFilename)

## Protected Attributes

- int **m\_nFreq**
- int **m\_nSrcFreq**
- **FilenameFormatter m\_oDest**
- **FilenameFormatter m\_oSrc**

## Additional Inherited Members

---

### Detailed Description

Processes the files created from Inrix data downloads and converts them to IMRCP's traffic speed file format.

#### Author

Federal Highway Administration

---

### Member Function Documentation

#### *void imrcp.comp.InrixComp.parseFile (String sFilename) [protected]*

Parses the JSON file created by **imrcp.collect.Inrix** and writes speed data to the rolling binary traffic speed file.

##### Parameters

<i>sFilename</i>	JSON file to parse
------------------	--------------------

#### *void imrcp.comp.InrixComp.process (String[] sMessage)*

Called when a message from **imrcp.system.Directory** is received. If the message is "file download" **InrixComp#parseFile (java.lang.String)** is called.

##### Parameters

<i>sMessage</i>	[BaseBlock message is from, message name, file to process]
-----------------	--

Reimplemented from **imrcp.system.BaseBlock** (p.97).

#### *void imrcp.comp.InrixComp.reset ()*

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.system.BaseBlock** (p.98).

---

### Member Data Documentation

#### *int imrcp.comp.InrixComp.m\_nFreq [protected]*

How often a file should be made to store speed obs

#### *int imrcp.comp.InrixComp.m\_nSrcFreq [protected]*

How often source files are expected to downloaded

*FilenameFormatter imrcp.comp.InrixComp.m\_oDest [protected]*

Object used to create time dependent file names to save on disk

*FilenameFormatter imrcp.comp.InrixComp.m\_oSrc [protected]*

Object used to create time dependent file names of source files

---

*The documentation for this class was generated from the following file:*

- comp/InrixComp.java

---

[imrcp.store.ImrcpCapResultSet.IntDelegate Interface Reference](#)

---

**Detailed Description**

Necessary template definition for **ImrcpResultSet#m\_oIntDelegates**

---

The documentation for this interface was generated from the following file:

- store/ImrcpCapResultSet.java

---

[imrcp.store.ImrcpEventResultSet.IntDelegate Interface Reference](#)

---

**Detailed Description**

Necessary template definition for **ImrcpResultSet#m\_oIntDelegates**

---

The documentation for this interface was generated from the following file:

- store/ImrcpEventResultSet.java

---

[imrcp.store.ImrcpObsResultSet.IntDelegate Interface Reference](#)

---

**Detailed Description**

Necessary template definition for **ImrcpResultSet#m\_oIntDelegates**

---

The documentation for this interface was generated from the following file:

- store/ImrcpObsResultSet.java

---

[imrcp.system.IntKeyValue< T > Class Template Reference](#)

**Public Member Functions**

- **IntKeyValue ()**

- **IntKeyValue** (int nKey, T oT)
  - final void **setKey** (int nKey)
  - int **getKey** ()
  - T **value** ()
  - int **compareTo** (IntKeyValue oRhs)
- 

### Detailed Description

Maps an integer key to a value of templated type T, that can be ordered based upon this key.  
Implements Comparable<IntKeyValue> to enforce an ordering upon IntKeyValue objects.

#### Parameters

<code>&lt;T&gt;</code>	template type - must be specified when a new instance of IntKeyValue is created.
------------------------	--

---

### Constructor & Destructor Documentation

`imrcp.system.IntKeyValue< T >.IntKeyValue ()`

#### Default Constructor

Creates new instances of IntKeyValue

`imrcp.system.IntKeyValue< T >.IntKeyValue (int nKey, T oT)`

#### Constructor

Initializes attributes of new instances to the provided values.

#### Parameters

<code>nKey</code>	new value for the key attribute.
<code>oT</code>	new value to be mapped.

---

### Member Function Documentation

`int imrcp.system.IntKeyValue< T >.compareTo (IntKeyValue< T > oRhs)`

Enforces an ordering on IntKeyValue objects based off the key.

#### Parameters

<code>oRhs</code>	the object to compare to <i>this</i>
-------------------	--------------------------------------

#### Returns

0 if the keys match. &lt 0 means *this* key is smaller.

`int imrcp.system.IntKeyValue< T >.getKey ()`

#### Returns

`final void imrcp.system.IntKeyValue< T >.setKey (int nKey)`

**Mutator**

*Parameters*

<code>nKey</code>	sets the key attribute to the supplied value.
-------------------	---

`T imrcp.system.IntKeyValue< T >.value ()`

**Accessor**

*Returns*

the value contained by the `IntKeyValue` instance.

---

*The documentation for this class was generated from the following file:*

- `system/IntKeyValue.java`
- 

## imrcp.system.Introsort Class Reference

### Static Public Member Functions

- `static< T extends Comparable<? super T > void usort (List< T > iList)`
  - `static< T > void usort (List< T > iList, Comparator< T > iCompare)`
  - `static< T > void usort (List< T > iList, Comparator< T > iCompare, int nBegin, int nEnd)`
  - `static< T > int binarySearch (List< T > iList, T oKey, Comparator< T > iCompare)`
  - `static< T > int binarySearch (List< T > iList, T oKey, int nLow, int nHigh, Comparator< T > iCompare)`
  - `static int floor (int nValue, int nPrecision)`
- 

### Detailed Description

**Introsort** algorithm. Modified quicksort algorithm, that attempts to avoid the degenerate case when supplied a nearly-sorted input. The modification is that when this degenerate case is detected, heap-sort and insertion sort take over to avoid this worst case. Operates in the worst case  $O(n \lg n)$ .

This class also provides search methods, and number manipulators.

---

### Member Function Documentation

`static< T > int imrcp.system.Introsort.binarySearch (List< T > iList, T oKey, Comparator< T > iCompare) [static]`

Wraps `Introsort#binarySearch(List, Object, int, int, Comparator)`, providing a high and low value that enclose the entire list.

*Parameters*

<code>&lt;T&gt;</code>	templated type.
<code>iList</code>	list to search, sorted in ascending order.
<code>oKey</code>	item to search for.
<code>iCompare</code>	comparator the list items are ordered by.

### Returns

negative value if the key isn't found, otherwise the index of the key in the provided list.

**static< T > int imrcp.system.Introsort.binarySearch (List< T > iList, T oKey, int nLow, int nHigh, Comparator< T > iCompare) [static]**

Binary search algorithm. Searches the provided array for the supplied key.

### Parameters

<T>	templated type.
iList	list to search, sorted in ascending order.
oKey	item to search for.
nLow	low value for midpoint calculation.
nHigh	high value for midpoint calculation.
iCompare	comparator the list items are ordered by.

### Returns

negative value if the key isn't found, otherwise the index of the key in the provided list.

**static int imrcp.system.Introsort.floor (int nValue, int nPrecision) [static]**

Maps the provided value to the nearest multiple of the supplied precision, that's less than the value.

Ex: given a value of 17, and a grid size (precision) of 5, the value would be mapped into index 15.

### Parameters

nValue	the value to floor
nPrecision	the grid size.

### Returns

the floored index.

**static< T extends Comparable<? super T > void imrcp.system.Introsort.usort (List< T > iList) [static]**

Wraps **usort(java.util.List, java.util.Comparator)** creating a Comparator using a lambda expression to pass into the function.

### Parameters

<T>	template type
iList	list to sort

**static< T > void imrcp.system.Introsort.usort (List< T > iList, Comparator< T > iCompare) [static]**

Wraps **Introsort#usort(List, Comparator, int, int)** passing the appropriate begin-end values.

### Parameters

<T>	template type.
iList	list to sort.
iCompare	comparator to sort the list by.

`static< T > void imrcp.system.Introsort.usort (List< T > iList, Comparator< T > iCompare, int nBegin, int nEnd) [static]`

Calculates a recursion depth and uses introsort for that depth, then uses insertion sort for the remaining unordered portion of the list.

*Parameters*

<code>&lt;T&gt;</code>	template type.
<code>iList</code>	list to sort.
<code>iCompare</code>	comparator to sort list by.
<code>nBegin</code>	lower index of the section of the list to sort.
<code>nEnd</code>	upper index of the section of the list to sort.

---

*The documentation for this class was generated from the following file:*

- system/Introsort.java
- 

## imrcp.system.IRunTarget< E > Interface Template Reference

### Public Member Functions

- `void run (E e)`
- 

### Detailed Description

Runnable target interface

*Parameters*

<code>&lt;E&gt;</code>	Template type. Implementations must specify a concrete type at declaration time in place of type: T.
------------------------	--

*Author*

Federal Highway Administration

*Version*

1.0

---

## Member Function Documentation

`void imrcp.system.IRunTarget< E >.run (E e)`

Run method for target object.

*Parameters*

<code>e</code>	target object to run
----------------	----------------------

---

*The documentation for this interface was generated from the following file:*

- system/IRunTarget.java
-

---

## imrcp.collect.LAc2c Class Reference

### Public Member Functions

- void **reset** ()
- boolean **start** ()
- void **execute** ()

### Additional Inherited Members

---

#### Detailed Description

Generic collector for xml feeds from LADOTD's center to center interface

#### Author

Federal Highway Administration

---

#### Member Function Documentation

##### `void imrcp.collect.LAc2c.execute ()`

First downloads the list of available files from the LADOTD c2c server and downloads the configured file if an updated version exists.

Reimplemented from **imrcp.system.BaseBlock** (*p.95*).

##### `void imrcp.collect.LAc2c.reset ()`

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.collect.Collector** (*p.114*).

##### `boolean imrcp.collect.LAc2c.start ()`

Sets a schedule to execute on a fixed interval.

#### Returns

true if no exceptions are thrown

Reimplemented from **imrcp.system.BaseBlock** (*p.99*).

---

*The documentation for this class was generated from the following file:*

- collect/LAc2c.java
- 

---

## imrcp.comp.LAc2cDetectorsComp Class Reference

### Classes

- class **Parser**

## Public Member Functions

- void **reset ()**
- void **process (String[] sMessage)**

## Protected Member Functions

- void **rollupFiles (long lTimestamp, boolean bDeleteDest)**

## Protected Attributes

- int **m\_nRollup**
- int **m\_nOffset**
- int **m\_nSrcRange**
- int **m\_nFileFrequency**
- int **m\_nRange**
- **FilenameFormatter m\_oDest**
- **FilenameFormatter m\_oSrc**

## Additional Inherited Members

---

### Detailed Description

Processes the traffic xml files downloaded from LADOTD's c2c feed and converts them to IMRCP's traffic speed file format.

#### Author

Federal Highway Administration

---

## Member Function Documentation

### *void imrcp.comp.LAc2cDetectorsComp.process (String[] sMessage)*

Called when a message from **imrcp.system.Directory** is received. If the message is "file download" the function checks if it is the correct time to rollup source observation files into an average observation, if so calls **LAc2cDetectorsComp#rollupFiles (long, boolean)**

#### Parameters

<i>sMessage</i>	[BaseBlock message is from, message name]
-----------------	---

Reimplemented from **imrcp.system.BaseBlock** (p.97).

### *void imrcp.comp.LAc2cDetectorsComp.reset ()*

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.system.BaseBlock** (p.98).

### *void imrcp.comp.LAc2cDetectorsComp.rollupFiles (long lTimestamp, boolean bDeleteDest) [protected]*

Searches for source files within the rollup period of the given timestamp and converts the observations into an average that is written to the rolling binary traffic speed file.

#### Parameters

<i>lTimestamp</i>	Timestamp to start searching backwards from for files to rollup.
<i>bDeleteDest</i>	flag to delete existing data files

---

## Member Data Documentation

*int imrcp.comp.LAc2cDetectorsComp.m\_nFileFrequency [protected]*

How often a file should be made to store speed obs

*int imrcp.comp.LAc2cDetectorsComp.m\_nOffset [protected]*

Offset in milliseconds for when a source file is valid from when it is downloaded. For example an offset of 60000 means files with timestamps of 2022-07-28 14:00 are downloaded at 2022-07-28 14:01

*int imrcp.comp.LAc2cDetectorsComp.m\_nRange [protected]*

Number of milliseconds the observation file is valid

*int imrcp.comp.LAc2cDetectorsComp.m\_nRollup [protected]*

Range in milliseconds to combine the source 1 minute files into an averaged observation

*int imrcp.comp.LAc2cDetectorsComp.m\_nSrcRange [protected]*

Time in milliseconds source files are valid for

*FilenameFormatter imrcp.comp.LAc2cDetectorsComp.m\_oDest [protected]*

Object used to create time dependent file names to save on disk

*FilenameFormatter imrcp.comp.LAc2cDetectorsComp.m\_oSrc [protected]*

Object used to create time dependent file names of source files

---

*The documentation for this class was generated from the following file:*

- comp/LAc2cDetectorsComp.java
- 

---

## imrcp.comp.LAc2cEventsComp Class Reference

### Public Member Functions

- void **reset** ()

### Protected Member Functions

- long **getEvents** (String sFile, long lTime) throws Exception

### Additional Inherited Members

---

#### Detailed Description

Processes the event xml files downloaded from LADOTD's c2c feed and converts them to IMRCP's incident and work zone .csv file format

#### Author

Federal Highway Administration

---

## Member Function Documentation

`long imrcp.comp.LAc2cEventsComp.getEvents (String sFile, long lTime) throws Exception [protected]`

Wrapper for `LAc2cEventsComp.Parser#parse (java.io.InputStream)`

### Parameters

<code>sFile</code>	File name of LADOTD's xml event feed
<code>lTime</code>	Start time of the file

### Returns

The time the file was last updated

### Exceptions

<code>Exception</code>
------------------------

Reimplemented from `imrcp.comp.EventComp` (*p.204*).

`void imrcp.comp.LAc2cEventsComp.reset ()`

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from `imrcp.comp.EventComp` (*p.204*).

---

*The documentation for this class was generated from the following file:*

- `comp/LAc2cEventsComp.java`
- 

## imrcp.collect.LADOTD511 Class Reference

### Public Member Functions

- `boolean start () throws Exception`
- `void reset ()`
- `void execute ()`

### Additional Inherited Members

---

### Detailed Description

**Collector** for the **LADOTD511** system. Used to collect both the line and point files.

### Author

Federal Highway Administration

---

## Member Function Documentation

`void imrcp.collect.LADOTD511.execute ()`

Logs into the **LADOTD511** system and downloads the configured filename and saves it to disk.

Reimplemented from **imrcp.system.BaseBlock** (p.95).

***void imrcp.collect.LADOTD511.reset ()***

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.system.BaseBlock** (p.98).

***boolean imrcp.collect.LADOTD511.start () throws Exception***

Sets a schedule to execute on a fixed interval.

*Returns*

true if no exceptions are thrown

*Exceptions*

<i>Exception</i>	
------------------	--

Reimplemented from **imrcp.system.BaseBlock** (p.99).

---

*The documentation for this class was generated from the following file:*

- collect/LADOTD511.java
- 

## imrcp.store.LADOTD511Event Class Reference

### Public Member Functions

- **void close (long lTime, SimpleDateFormat oSdf)**

### Additional Inherited Members

---

#### Detailed Description

Represents an event received from the LADOTD 511 system.

#### Author

Federal Highway Administration

---

#### Member Function Documentation

***void imrcp.store.LADOTD511Event.close (long lTime, SimpleDateFormat oSdf)***

Nothing needs to be done to close this kind of event so this function does nothing.

Reimplemented from **imrcp.store.EventObs** (p.207).

---

*The documentation for this class was generated from the following file:*

- store/LADOTD511Event.java
-

## imrcp.comp.LADOTD511Line Class Reference

### Public Member Functions

- void **reset** ()
- boolean **start** () throws Exception

### Protected Member Functions

- long **getEvents** (String sFile, long lTime) throws Exception

### Protected Attributes

- String **m\_sRollingFile**
- ArrayList< long[]> **m\_oUpdateTimes** = new ArrayList()

### Additional Inherited Members

---

### Detailed Description

Processes the rolling event file for linestrings from the LADOTD's 511 system and converts it into IMRCP's incident and work zone .csv file format. The file from 511 is a .csv file and has records appended to it that are never removed so the file continually gets larger.

#### Author

Federal Highway Administration

---

### Member Function Documentation

*long imrcp.comp.LADOTD511Line.getEvents (String sFile, long lTime) throws Exception [protected]*

Parses the event file and updates the currently active events in memory.

#### Parameters

<i>sFile</i>	Event file to parse
<i>lFileTime</i>	Timestamp for the file in milliseconds since Epoch

#### Returns

Timestamp the file was last updated in milliseconds since Epoch

#### Exceptions

<i>Exception</i>	
------------------	--

Reimplemented from **imrcp.comp.EventComp** (p.204).

*void imrcp.comp.LADOTD511Line.reset ()*

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.comp.EventComp** (p.204).

*boolean imrcp.comp.LADOTD511Line.start () throws Exception*

Reads the current 511 rolling .csv on disk, if it exists, to determine the last updated time for each event.

#### Returns

true if no Exceptions are thrown

#### Exceptions

Exception
Reimplemented from <b>imrcp.system.BaseBlock</b> (p.99).

---

## Member Data Documentation

*ArrayList<long[]> imrcp.comp.LADOTD511Line.m\_oUpdateTimes = new ArrayList() [protected]*

Stores long[] in the format [511 id, last updated time] to determine if an event has been updated from collection cycle to the next.

*String imrcp.comp.LADOTD511Line.m\_sRollingFile [protected]*

Path where the rolling 511 .csv file is saved on disk

---

*The documentation for this class was generated from the following file:*

- comp/LADOTD511Line.java

---

## imrcp.comp.LADOTD511Point Class Reference

### Public Member Functions

- void **reset** ()
- boolean **start** () throws Exception

### Protected Member Functions

- long **getEvents** (String sFile, long lTime) throws Exception

### Protected Attributes

- String **m\_sRollingFile**
- ArrayList< long[]> **m\_oUpdateTimes** = new ArrayList()

### Additional Inherited Members

---

#### Detailed Description

Processes the rolling event file for points from the LADOTD's 511 system and converts it into IMRCP's incident and work zone .csv file format. The file from 511 is a .csv file and has records appended to it that are never removed so the file continually gets larger.

#### Author

Federal Highway Administration

## Member Function Documentation

`long imrcp.comp.LADOTD511Point.getEvents (String sFile, long lTime) throws Exception [protected]`

Parses the event file and updates the currently active events in memory.

### Parameters

<code>sFile</code>	Event file to parse
<code>lFileTime</code>	Timestamp for the file in milliseconds since Epoch

### Returns

Timestamp the file was last updated in milliseconds since Epoch

### Exceptions

<code>Exception</code>
------------------------

Reimplemented from **imrcp.comp.EventComp** (*p.204*).

`void imrcp.comp.LADOTD511Point.reset ()`

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.comp.EventComp** (*p.204*).

`boolean imrcp.comp.LADOTD511Point.start () throws Exception`

Reads the current 511 rolling .csv on disk, if it exists, to determine the last updated time for each event.

### Returns

true if no Exceptions are thrown

### Exceptions

<code>Exception</code>
------------------------

Reimplemented from **imrcp.system.BaseBlock** (*p.99*).

---

## Member Data Documentation

`ArrayList<long[]> imrcp.comp.LADOTD511Point.m_oUpdateTimes = new ArrayList() [protected]`

Stores long[] in the format [511 id, last updated time] to determine if an event has been updated from collection cycle to the next.

`String imrcp.comp.LADOTD511Point.m_sRollingFile [protected]`

Path where the rolling 511 .csv file is saved on disk

---

*The documentation for this class was generated from the following file:*

- comp/LADOTD511Point.java
-

## [imrcp.store.LaDOTDEvent Class Reference](#)

### [Public Member Functions](#)

- [\*\*LaDOTDEvent \(\)\*\*](#)
- [\*\*LaDOTDEvent \(LaDOTDEvent oEvent\)\*\*](#)
- [\*\*void reset \(\)\*\*](#)
- [\*\*void copyValues \(LaDOTDEvent oEvent\)\*\*](#)
- [\*\*void writeToFile \(Writer oOut, SimpleDateFormat oSdf\) throws Exception\*\*](#)
- [\*\*void setTimesFromBuffers \(SimpleDateFormat oSdf\) throws Exception\*\*](#)
- [\*\*void close \(long lTime, SimpleDateFormat oSdf\)\*\*](#)

### [Public Attributes](#)

- [\*\*StringBuilder m\\_sUpdateTime\*\*](#)
- [\*\*StringBuilder m\\_sStartTime\*\*](#)
- [\*\*StringBuilder m\\_sEndTime\*\*](#)

### [Additional Inherited Members](#)

---

#### [Detailed Description](#)

Represents events received from the LaDOTD ATMS data feed.

#### *Author*

Federal Highway Administration

---

#### [Constructor & Destructor Documentation](#)

##### [\*imrcp.store.LaDOTDEvent.LaDOTDEvent \(\)\*](#)

Default constructor. Initializes the time StringBuilders to the size of the date time strings found in the LaDOTD ATMS then calls **reset ()**

##### [\*imrcp.store.LaDOTDEvent.LaDOTDEvent \(LaDOTDEvent oEvent\)\*](#)

Copy constructor. Initializes the time StringBuilders to the size of the date time strings found in the LaDOTD ATMS then calls **copyValues ()** with the given event.

#### *Parameters*

<i>oEvent</i>	event to copy
---------------	---------------

---

#### [Member Function Documentation](#)

##### [\*void imrcp.store.LaDOTDEvent.close \(long lTime, SimpleDateFormat oSdf\)\*](#)

Sets the end time and update time to the given time.

#### *Parameters*

<i>lTime</i>	time in milliseconds since Epoch that the event ended.
<i>oSdf</i>	date time format object

Reimplemented from **imrcp.store.EventObs** (*p.207*).

`void imrcp.store.LaDOTDEvent.copyValues (LaDOTDEvent oEvent)`

Calls `EventObs#copyValues (imrcp.store.EventObs)` then copies the time buffers from the given event.

*Parameters*

<code>oEvent</code>	event to copy
---------------------	---------------

`void imrcp.store.LaDOTDEvent.reset ()`

Calls `EventObs#reset ()` then clears the time buffers.

Reimplemented from `imrcp.store.EventObs` (p.207).

`void imrcp.store.LaDOTDEvent.setTimesFromBuffers (SimpleDateFormat oSdf) throws Exception`

Sets the start, end, and updated time by parsing the time buffers with the given `SimpleDateFormat`

*Parameters*

<code>oSdf</code>	date parsing object
-------------------	---------------------

*Exceptions*

<code>Exception</code>
------------------------

`void imrcp.store.LaDOTDEvent.writeToFile (Writer oOut, SimpleDateFormat oSdf) throws Exception`

Writes the parameters of the event to the given writer in IMRCP's CSV work zone and event file format

*Parameters*

<code>oOut</code>	Writer that writes the event
<code>oSdf</code>	date formatting object

*Exceptions*

<code>Exception</code>
------------------------

Reimplemented from `imrcp.store.EventObs` (p.207).

---

## Member Data Documentation

`StringBuilder imrcp.store.LaDOTDEvent.m_sEndTime`

Buffer used to store the end time from the LaDOTD ATMS data feed

`StringBuilder imrcp.store.LaDOTDEvent.m_sStartTime`

Buffer used to store the start time from the LaDOTD ATMS data feed

`StringBuilder imrcp.store.LaDOTDEvent.m_sUpdateTime`

Buffer used to store the updated time from the LaDOTD ATMS data feed

---

*The documentation for this class was generated from the following file:*

- `store/LaDOTDEvent.java`

---

## imrcp.store.grib.LambertConformalProj Class Reference

### Public Member Functions

- **LambertConformalProj** (DataInputStream oIn, int nSecLen, int nTemplate) throws IOException

### Public Attributes

- double **m\_dRadius**
- double **m\_dOriginLat**
- double **m\_dOriginLon**
- double **m\_dParallelOne**
- double **m\_dParallelTwo**

### Additional Inherited Members

---

#### Detailed Description

**Projection** implementation for Section 3 template 30 (Lambert Conformal) of GRIB2 files.

#### Author

Federal Highway Administration

---

#### Constructor & Destructor Documentation

*imrcp.store.grib.LambertConformalProj.LambertConformalProj (DataInputStream oIn, int nSecLen, int nTemplate) throws IOException*

Constructs a **LambertConformalProj** from Section 3 of a GRIB2 file

#### Parameters

<i>oIn</i>	DataInputStream of the GRIB2 file. It should be ready to read the 15th byte of Section 3 (meaning the length (4 bytes) of the section, section number (1 byte), source of grid definition (1 byte), number of data points (4 bytes), number of octets for optional list of numbers, (1 byte), Interpretation of list of numbers defining number of points (1 byte) and the grid definition template number (2 bytes) has been read)
<i>nSecLen</i>	length of Section 3 in bytes
<i>nTemplate</i>	<b>Grid</b> definition template number, for this case should be 30

#### Exceptions

<i>IOException</i>
--------------------

## Member Data Documentation

*double imrcp.store.grib.LambertConformalProj.m\_dOriginLat*

LaD - Latitude where Dx (x-direction grid length) and Dy (y-direction grid length) are specified

*double imrcp.store.grib.LambertConformalProj.m\_dOriginLon*

LoV - longitude of meridian parallel to y-axis along which latitude increases as the y-coordinate increases

*double imrcp.store.grib.LambertConformalProj.m\_dParallelOne*

Latin 1 - first latitude from the pole at which the secant cone cuts the sphere

*double imrcp.store.grib.LambertConformalProj.m\_dParallelTwo*

Latin 2 - second latitude from the pole at which the secant cone cuts the sphere

*double imrcp.store.grib.LambertConformalProj.m\_dRadius*

Radius of spherical Earth in km

---

*The documentation for this class was generated from the following file:*

- store/grib/LambertConformalProj.java
- 

## imrcp.web.LaneServlet Class Reference

### Public Member Functions

- void **reset** ()
- int **doLanes** (HttpServletRequest oReq, HttpServletResponse oRes, Session oSession) throws IOException, ServletException

### Additional Inherited Members

---

#### Detailed Description

Servlet used to save manually entered number of lanes using lanemap.html

#### Author

Federal Highway Administration

---

## Member Function Documentation

*int imrcp.web.LaneServlet.doLanes (HttpServletRequest oReq, HttpServletResponse oRes, Session oSession) throws IOException, ServletException*

Writes the requested segment id and number of lanes to the configured file

#### Parameters

<i>oRes</i>	an HttpServletRequest object that contains the request the client has made of the servlet
-------------	---

<i>oReq</i>	an <code>HttpServletResponse</code> object that contains the response the servlet sends to the client
<i>oSession</i>	

*Returns*

`HttpServletResponse#SC_OK`

*Exceptions*

<i>IOException</i>	
<i>ServletException</i>	
<i>n</i>	

`void imrcp.web.LaneServlet.reset ()`

Sets configurable member variables from the `BlockConfig` object, should be overridden by child classes.

Reimplemented from `imrcp.web.SecureBaseBlock` (p.492).

*The documentation for this class was generated from the following file:*

- `web/LaneServlet.java`

## imrcp.web.LatLng Class Reference

### Public Member Functions

- `LatLng ()`
- `LatLng (int nLat, int nLng)`
- `LatLng (double nLat, double nLng)`
- `int getLat ()`
- `void setLat (int nLat)`
- `int getLng ()`
- `void setLng (int nLng)`

### Detailed Description

A point represented by a latitude and a longitude in decimal degrees scaled to 7 decimal places

### Author

Federal Highway Administration

### Constructor & Destructor Documentation

`imrcp.web.LatLng.LatLng ()`

Default constructor. Does nothing.

`imrcp.web.LatLng.LatLng (int nLat, int nLng)`

Constructs a new `LatLng` with the given values

*Parameters*

<i>nLat</i>	latitude in decimal degrees scaled to 7 decimal places
<i>nLng</i>	longitude in decimal degrees scaled to 7 decimal places

*imrcp.web.LatLng.LatLng (double nLat, double nLng)*

Constructs a new **LatLng** with the given values by converting the decimal degrees doubles into integer decimal degrees scaled to 7 decimal places

*Parameters*

<i>nLat</i>	latitude in decimal degrees
<i>nLng</i>	longitude in decimal degrees

---

## Member Function Documentation

*int imrcp.web.LatLng.getLat ()*

Gets the latitude value

*Returns*

latitude of the point

*int imrcp.web.LatLng.getLng ()*

Gets the longitude value

*Returns*

longitude of the point

*void imrcp.web.LatLng.setLat (int nLat)*

Sets the latitude value

*Parameters*

<i>nLat</i>	latitude in decimal degrees scaled to 7 decimal places
-------------	--

*void imrcp.web.LatLng.setLng (int nLng)*

Sets the longitude value

*Parameters*

<i>nLng</i>	longitude in decimal degrees scaled to 7 decimal places
-------------	---

---

*The documentation for this class was generated from the following file:*

- web/LatLng.java
- 

## imrcp.web.LatLngBounds Class Reference

### Public Member Functions

- **LatLngBounds** (int nLat1, int nLng1, int nLat2, int nLng2)
- **LatLngBounds** (double dLat1, double dLng1, double dLat2, double dLng2)

- boolean **intersects** (int nLat1, int nLng1, int nLat2, int nLng2)
  - int **getNorth** ()
  - int **getSouth** ()
  - int **getEast** ()
  - int **getWest** ()
- 

## Detailed Description

A bounding box represented by two **LatLng** points, namely the north west corner and the south east corner.

### Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

*imrcp.web.LatLngBounds.LatLngBounds (int nLat1, int nLng1, int nLat2, int nLng2)*

Constructs a new **LatLngBounds** with the given latitudes and longitudes in integer decimal degrees scaled to 7 decimal places

### Parameters

<i>nLat1</i>	first latitude in decimal degrees scaled to 7 decimal places
<i>nLng1</i>	first longitude in decimal degrees scaled to 7 decimal places
<i>nLat2</i>	second latitude in decimal degrees scaled to 7 decimal places
<i>nLng2</i>	second longitude in decimal degrees scaled to 7 decimal places

*imrcp.web.LatLngBounds.LatLngBounds (double dLat1, double dLng1, double dLat2, double dLng2)*

Constructs a new **LatLngBounds** with the given latitudes and longitudes in double decimal degrees

### Parameters

<i>dLat1</i>	first latitude in decimal degrees
<i>dLng1</i>	first longitude in decimal degrees
<i>dLat2</i>	second latitude in decimal degrees
<i>dLng2</i>	second longitude in decimal degrees

---

## Member Function Documentation

*int imrcp.web.LatLngBounds.getEast ()*

Gets the maximum longitude value

### Returns

Maximum (east) longitude value

*int imrcp.web.LatLngBounds.getNorth ()*

Gets the maximum latitude value

*Returns*

Maximum (north) latitude value

*int imrcp.web.LatLngBounds.getSouth ()*

Gets the minimum latitude value

*Returns*

Minimum (south) latitude value

*int imrcp.web.LatLngBounds.getWest ()*

Get the minimum longitude value

*Returns*

Minimum (west) longitude value

*boolean imrcp.web.LatLngBounds.intersects (int nLat1, int nLng1, int nLat2, int nLng2)*

Determines if the bounding boxes made from the given latitudes and longitudes in integer decimal degrees scaled to 7 decimal places intersects this **LatLngBounds**

*Parameters*

<i>nLat1</i>	first latitude in decimal degrees scaled to 7 decimal places
<i>nLng1</i>	first longitude in decimal degrees scaled to 7 decimal places
<i>nLat2</i>	second latitude in decimal degrees scaled to 7 decimal places
<i>nLng2</i>	second longitude in decimal degrees scaled to 7 decimal places

*Returns*

true if the bounding boxes intersect, otherwise false

---

*The documentation for this class was generated from the following file:*

- web/LatLngBounds.java
- 

## [imrcp.store.grib.LatLngProj Class Reference](#)

### [Public Member Functions](#)

- **LatLngProj** (DataInputStream oIn, int nSecLen, int nTemplate) throws IOException

### [Public Attributes](#)

- double **m\_dStartLat**
- double **m\_dStartLon**
- double **m\_dLatInc**
- double **m\_dLonInc**

### [Additional Inherited Members](#)

---

## Detailed Description

**Projection** implementation for Section 3 template 0 (Latitude/Longitude) of GRIB2 files.

### Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

*imrcp.store.grib.LatLonProj.LatLonProj (DataInputStream oIn, int nSecLen, int nTemplate) throws IOException*

Constructs a **LatLonProj** from Section 3 of a GRIB2 file

### Parameters

<i>oIn</i>	DataInputStream of the GRIB2 file. It should be ready to read the 15th byte of Section 3 (meaning the length (4 bytes) of the section, section number (1 byte), source of grid definition (1 byte), number of data points (4 bytes), number of octets for optional list of numbers, (1 byte), Interpretation of list of numbers defining number of points (1 byte) and the grid definition template number (2 bytes) has been read)
<i>nSecLen</i>	length of Section 3 in bytes
<i>nTemplate</i>	<b>Grid</b> definition template number, for this case should be 0

---

## Member Data Documentation

*double imrcp.store.grib.LatLonProj.m\_dLatInc*

Latitude increment

*double imrcp.store.grib.LatLonProj.m\_dLonInc*

Longitude increment

*double imrcp.store.grib.LatLonProj.m\_dStartLat*

Latitude of first grid point

*double imrcp.store.grib.LatLonProj.m\_dStartLon*

Longitude of first grid point

---

*The documentation for this class was generated from the following file:*

- store/grib/LatLonProj.java
- 

---

## imrcp.web.layers.LayerServlet Class Reference

### Public Member Functions

- void **reset ()**

- int **doPlatformObs** (HttpServletRequest oReq, HttpServletResponse oRes, **Session** oSess) throws IOException, ServletException
- int **doChartObs** (HttpServletRequest oReq, HttpServletResponse oRes, **Session** oSession) throws ServletException, IOException

### Protected Member Functions

- void **processObsRequest** (HttpServletResponse oResp, **ObsRequest** oObsRequest) throws Exception
- void **processChartRequest** (HttpServletResponse oResp, **ObsChartRequest** oObsRequest) throws Exception
- void **serializeObsRecord** (JsonGenerator oOutputGenerator, DecimalFormat oNumberFormatter, **Obs** oObs) throws IOException
- abstract void **buildObsResponseContent** (JsonGenerator oOutputGenerator, **ObsRequest** oObsRequest) throws Exception
- void **buildObsChartResponseContent** (JsonGenerator oOutputGenerator, **ObsChartRequest** oObsRequest) throws Exception
- String **formatDetailString** (String sDetail, String sInsert, int nStep)

### Static Protected Attributes

- static Comparator< **Obs** > **g\_oObsDetailComp**

### Additional Inherited Members

---

#### Detailed Description

Base class used for handling different requests based on the IMRCP Map UI's layers which include points (sensors, events, alerts), lines (roads), and polygons (areal weather)

#### Author

Federal Highway Administration

---

#### Member Function Documentation

`void imrcp.web.layers.LayerServlet.buildObsChartResponseContent (JsonGenerator oOutputGenerator, ObsChartRequest oObsRequest) throws Exception [protected]`

Child classes implement this method to create the response when a chart is requested from the map.

#### Parameters

<code>oOutputGenerator</code>	JSON stream
<code>oObsRequest</code>	object that contains the query parameters for the request

#### Exceptions

<code>Exception</code>	
------------------------	--

Reimplemented in **imrcp.web.layers.AreaLayerServlet** (p.86), **imrcp.web.layers.PointsLayerServlet** (p.453), and **imrcp.web.layers.RoadLayerServlet** (p.481).

```
abstract void imrcp.web.layers.LayerServlet.buildObsResponseContent (JsonGenerator oOutputGenerator, ObsRequest oObsRequest) throws Exception [abstract], [protected]
```

Child classes implement this method to create the response when an object in its layer is clicked on the map.

#### Parameters

<i>oOutputGenerator</i>	JSON stream
<i>oObsRequest</i>	object that contains the query parameters for the request

#### Exceptions

<i>Exception</i>	
Reimplemented in <b>imrcp.web.layers.AreaLayerServlet</b> (p.86), <b>imrcp.web.layers.PointsLayerServlet</b> (p.453), and <b>imrcp.web.layers.RoadLayerServlet</b> (p.481).	

```
int imrcp.web.layers.LayerServlet.doChartObs (HttpServletRequest oReq, HttpServletResponse oRes, Session oSession) throws ServletException, IOException
```

Handles finding observation that match the HTTP request to generate a chart.

#### Parameters

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client
<i>oSession</i>	object that contains information about the user that made the request

#### Returns

HTTP status code to be included in the response.

#### Exceptions

<i>ServletException</i>	
<i>IOException</i>	

```
int imrcp.web.layers.LayerServlet.doPlatformObs (HttpServletRequest oReq, HttpServletResponse oRes, Session oSess) throws IOException, ServletException
```

Handles finding observation that match the HTTP request.

#### Parameters

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client
<i>oSess</i>	object that contains information about the user that made the request

#### Returns

HTTP status code to be included in the response.

#### Exceptions

<i>IOException</i>	
<i>ServletException</i>	

*String imrcp.web.layers.LayerServlet.formatDetailString (String sDetail, String sInsert, int nStep) [protected]*

Used to format the detail string of a response to produce a better presentation of it in the observation table on the IMRCP Map UI

#### Parameters

<i>sDetail</i>	String to format
<i>sInsert</i>	String to insert approximately every nStep characters, usually "@iliteral   @endiliteral "
<i>nStep</i>	Number of characters to skip before inserting sInsert into sDetail

#### Returns

sDetail with sInsert added approxiamately every nStep characters

*void imrcp.web.layers.LayerServlet.processChartRequest (HttpServletResponse oResp, ObsChartRequest oObsRequest) throws Exception [protected]*

Creates a **JsonGenerator** to pass into  
**buildObsChartResponseContent (org.codehaus.jackson.JsonGenerator, imrcp.web.ObsChartRequest)**

#### Parameters

<i>oResp</i>	object that contains the response the servlet sends to the client
<i>oObsRequest</i>	object that contains the query parameters for the request

#### Exceptions

<i>Exception</i>	
------------------	--

*void imrcp.web.layers.LayerServlet.processObsRequest (HttpServletResponse oResp, ObsRequest oObsRequest) throws Exception [protected]*

Creates a **JsonGenerator** to pass into  
**buildObsResponseContent (org.codehaus.jackson.JsonGenerator, imrcp.web.ObsRequest)**

#### Parameters

<i>oResp</i>	object that contains the response the servlet sends to the client
<i>oObsRequest</i>	object that contains the query parameters for the request

#### Exceptions

<i>Exception</i>	
------------------	--

*void imrcp.web.layers.LayerServlet.reset ()*

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.web.SecureBaseBlock** (p.492).

Reimplemented in [imrcp.web.layers.AreaLayerServlet](#) (p.86), and [imrcp.web.layers.RoadLayerServlet](#) (p.481).

```
void imrcp.web.layers.LayerServlet.serializeObsRecord(JsonGenerator oOutputGenerator, DecimalFormat oNumberFormatter, Obs oObs) throws IOException [protected]
```

Formats the Obs as a JSON object in the given JsonGenerator.

#### Parameters

<i>oOutputGenerator</i>	JSON stream
<i>oNumberFormatter</i>	Formatting object for converting numbers to Strings
<i>oObs</i>	the Obs to serialize into JSON object

#### Exceptions

<i>IOException</i>
--------------------

---

## Member Data Documentation

*Comparator<Obs>* [imrcp.web.layers.LayerServlet.g\\_oObsDetailComp \[static\], \[protected\]](#)

```
Initial value:= (Obs o1, Obs o2) ->
{
    int nReturn =
ObsType.getDescription(o1.m_nObsTypeId).compareTo(ObsType.getDescription(o2.m_nObs
TypeId));
    if (nReturn == 0)
    {
        nReturn = o1.m_nContribId - o2.m_nContribId;
        if (nReturn == 0)
        {
            nReturn = Long.compare(o1.m_lObsTime1, o2.m_lObsTime1);
            if (nReturn == 0)
                nReturn = Long.compare(o1.m_lObsTime2, o2.m_lObsTime2);
        }
    }
    return nReturn;
}
```

Compares Obs by observation type id, then contributor id, then start time, then end time.

---

*The documentation for this class was generated from the following file:*

- [web/layers/LayerServlet.java](#)

---

## imrcp.system.Locks Class Reference

### Public Member Functions

- `void reset ()`
- `boolean start ()`
- `ReentrantReadWriteLock getLock (String sName)`
- `void execute ()`

## Additional Inherited Members

---

### Detailed Description

System component used to map a String (usually a file name) and a RenentrantReadWriteLock. This allows for synchronized processing on files that can be read and written by different threads in the system.

### Author

Federal Highway Administration

---

### Member Function Documentation

#### `void imrcp.system.Locks.execute ()`

Checks the map of locks for stale locks no longer being used by any system components  
Reimplemented from **imrcp.system.BaseBlock** (p.95).

#### `RenentrantReadWriteLock imrcp.system.Locks.getLock (String sName)`

Get the lock associated with the given string. If there is not an associated lock it is created before being returned.

##### Parameters

<code>sName</code>	Name of the desired lock, usually a file path
--------------------	---

##### Returns

RenentrantReadWriteLock associated with the name

#### `void imrcp.system.Locks.reset ()`

Sets configurable member variables from the **BlockConfig** object, should be overridden by child classes.

Reimplemented from **imrcp.system.BaseBlock** (p.98).

#### `boolean imrcp.system.Locks.start ()`

sets a schedule to execute on a fixed interval.

##### Returns

true if no exceptions are thrown

Reimplemented from **imrcp.system.BaseBlock** (p.99).

---

*The documentation for this class was generated from the following file:*

- system/Locks.java
- 

## imrcp.system.MathUtil Class Reference

### Static Public Member Functions

- static double **standardDeviation** (double[] dVals, double[] dMeanOutput)

---

## Detailed Description

This class contains static utility methods for other classes to use

---

## Member Function Documentation

*static double imrcp.system.MathUtil.standardDeviation (double[] dVals, double[] dMeanOutput) [static]*

Gets the standard deviation of the values in the given growable array. The mean is calculated and stored in the first position of dMeanOutput

### Parameters

<i>dVals</i>	growable array containing the values used to calculate the standard deviation
<i>dMeanOutput</i>	array to store the mean of the values

### Returns

The standard deviation of the values in the growable array

---

*The documentation for this class was generated from the following file:*

- system/MathUtil.java
- 

---

## imrcp.geosrv.Mercator Class Reference

### Public Member Functions

- **Mercator ()**
- **Mercator (int nTileSize)**
- **void metersToLonLat (double dX, double dY, double[] dLatLon)**
- **void pixelsToMeters (double dXp, double dYp, int nZoom, double[] dMeters)**
- **void metersToPixels (double dXm, double dYm, int nZoom, double[] dPixels)**
- **void pixelsToTile (double dXp, double dYp, int[] nTiles)**
- **void tileBounds (double dXt, double dYt, int nZoom, double[] dBounds)**
- **void lonLatBounds (double dXt, double dYt, int nZoom, double[] dBounds)**
- **void lonLatMdpt (double dXt, double dYt, int nZoom, double[] dMdpt)**
- **void lonLatToTile (double dLon, double dLat, int nZoom, int[] nTiles)**
- **void metersToTile (double dXm, double dYm, int nZoom, int[] nTiles)**
- **double resolution (int nZoom)**

### Static Public Member Functions

- **static int getExtent (int nZoom)**
- **static double lonToMeters (double dLon)**
- **static double latToMeters (double dLat)**
- **static double xToLon (double dX)**
- **static double yToLat (double dY)**
- **static void lonLatToMeters (double dLon, double dLat, double[] dMeters)**

## Static Public Attributes

- static final double **PI\_OVER\_TWO** = Math.PI / 2.0
  - static final double **PI\_OVER\_FOUR** = Math.PI / 4.0
- 

## Detailed Description

Contains methods for converting WGS 84 longitude/latitude (epsg:4326), Spherical **Mercator** (epsg:3857), and map tile coordinates. Most methods were adapted from algorithms found at <https://www.maptiler.com/google-maps-coordinates-tile-bounds-projection>

### Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

### *imrcp.geosrv.Mercator.Mercator ()*

Default constructor. Wrapper for **Mercator#Mercator(int)** with tile size 256.

### *imrcp.geosrv.Mercator.Mercator (int nTileSize)*

Constructs a **Mercator** object with the given tile size. Calculates the resolution for each zoom level.

#### Parameters

<i>nTileSize</i>
------------------

## Member Function Documentation

### *static int imrcp.geosrv.Mercator.getExtent (int nZoom) [static]*

Gets the extent used in vector tiles for the given zoom level

#### Parameters

<i>nZoom</i>
--------------

map zoom level
----------------

#### Returns

extent used in vector tiles for the given zoom level

### *static double imrcp.geosrv.Mercator.latToMeters (double dLat) [static]*

Converts a WGS 84 latitude to Spherical **Mercator** meters

#### Parameters

<i>dLat</i>
-------------

latitude in decimal degrees
-----------------------------

#### Returns

Corresponding mercator meter y coordinate

```
void imrcp.geosrv.Mercator.IonLatBounds (double dXt, double dYt, int nZoom,
double[] dBounds)
```

Fills the given double array with the corresponding bounds in longitude and latitude in decimal degrees of the given map tile coordinates at the given zoom level

#### Parameters

<i>dXt</i>	map tile x coordinate
<i>dYt</i>	map tile y coordinate
<i>nZoom</i>	map zoom level
<i>dBounds</i>	array to be filled with [min longitude, min latitude, max longitude, max latitude]

```
void imrcp.geosrv.Mercator.IonLatMdpt (double dXt, double dYt, int nZoom, double[]
dMdpt)
```

Fills the given double array with the corresponding longitude and latitude in decimal degrees midpoint of the given tile coordinates.

#### Parameters

<i>dXt</i>	tile x coordinate
<i>dYt</i>	tile y coordinate
<i>nZoom</i>	map zoom level
<i>dMdpt</i>	array to be filled with [midpoint longitude, midpoint latitude]

```
static void imrcp.geosrv.Mercator.IonLatToMeters (double dLon, double dLat,
double[] dMeters) [static]
```

Fills the given double array with the corresponding x and y Spherical **Mercator** meter coordinate of the given longitude and latitude in decimal degrees

#### Parameters

<i>dLon</i>	longitude in decimal degrees
<i>dLat</i>	latitude in decimal degrees
<i>dMeters</i>	array to be filled with [mercator meter x, mercator meter y]

```
void imrcp.geosrv.Mercator.IonLatToTile (double dLon, double dLat, int nZoom, int[]
nTiles)
```

Fills the given array with the corresponding tile coordinate of the given longitude and latitude in decimal degrees for the given map zoom level

#### Parameters

<i>dLon</i>	longitude in decimal degrees
<i>dLat</i>	latitude in decimal degrees
<i>nZoom</i>	map zoom level
<i>nTiles</i>	array to be filled with [tile x, tile y]

```
static double imrcp.geosrv.Mercator.IonToMeters (double dLon) [static]
```

Converts a WGS 84 longitude to Spherical **Mercator** meters

#### Parameters

<i>dLon</i>	longitude in decimal degrees
-------------	------------------------------

#### Returns

Corresponding mercator meter x coordinate

`void imrcp.geosrv.Mercator.metersToLonLat (double dX, double dY, double[] dLatLon)`

Fills the given double array with the corresponding longitude and latitude in decimal degrees of the given x and y Spherical **Mercator** meter coordinates.

#### Parameters

<i>dX</i>	mercator meters x
<i>dY</i>	mercator meters y
<i>dLatLon</i>	array to be filled with [longitude, latitude]

`void imrcp.geosrv.Mercator.metersToPixels (double dXm, double dYm, int nZoom, double[] dPixels)`

Fills the given double array by converting Spherical **Mercator** meters to pixel coordinates at the given zoom level.

#### Parameters

<i>dXm</i>	mercator meters x
<i>dYm</i>	mercator meters y
<i>nZoom</i>	map zoom level
<i>dPixels</i>	array to be filled with [pixel x, pixel y]

`void imrcp.geosrv.Mercator.metersToTile (double dXm, double dYm, int nZoom, int[] nTiles)`

Fills the given array with the corresponding tile coordinate of the given Spherical **Mercator** meter coordinates for the given map zoom level.

#### Parameters

<i>dXm</i>	mercator meters x
<i>dYm</i>	mercator meters y
<i>nZoom</i>	map zoom level
<i>nTiles</i>	array to be filled with [tile x, tile y]

`void imrcp.geosrv.Mercator.pixelsToMeters (double dXp, double dYp, int nZoom, double[] dMeters)`

Fills the given double array by converting pixel coordinates to Spherical **Mercator** meters at the given zoom level.

#### Parameters

<i>dXp</i>	pixel x coordinate
<i>dYp</i>	pixel y coordinate
<i>nZoom</i>	map zoom level
<i>dMeters</i>	array to be filled with [mercator meters x, mercator meters y]

`void imrcp.geosrv.Mercator.pixelsToTile (double dXp, double dYp, int[] nTiles)`

Fills the given int array with the corresponding tile x and y coordinates of the given pixel coordinates

*Parameters*

<i>dXp</i>	pixel x coordinate
<i>dYp</i>	pixel y coordinate
<i>nTiles</i>	array to be filled with [tile x, tile y]

*double imrcp.geosrv.Mercator.resolution (int nZoom)*

Gets the resolution (meters/pixel) at the given zoom level

*Parameters*

<i>nZoom</i>	map zoom level
--------------	----------------

*Returns*

Resolution at the zoom level

*void imrcp.geosrv.Mercator.tileBounds (double dXt, double dYt, int nZoom, double[] dBounds)*

Fills the given double array with the corresponding bounds in Spherical **Mercator** meters of the given map tile coordinates at the given zoom level

*Parameters*

<i>dXt</i>	map tile x coordinate
<i>dYt</i>	map tile y coordinate
<i>nZoom</i>	map zoom level
<i>dBounds</i>	array to be filled with [min x mercator meters, min y mercator meters, max x mercator meters, max y mercator meters]

*static double imrcp.geosrv.Mercator.xToLon (double dX) [static]*

Converts a Spherical **Mercator** meters x coordinate to a WGS 84 longitude

*Parameters*

<i>dX</i>	mercator meter x coordinate
-----------	-----------------------------

*Returns*

Corresponding longitude in decimal degrees

*static double imrcp.geosrv.Mercator.yToLat (double dY) [static]*

Converts a Spherical **Mercator** meters y coordinate to a WGS 84 latitude

*Parameters*

<i>dY</i>	mercator meter y coordinate
-----------	-----------------------------

*Returns*

Corresponding latitude in decimal degrees

---

## Member Data Documentation

*final double imrcp.geosrv.Mercator.PI\_OVER\_FOUR = Math.PI / 4.0 [static]*

Pi divided by 4

```
final double imrcp.geosrv.Mercator.PI_OVER_TWO = Math.PI / 2.0 [static]
```

Pi divided by 2

---

*The documentation for this class was generated from the following file:*

- geosrv/Mercator.java
- 

## imrcp.forecast.mdss.Metro Class Reference

### Classes

- class **MetroTile**

### Public Member Functions

- void **reset** ()
- boolean **start** () throws Exception
- boolean **stop** ()
- void **process** (String[] sNotification)
- void **queue** (String sStart, String sEnd, StringBuilder sBuffer)
- void **queueStatus** (StringBuilder sBuffer)
- void **execute** ()
- void **runMetro** (long lRunTime)

### Additional Inherited Members

---

### Detailed Description

Manages running the METRo model in real-time or can be configured to process specific dates on demand for all the roadway segments in the system. A multi threaded approach is used along with categorizing locations that share the same characteristics and weather forecasts to optimize performance

### Author

Federal Highway Administration

---

### Member Function Documentation

#### `void imrcp.forecast.mdss.Metro.execute ()`

If **Metro#m\_bRealTime** is true, adds the current time to the front of the queue. Then processes up to **Metro#m\_nRunsPerPeriod** times in the queue.

Reimplemented from **imrcp.system.BaseBlock** (p.95).

#### `void imrcp.forecast.mdss.Metro.process (String[] sNotification)`

Called when a message from **imrcp.system.Directory** is received. If the message is "new ways", the tiles that will be processed each period of execution is updated

#### Parameters

<code>sNotification</code>	
----------------------------	--

Reimplemented from **imrcp.system.BaseBlock** (p.97).

`void imrcp.forecast.mdss.Metro.queue (String sStart, String sEnd, StringBuilder sBuffer)`

If this instance is configured to reprocess files, it parses the given start and end times and queues times within that range to be reprocessed. The given `StringBuilder` gets filled with basic html that contains the files that were queued.

*Parameters*

<code>sStart</code>	start timestamp in the format yyyy-MM-ddTHH:mm
<code>sEnd</code>	end timestamp in the format yyyy-MM-ddTHH:mm
<code>sBuffer</code>	Buffer that gets fills with html containing the queued files

`void imrcp.forecast.mdss.Metro.queueStatus (StringBuilder sBuffer)`

Adds the current times in the queue to the `StringBuilder` in basic html

*Parameters*

<code>sBuffer</code>	<code>StringBuilder</code> to fill with current files queued
----------------------	--

`void imrcp.forecast.mdss.Metro.reset ()`

Sets configurable member variables from the `BlockConfig` object, should be overridden by child classes.

Reimplemented from `imrcp.system.BaseBlock` (p.98).

`void imrcp.forecast.mdss.Metro.runMetro (long lRunTime)`

Runs the METRo model for the each tile in `Metro#m_oTiles` by aggregating all of the input files (except previous METRo runs, which is done later by each thread) needed and then creating a `java.util.concurrent.Callable` object for each `MetroTile` and calling `java.util.concurrent.ExecutorService#invokeAll(java.util.Collection)` on `Metro#m_oThreadPool`.

*Parameters*

<code>lRunTime</code>	Time in milliseconds since Epoch of the METRo run. The first forecast generated by the run is this time.
-----------------------	--

`boolean imrcp.forecast.mdss.Metro.start () throws Exception`

Queues any times found in the queue file. Then updates the tiles that will be processed each period of execution. Then sets a schedule to execute this instance and `Metro#m_oTileUpdate` on a fixed interval.

*Returns*

true if no exceptions are thrown

*Exceptions*

<code>Exception</code>
------------------------

Reimplemented from `imrcp.system.BaseBlock` (p.99).

`boolean imrcp.forecast.mdss.Metro.stop ()`

Wrapper for `java.util.concurrent.ExecutorService#shutdown()` and cancels the schedule for `Metro#m_oTileUpdate`

### Returns

true

Reimplemented from **imrcp.system.BaseBlock** (*p.100*).

---

*The documentation for this class was generated from the following file:*

- forecast/mdss/Metro.java
- 

## imrcp.forecast.mdss.MetroFileset Class Reference

### Public Member Functions

- int **compareTo** (**MetroFileset** o)
- **MetroFileset** (int nX, int nY)
- **MetroFileset** (int nX, int nY, int nObsHours, int nFcstHours)
- **MetroFileset** (int nX, int nY, **MetroFileset** oFileset)
- void **fillFiles** (long lRuntime, int nObsHrs, int nFcstHrs)
- void **fillMetroFiles** (long lRuntime, int nObsHrs, int nLon, int nLat)
- String **checkFiles** ()

### Static Public Member Functions

- static void **setStores** (String sMetro, String sTpvt, String sTssrf, String sRtma, String sRap, String sMrms, String sNdfdTemp, String sNdfdTd, String sNdfdWspd, String sNdfdSky)

### Public Attributes

- int **m\_nX**
- int **m\_nY**
- **MetroWrapper[] m\_oObsMetro**
- **GriddedFileWrapper[] m\_oObsTpvt**
- **GriddedFileWrapper[] m\_oObsTssrf**
- **GriddedFileWrapper[] m\_oObsRtma**
- **GriddedFileWrapper[] m\_oObsRap**
- **GriddedFileWrapper[] m\_oFcstNdfdTemp**
- **GriddedFileWrapper[] m\_oFcstNdfdTd**
- **GriddedFileWrapper[] m\_oFcstNdfdWspd**
- **GriddedFileWrapper[] m\_oFcstNdfdSky**
- **GriddedFileWrapper[] m\_oFcstRap**
- **ArrayList< GriddedFileWrapper > m\_oMrmsPrecip**

### Detailed Description

Encapsulate all of the data files needed for input for a single **imrcp.forecast.mdss.Metro.MetroTile**

### Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

*imrcp.forecast.mdss.MetroFileset.MetroFileset (int nX, int nY)*

Constructs a **MetroFileset** with the given x and y index

### Parameters

<i>nX</i>	x index of map tile
<i>nY</i>	y index of map tile

*imrcp.forecast.mdss.MetroFileset.MetroFileset (int nX, int nY, int nObsHours, int nFcstHours)*

Constructs a **MetroFileset** with the given x and y index and allocates the file arrays according to the number of observation and forecast hours

### Parameters

<i>nX</i>	x index of map tile
<i>nY</i>	y index of map tile
<i>nObsHours</i>	number of observation hours used as input for the METRo run
<i>nFcstHours</i>	number of forecast hours used as input for the METRo run

*imrcp.forecast.mdss.MetroFileset.MetroFileset (int nX, int nY, MetroFileset oFileset)*

Constructs a **MetroFileset** with the given x and y index and uses the file arrays of the given **MetroFileset**, except for the **Metro** files since those are specific for a single map tile

### Parameters

<i>nX</i>	x index of map tile
<i>nY</i>	y index of map tile
<i>oFileset</i>	<b>MetroFileset</b> that already has the file arrays (except <b>Metro</b> files) filled

---

## Member Function Documentation

*String imrcp.forecast.mdss.MetroFileset.checkFiles ()*

Checks if there are any missing files that would cause the METRo run to fail and returns a message detailing the index and store of the first missing file found. If there are no missing files null is returned.

### Returns

null if there are no missing files. Otherwise a String message telling which index and store of the first missing file.

*int imrcp.forecast.mdss.MetroFileset.compareTo (MetroFileset o)*

Compares MetroFilesets by x index then y index.

### Parameters

<i>o</i>	the object to be compared
----------	---------------------------

### Returns

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

### See also

`java.lang.Comparable::compareTo(java.lang.Object)`

**`void imrcp.forecast.mdss.MetroFileset.fillFiles (long lRuntime, int nObsHrs, int nFcstHrs)`**

Fills in the file arrays from the configured data stores based off the **Metro** run time, number of observation hours, and number forecast hours

### Parameters

<code>lRuntime</code>	<b>Metro</b> run time in milliseconds since Epoch
<code>nObsHrs</code>	number of observation hours used as input for the METRo run
<code>nFcstHrs</code>	number of forecast hours used as input for the METRo run

**`void imrcp.forecast.mdss.MetroFileset.fillMetroFiles (long lRuntime, int nObsHrs, int nLon, int nLat)`**

Files the **Metro** file array from the configured data store based off the **Metro** run time, number of observation hours, and location

### Parameters

<code>lRuntime</code>	<b>Metro</b> run time in milliseconds since Epoch
<code>nObsHrs</code>	number of observation hours used as input for the METRo run
<code>nLon</code>	Longitude of location in decimal degrees scaled to 7 decimal places
<code>nLat</code>	Latitude of location in decimal degrees scaled to 7 decimal places

**`static void imrcp.forecast.mdss.MetroFileset.setStores (String sMetro, String sTpvt, String sTssrf, String sRtma, String sRap, String sMrms, String sNdfdTemp, String sNdfdTd, String sNdfdWspd, String sNdfdSky) [static]`**

Sets the static Strings that contain the names of the data stores to query

### Parameters

<code>sMetro</code>	instance name of <b>Metro</b> Store
<code>sTpvt</code>	instance name of kriged pavement temperature store
<code>sTssrf</code>	instance name of kriged subsurface temperature store
<code>sRtma</code>	instance name of RTMA store
<code>sRap</code>	instance name of RAP store
<code>sMrms</code>	instance name of MRMS precip store
<code>sNdfdTemp</code>	instance name of NDFD Temperature store
<code>sNdfdTd</code>	instance name of NDFD Dew Point store
<code>sNdfdWspd</code>	instance name of NDFD Wind Speed store
<code>sNdfdSky</code>	instance name of NDFD cloud cover store

## Member Data Documentation

*int imrcp.forecast.mdss.MetroFileset.m\_nX*

x index of the **imrcp.forecast.mdss.Metro.MetroTile** using this **MetroFileset**

*int imrcp.forecast.mdss.MetroFileset.m\_nY*

y index of the **imrcp.forecast.mdss.Metro.MetroTile** using this **MetroFileset**

*GriddedFileWrapper [] imrcp.forecast.mdss.MetroFileset.m\_oFcstNdfdSky*

Contains the NDFD Cloud Cover files used as input for the current METRo run

*GriddedFileWrapper [] imrcp.forecast.mdss.MetroFileset.m\_oFcstNdfdTd*

Contains the NDFD Dew Point files used as input for the current METRo run

*GriddedFileWrapper [] imrcp.forecast.mdss.MetroFileset.m\_oFcstNdfdTemp*

Contains the NDFD Temperature files used as input for the current METRo run

*GriddedFileWrapper [] imrcp.forecast.mdss.MetroFileset.m\_oFcstNdfdWspd*

Contains the NDFD Wind Speed files used as input for the current METRo run

*GriddedFileWrapper [] imrcp.forecast.mdss.MetroFileset.m\_oFcstRap*

Contains the RAP files used as input for the current METRo run's forecasted values

*ArrayList<GriddedFileWrapper> imrcp.forecast.mdss.MetroFileset.m\_oMrmsPrecip*

Contains the MRMS Precipitation files used as input for the current METRo run

*MetroWrapper [] imrcp.forecast.mdss.MetroFileset.m\_oObsMetro*

Contains the METRo files used as input for the current METRo run

*GriddedFileWrapper [] imrcp.forecast.mdss.MetroFileset.m\_oObsRap*

Contains the RAP files used as input for the current METRo run's observed values

*GriddedFileWrapper [] imrcp.forecast.mdss.MetroFileset.m\_oObsRtma*

Contains the RTMA files used as input for the current METRo run

*GriddedFileWrapper [] imrcp.forecast.mdss.MetroFileset.m\_oObsTpvt*

Contains the kriged pavement temperature files used as input for the current METRo run

*GriddedFileWrapper [] imrcp.forecast.mdss.MetroFileset.m\_oObsTssrf*

Contains the kriged subsurface temperature files used as input for the current METRo run

---

*The documentation for this class was generated from the following file:*

- forecast/mdss/MetroFileset.java
-

## [imrcp.forecast.mdss.MetroProcess Class Reference](#)

### [Public Member Functions](#)

- **MetroProcess** (int nHours)
- void **process** (**Scenario** oScenario) throws Exception
- void **fillPrecip** (**ObsView** oOv, long lStart, long lEnd, long lRef, double[] dRates, int[] nTypes, double[] dTemps, **OsmWay** oWay)
- void **resetArrays** ()

### [Public Attributes](#)

- double[] **m\_dAirTemp**
- double[] **m\_dDewPoint**
- double[] **m\_dWindSpeed**
- double[] **m\_dKrigedTpvt**
- double[] **m\_dKrigedSubSurf**
- double[] **m\_dPavementTemp**
- double[] **m\_dSubSurfTemp**
- int[] **m\_nPavementState**
- double[] **m\_dCloudCover**
- double[] **m\_dPressure**
- int[] **m\_nPrecipType**
- double[] **m\_dPrecipRate**
- double[] **m\_dRainRes**
- double[] **m\_dSnowRes**
- boolean[] **m\_bTreated**
- boolean[] **m\_bPlowed**
- long[] **m\_lTimes**
- int **m\_nFcstPerRun** = 4
- int **m\_nObsPerRun** = 2
- ArrayList< int[]> **m\_oStPvts** = new ArrayList()
- ArrayList< double[]> **m\_oTpvts** = new ArrayList()

---

### [Detailed Description](#)

Manages running the METRo model for Scenarios. For each segment in a Scenario METRo is ran for each hour of a 24 forecast, using the previous run's outputs as inputs for the next.

#### *See also*

[imrcp.web.Scenarios](#)

#### *Author*

Federal Highway Administration

---

### [Constructor & Destructor Documentation](#)

#### [\*imrcp.forecast.mdss.MetroProcess.MetroProcess \(int nHours\)\*](#)

Allocates the memory for the observation/forecast array. The size of each array is the number hours of the scenario + **MetroProcess#m\_nObsPerRun**

- **MetroProcess#m\_nFcstPerRun**
- **Parameters**

<i>nHours</i>	how long the scenario is in hours
---------------	-----------------------------------

## Member Function Documentation

`void imrcp.forecast.mdss.MetroProcess.fillPrecip (ObsView oOv, long lStart, long lEnd, long lRef, double[] dRates, int[] nTypes, double[] dTemps, OsmWay oWay)`

Fills the precip rate and type array by querying ObsView with the given parameters and averaging the rate and if no type observations are found inferring the type using the temperature array

### Parameters

<i>oOv</i>	ObsView instance
<i>lStart</i>	start time of query in milliseconds since Epoch
<i>lEnd</i>	end time of query in milliseconds since Epoch
<i>lRef</i>	reference time of query in millisecond since Epoch
<i>dRates</i>	precip rate array to be filled
<i>nTypes</i>	precip type array to be filled
<i>dTemps</i>	array containing the air temperature for each hour. Used to infer the precip type when it is unknown
<i>oWay</i>	segment METRo is currently being ran on

`void imrcp.forecast.mdss.MetroProcess.process (Scenario oScenario) throws Exception`

Runs the METRo model for each segment in the given Scenario for each hour of the Scenario.

### Parameters

<i>oScenario</i>	Scenario being processed
------------------	--------------------------

### Exceptions

<i>Exception</i>
------------------

`void imrcp.forecast.mdss.MetroProcess.resetArrays ()`

Resets the value of double[]s to Double.NaN and int[]s to Integer.MIN\_VALUE

---

## Member Data Documentation

`boolean [] imrcp.forecast.mdss.MetroProcess.m_bPlowed`

Indicates which hours of the scenario the segments are plowed

`boolean [] imrcp.forecast.mdss.MetroProcess.m_bTreated`

Indicates which hours of the scenario the segments are treated

`double [] imrcp.forecast.mdss.MetroProcess.m_dAirTemp`

Contains air temperature observations and forecasts

`double [] imrcp.forecast.mdss.MetroProcess.m_dCloudCover`

Contains cloud cover observations and forecasts

`double [] imrcp.forecast.mdss.MetroProcess.m_dDewPoint`

Contains dew point observations and forecasts

*double [] imrcp.forecast.mdss.MetroProcess.m\_dKrigedSubSurf*

Contains kriged subsurface temperature observations

*double [] imrcp.forecast.mdss.MetroProcess.m\_dKrigedTpvt*

Contains kriged pavement temperature observations

*double [] imrcp.forecast.mdss.MetroProcess.m\_dPavementTemp*

Contains pavement temperature observations and forecasts

*double [] imrcp.forecast.mdss.MetroProcess.m\_dPrecipRate*

Contains precipitation rate observations and forecasts

*double [] imrcp.forecast.mdss.MetroProcess.m\_dPressure*

Contains surface pressure observations and forecasts

*double [] imrcp.forecast.mdss.MetroProcess.m\_dRainRes*

Contains rain reservoir observations

*double [] imrcp.forecast.mdss.MetroProcess.m\_dSnowRes*

Contains snow reservoir observations

*double [] imrcp.forecast.mdss.MetroProcess.m\_dSubSurfTemp*

Contains subsurface temperature observations and forecasts

*double [] imrcp.forecast.mdss.MetroProcess.m\_dWindSpeed*

Contains wind speed observations and forecasts

*long [] imrcp.forecast.mdss.MetroProcess.m\_lTimes*

Contains the timestamps of each hour of the scenario in milliseconds since Epoch

*int imrcp.forecast.mdss.MetroProcess.m\_nFcstPerRun = 4*

Number of forecast hours used as input for the METRo runs

*int imrcp.forecast.mdss.MetroProcess.m\_nObsPerRun = 2*

Number of observation hours used as input for the METRo runs

*int [] imrcp.forecast.mdss.MetroProcess.m\_nPavementState*

Contains pavement state observations and forecasts

*int [] imrcp.forecast.mdss.MetroProcess.m\_nPrecipType*

Contains precipitation type observations and forecasts

*ArrayList<int[]> imrcp.forecast.mdss.MetroProcess.m\_oStPvts = new ArrayList()*

Stores the output pavement state arrays for each segment in the scenario

*ArrayList<double[]> imrcp.forecast.mdss.MetroProcess.m\_oTPvts = new ArrayList()*

Stores the output pavement temperature arrays for each segment in the scenario

---

*The documentation for this class was generated from the following file:*

- forecast/mdss/MetroProcess.java

---

## [imrcp.store.MetroStore Class Reference](#)

### Public Member Functions

- **MetroStore ()**
- **void reset ()**
- **FileWrapper getNewFileWrapper ()**
- **void getData (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)**

### Additional Inherited Members

---

#### Detailed Description

**FileCache** that manages the binary files produced by the METRo process.

#### Author

Federal Highway Administration

---

#### Constructor & Destructor Documentation

##### [\*imrcp.store.MetroStore.MetroStore \(\)\*](#)

Default constructor. Sets the zoom level used to spatially index files to 9.

---

#### Member Function Documentation

##### [\*void imrcp.store.MetroStore.getData \(ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime\)\*](#)

Determines the files that match the query and calls **MetroWrapper#getData(int, long, long, long, int, int, int, int, imrcp.store.ImrcpResultSet)** for each of those files.

Reimplemented from **imrcp.system.BaseBlock** (p.96).

##### [\*FileWrapper imrcp.store.MetroStore.getNewFileWrapper \(\)\*](#)

#### Returns

a new **MetroWrapper** with the configure tolerance and zoom level.

Reimplemented from **imrcp.store.FileCache** (p.212).

##### [\*void imrcp.store.MetroStore.reset \(\)\*](#)

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.store.FileCache** (p.214).

---

The documentation for this class was generated from the following file:

- store/MetroStore.java
- 

## imrcp.store.MetroWrapper Class Reference

### Public Member Functions

- **MetroWrapper** (int nTol, int nZoom)
- void **load** (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId, int nTileX, int nTileY) throws Exception
- void **getData** (int nObsTypeId, long lStartTime, long lEndTime, long lRefTime, int nLat1, int nLon1, int nLat2, int nLon2, **ImrcpResultSet** oObsList)
- double **getReading** (int nObsTypeId, long lTimestamp, int nLat, int nLon)
- float[] **getRes** (int nLon, int nLat)
- void **cleanup** (boolean bDelete)

### Additional Inherited Members

---

#### Detailed Description

Parses and creates **Obs** from binary files produced by the METRo process.

#### Author

Federal Highway Administration

---

#### Constructor & Destructor Documentation

##### *imrcp.store.MetroWrapper.MetroWrapper (int nTol, int nZoom)*

Constructs a new **MetroWrapper** with the given parameters

#### Parameters

<i>nTol</i>	Tolerance in decimal degrees scaled to 7 decimal places used for snap algorithms
<i>nZoom</i>	Zoom level used to spatially index files

---

#### Member Function Documentation

##### *void imrcp.store.MetroWrapper.cleanup (boolean bDelete)*

Wrapper for **HashMap#clear()** on **m\_oObsMap**

#### Parameters

<i>bDelete</i>	not used for this <b>FileWrapper</b>
----------------	--------------------------------------

Reimplemented from **imrcp.store.SpatialFileWrapper** (*p.501*).

```
void imrcp.store.MetroWrapper.getData (int nObsTypeId, long lStartTime, long lEndTime, long lRefTime, int nLat1, int nLon1, int nLat2, int nLon2, ImrcpResultSet oObsList)
```

Fills the given **ImrcpResultSet** with **Obs** that match the query.

#### Parameters

<i>nObsTypeId</i>	requested obstype id
<i>lStartTime</i>	start time of the query in milliseconds since Epoch
<i>lEndTime</i>	end time of the query in milliseconds since Epoch
<i>lRefTime</i>	reference time (observations received after this time will not be included)
<i>nLat1</i>	lower bound of latitude in decimal degrees scaled to 7 decimal places
<i>nLon1</i>	lower bound of longitude in decimal degrees scaled to 7 decimal places
<i>nLat2</i>	upper bound of latitude in decimal degrees scaled to 7 decimal places
<i>nLon2</i>	upper bound of longitude in decimal degrees scaled to 7 decimal places
<i>oObsList</i>	list that gets filled with matching obs

```
double imrcp.store.MetroWrapper.getReading (int nObsTypeId, long lTimestamp, int nLat, int nLon)
```

Gets the value at the given location and time for the given observation type

#### Parameters

<i>nObsTypeId</i>	desired observation type
<i>lTimestamp</i>	query time in milliseconds since Epoch
<i>nLat</i>	latitude of query in decimal degrees scaled to 7 decimal places
<i>nLon</i>	longitude of query in decimal degrees scaled to 7 decimal palces

#### Returns

the value at the given location and time if the file contains data that matches the request, otherwise Double.NaN

```
float[] imrcp.store.MetroWrapper.getRes (int nLon, int nLat)
```

Gets the water and snow/ice reservoir at the given location. Wrapper for **TreeMap#get(java.lang.Object)**.

#### Parameters

<i>nLon</i>	longitude of query in decimal degrees scaled to 7 decimal places
<i>nLat</i>	latitude of query in decimal degrees scaled to 7 decimal places

#### Returns

[water reservoir, snow/ice reservoir] if a value exists for the requested location, otherwise null

```
void imrcp.store.MetroWrapper.load (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId, int nTileX, int nTileY) throws Exception
```

Parses the binary METRo file creating **Obs** for the defined values in the file.

Reimplemented from **imrcp.store.SpatialFileWrapper** (*p.501*).

---

*The documentation for this class was generated from the following file:*

- store/MetroWrapper.java
- 

## imrcp.forecast.mlp.MLPBlock Class Reference

### Classes

- class **WorkDelegate**

### Public Member Functions

- void **reset** ()

### Static Public Member Functions

- static void **fillWorkObjects** (ArrayList< **WorkObject** > oWorkObjects, **Network** oNetwork, **Id**[] oIds)
- static REXP **evalToGetError** (RConnection oConn, String sCmd) throws Exception
- static int[] **getIncidentData** (**ImrcpResultSet** oEvents, **OsmWay** oWay, ArrayList< **OsmWay** > oDownstream, long lTimestamp)
- static int **getPrecipitation** (long lTimestamp, int nLat, int nLon, **GriddedFileWrapper** oRtmaFile, **GriddedFileWrapper**[] oPrecipRateFile, double dTemp)
- static int **getPrecip** (double dRate, double dVisInFt, double dTempInF)
- static int **getVisibility** (long lTimestamp, int nLat, int nLon, **GriddedFileWrapper** oRtmaFile, **GriddedFileWrapper** oRapFile)
- static int **getVisibility** (double dVisInFt)
- static double **getTemperature** (long lTimestamp, int nLat, int nLon, **GriddedFileWrapper** oRtmaFile, **GriddedFileWrapper** oNdfdTempFile)
- static int **getWindSpeed** (long lTimestamp, int nLat, int nLon, **GriddedFileWrapper** oRtmaFile, **GriddedFileWrapper** oNdfdWspdFile)
- static int **getDayOfWeek** (Calendar oCal)
- static int **getTimeOfDay** (Calendar oCal)

### Static Public Attributes

- static final String **g\_sRDataFile**
- static final String **g\_sRObjects**
- static final String **g\_sRHost**
- static final String **g\_sLongTsPredFf**
- static final String **HISTDATHEADER** = "Timestamp,Id,Precipitation,Visibility,Direction,Temperature,WindSpeed,DayOfWeek,TimeOfDay, Lanes,SpeedLimit,Curve,HOV,PavementCondition,OnRamps,OffRamps,IncidentDownstream, IncidentOnLink,LanesClosedOnLink,LanesClosedDownstream,WorkzoneOnLink,WorkzoneDown stream,SpecialEvents,Flow,Speed,Occupancy,road,contraflow"
- static final String **LONGTSHEADER** = "timestamplist,speed"

### Protected Member Functions

- abstract void **processWork** (**Work** oWork)
- abstract void **save** (long lTimestamp)
- abstract void **finishWork** (long lTimestamp)

### Protected Attributes

- int **g\_nThreads**
- int **m\_nOffset**

- int **m\_nPeriod**
- String **m\_sLocalDir**
- String **m\_sHostDir**
- ArrayDeque< Work > **m\_oWorkQueue**
- **WorkDelegate m\_oDelegate**
- String **m\_sNetwork**
- String **m\_sTz**
- String **m\_sInputFf**
- String **m\_sLongTsLocalDir**
- String **m\_sLongTsHostDir**

#### Static Protected Attributes

- static final double **g\_dRAINTEMPK**
- static final double **g\_dSNOWTEMPK**
- static final double **g\_dRAINTEMPF**
- static final double **g\_dSNOWTEMPF**
- static final **WeatherStore g\_oRtmaStore**
- static final **WeatherStore g\_oRAPStore**
- static final **WeatherStore g\_oNdfdTempStore**
- static final **WeatherStore g\_oNdfdWspdStore**

#### Additional Inherited Members

---

#### Detailed Description

Base class with methods and variables used by the MLP models that make online real time traffic speed predictions and long time series predictions.

#### Author

Federal Highway Administration

---

#### Member Function Documentation

**static REXP imrcp.forecast.mlp.MLPBlock.evalToGetError (RConnection oConn, String sCmd) throws Exception [static]**

Used to get better detailed error messages from evaluating R commands through Rserve and the **org.rosuda.REngine.Rserve.RConnection** interface

#### Parameters

<i>oConn</i>	RConnection used to send R command
<i>sCmd</i>	R command to send

#### Returns

the REXP from calling  
**org.rosuda.REngine.Rserve.RConnection#parseAndEval (java.lang.String)**

#### Exceptions

<i>Exception</i>	If the REXP from <b>org.rosuda.REngine.Rserve.RConnection#parseAndEval (java.lang.String)</b> is of the type "try-error" an
------------------	--

	Exception with <code>org.rosuda.REngine.REXPasString()</code> as the message
--	---

`static void imrcp.forecast.mlp.MLPBlock.fillWorkObjects (ArrayList<WorkObject> oWorkObjects, Network oNetwork, Id[] olds) [static]`

Accumulates the `imrcp.geosrv.osm.OsmWay` and the necessary metadata and creates `WorkObject`s to add to the given list

#### Parameters

<code>oWorkObjects</code>	List to be filled with the created WorkObjects
<code>oNetwork</code>	The Network being processed
<code>old</code> s	Contains specific Ids to create WorkObjects for. Usually null and only used for testing.

`abstract void imrcp.forecast.mlp.MLPBlock.finishWork (long lTimestamp) [abstract], [protected]`

Child classes implement this function which is called once all of the threads have finished `processWork (imrcp.forecast.mlp.Work)`

#### Parameters

<code>lTimestamp</code>	run time of the MLP model in milliseconds since Epoch
Reimplemented in <code>imrcp.forecast.mlp.MLPPredict</code> (p.357), and <code>imrcp.forecast.mlp.MLPUpdate</code> (p.365).	

`static int imrcp.forecast.mlp.MLPBlock.getDayOfWeek (Calendar oCal) [static]`

Gets the Day of Week category from the given Calendar

#### Parameters

<code>oCal</code>	Calendar object with its time already set
-------------------	---

#### Returns

1 = weekend, 2 = weekday

`static int[] imrcp.forecast.mlp.MLPBlock.getIncidentData (ImrcpResultSet oEvents, OsmWay oWay, ArrayList<OsmWay> oDownstream, long lTimestamp) [static]`

Creates and returns an int[] that contains flags and values for incident and work zone data used in the MLP model

#### Parameters

<code>oEvents</code>	Contains the <code>imrcp.system.ObsType#EVT</code> observations that could be associated with the different Ways
<code>oWay</code>	roadway that the MLP model is being ran on
<code>oDownstream</code>	roadways that are downstream <code>oWay</code>
<code>lTimestamp</code>	time in milliseconds since Epoch of the current record of the input files being created

### Returns

[incident on link flag, incident downstream flag, road work on link flag, road work downstream flag, lanes affected on link, lanes affected downstream]

**static int imrcp.forecast.mlp.MLPBlock.getPrecip (double dRate, double dVisInFt, double dTempInF) [static]**

Gets the precipitation category from the given precipitation rate, visibility, and temperature

### Parameters

<i>dRate</i>	precipitation rate in mm hr
<i>dVisInFt</i>	visibility in feet
<i>dTempInF</i>	air temperature in F

### Returns

1 = no precip, 2 = light rain, 3 = moderate rain, 4 = heavy rain, 5 = light snow, 6 = moderate snow, 7 = heavy snow

**static int imrcp.forecast.mlp.MLPBlock.getPrecipitation (long lTimestamp, int nLat, int nLon, GriddedFileWrapper oRtmaFile, GriddedFileWrapper[] oPrecipRateFile, double dTemp) [static]**

Gets the precipitation category from the given files for the given location and time

### Parameters

<i>lTimestamp</i>	time in milliseconds since Epoch used to query the given files
<i>nLat</i>	latitude of location in decimal degrees scaled to 7 decimal places
<i>nLon</i>	longitude of location in decimal degrees scaled to 7 decimal places
<i>oRtmaFile</i>	RTMA file used to get the visibility if needed to determine snow intensity
<i>oPrecipRateFile</i>	RAP or files that contain precipitation rate values in mm/sec
<i>dTemp</i>	air temperature at the location in K

### Returns

1 = no precip, 2 = light rain, 3 = moderate rain, 4 = heavy rain, 5 = light snow, 6 = moderate snow, 7 = heavy snow

**static double imrcp.forecast.mlp.MLPBlock.getTemperature (long lTimestamp, int nLat, int nLon, GriddedFileWrapper oRtmaFile, GriddedFileWrapper oNdfdTempFile) [static]**

Gets the air temperature for the given location and time. First tries to get the air temperature from the given RTMA file, if it cannot get the air temperature from RTMA, tries to get it from the given NDFD file.

### Parameters

<i>lTimestamp</i>	time in milliseconds since Epoch used to query the given files
<i>nLat</i>	latitude of location in decimal degrees scaled to 7 decimal places
<i>nLon</i>	longitude of location in decimal degrees scaled to 7 decimal places

<i>oRtmaFile</i>	RTMA file valid for the given timestamp
<i>oNdfdTempFile</i>	NDFD temperature file valid for the given timestamp

*Returns*

*static int imrcp.forecast.mlp.MLPBlock.getTimeOfDay (Calendar oCal) [static]*

Gets the Time of Day category from the given Calendar

*Parameters*

<i>oCal</i>	Calendar object with its time already set
-------------	---

*Returns*

1 = morning, 2 = am peak, 3 = offpeak, 4 = pm peak, 5 = night

*static int imrcp.forecast.mlp.MLPBlock.getVisibility (double dVisInFt) [static]*

Gets the visibility category for the given visibility

*Parameters*

<i>dVisInFt</i>	visibility in feet
-----------------	--------------------

*Returns*

1 = clear visibility, 2 = reduced visibility, 3 = low visibility

*static int imrcp.forecast.mlp.MLPBlock.getVisibility (long lTimestamp, int nLat, int nLon, GriddedFileWrapper oRtmaFile, GriddedFileWrapper oRapFile) [static]*

Get the visibility category for the given location and time

*Parameters*

<i>lTimestamp</i>	time in milliseconds since Epoch used to query the given files
<i>nLat</i>	latitude of location in decimal degrees scaled to 7 decimal places
<i>nLon</i>	longitude of location in decimal degrees scaled to 7 decimal places
<i>oRtmaFile</i>	RTMA file valid for the given timestamp
<i>oRapFile</i>	RAP file valid for the given timestamp

*Returns*

1 = clear visibility, 2 = reduced visibility, 3 = low visibility

*static int imrcp.forecast.mlp.MLPBlock.getWindSpeed (long lTimestamp, int nLat, int nLon, GriddedFileWrapper oRtmaFile, GriddedFileWrapper oNdfdWspdFile) [static]*

Gets the wind speed for the given location and time. First tries to get the wind speed from the given RTMA file, if it cannot get the wind speed from RTMA, tries to get it from the given NDFD file.

*Parameters*

<i>lTimestamp</i>	time in milliseconds since Epoch used to query the given files
<i>nLat</i>	latitude of location in decimal degrees scaled to 7 decimal places

<i>nLon</i>	longitude of location in decimal degrees scaled to 7 decimal places
<i>oRtmaFile</i>	RTMA file valid for the given timestamp
<i>oNdfdWspdFile</i>	NDFD wind speed file valid for the given timestamp

*Returns*

*abstract void imrcp.forecast.mlp.MLPBlock.processWork (Work oWork) [abstract], [protected]*

Child classes implement this function to process the given **Work** object. The processing is done by passing R commands through an RConnection to to Rserve

*Parameters*

<i>oWork</i>	<b>Work</b> to process
--------------	------------------------

Reimplemented in **imrcp.forecast.mlp.MLPPredict** (p.357), and **imrcp.forecast.mlp.MLPUpdate** (p.365).

*void imrcp.forecast.mlp.MLPBlock.reset ()*

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.system.BaseBlock** (p.98).

Reimplemented in **imrcp.forecast.mlp.MLPPredict** (p.357), and **imrcp.forecast.mlp.MLPUpdate** (p.366).

*abstract void imrcp.forecast.mlp.MLPBlock.save (long lTimestamp) [abstract], [protected]*

Child classes implement this function to save the outputs for the given timestamp.

*Parameters*

<i>lTimestamp</i>	run time of the MLP model in milliseconds since Epoch
-------------------	---

Reimplemented in **imrcp.forecast.mlp.MLPPredict** (p.358), and **imrcp.forecast.mlp.MLPUpdate** (p.366).

## Member Data Documentation

*final double imrcp.forecast.mlp.MLPBlock.g\_dRAINTEMPF [static], [protected]*

Temperature cutoff for precipitation type being inferred as rain, in F

*final double imrcp.forecast.mlp.MLPBlock.g\_dRAINTEMPK [static], [protected]*

Temperature cutoff for precipitation type being inferred as rain, in K

*final double imrcp.forecast.mlp.MLPBlock.g\_dSNOWTEMPF [static], [protected]*

Temperature cutoff for precipitation type being inferred as snow, in F

*final double imrcp.forecast.mlp.MLPBlock.g\_dSNOWTEMPK [static],  
[protected]*

Temperature cutoff for precipitation type being inferred as snow, in K

*int imrcp.forecast.mlp.MLPBlock.g\_nThreads [protected]*

Number of threads to use for each period of execution of the MLP model

*final WeatherStore imrcp.forecast.mlp.MLPBlock.g\_oNdfdTempStore [static],  
[protected]*

Instance of the store that manages NDFD temperature files

*final WeatherStore imrcp.forecast.mlp.MLPBlock.g\_oNdfdWspdStore [static],  
[protected]*

Instance of the store that manages NDFD wind speed files

*final WeatherStore imrcp.forecast.mlp.MLPBlock.g\_oRAPStore [static],  
[protected]*

Instance of the store that manages RAP files

*final WeatherStore imrcp.forecast.mlp.MLPBlock.g\_oRtmaStore [static],  
[protected]*

Instance of the store that manages RTMA files

*final String imrcp.forecast.mlp.MLPBlock.g\_sLongTsPredFf [static]*

Format String used to generate Long Time Series speed prediction files

*final String imrcp.forecast.mlp.MLPBlock.g\_sRDataFile [static]*

Path of the R script file that contains methods for the MLP model

*final String imrcp.forecast.mlp.MLPBlock.g\_sRHost [static]*

IP address of the server running Rserve

*final String imrcp.forecast.mlp.MLPBlock.g\_sRObjects [static]*

Path of the Rdata file that contains the variables and transition matrices determined in training the MLP model

*final String imrcp.forecast.mlp.MLPBlock.HISTDATHEADER =  
"Timestamp,Id,Precipitation,Visibility,Direction,Temperature,WindSpeed,DayOfWeek,TimeOfDay,Lanes,SpeedLimit,Curve,HOV,PavementCondition,OnRamps,OffRamps,IncidentDownstream,IncidentOnLink,LanesClosedOnLink,LanesClosedDownstream,WorkzoneOnLink,WorkzoneDownstream,SpecialEvents,Flow,Speed,Occupancy,road,contraflow" [static]*

Header of the histdat CSV files

*final String imrcp.forecast.mlp.MLPBlock.LONGTSHEADER =  
"timestamplist,speed" [static]*

Header of the long\_ts CSV files

*int imrcp.forecast.mlp.MLPBlock.m\_nOffset [protected]*

Schedule offset from midnight in seconds

*int imrcp.forecast.mlp.MLPBlock.m\_nPeriod [protected]*

Period of execution in seconds

*WorkDelegate imrcp.forecast.mlp.MLPBlock.m\_oDelegate [protected]*

Delegate object used to execute **Work** in the queue in a different thread

*ArrayDeque<Work> imrcp.forecast.mlp.MLPBlock.m\_oWorkQueue [protected]*

Queue that stores the work to be done.

*String imrcp.forecast.mlp.MLPBlock.m\_sHostDir [protected]*

Base directory for MLP files on the server running Rserve

*String imrcp.forecast.mlp.MLPBlock.m\_sInputFf [protected]*

Format String used to generate the histdat files per thread

*String imrcp.forecast.mlp.MLPBlock.m\_sLocalDir [protected]*

Base directory for MLP files on the local server running IMRCP

*String imrcp.forecast.mlp.MLPBlock.m\_sLongTsHostDir [protected]*

Base directory for long time series files on the server running Rserve

*String imrcp.forecast.mlp.MLPBlock.m\_sLongTsLocalDir [protected]*

Base directory for long time series files on the local server running IMRCP

*String imrcp.forecast.mlp.MLPBlock.m\_sNetwork [protected]*

Id of the **imrcp.geosrv.Network** this instance is processing

*String imrcp.forecast.mlp.MLPBlock.m\_sTz [protected]*

Id of the time zone the network this instance is processing is a part of. Used to look up time zones by the **method**

---

*The documentation for this class was generated from the following file:*

- forecast/mlp/MLPBlock.java
- 

## imrcp.store.MLPCsv Class Reference

### Public Member Functions

- **MLPCsv** (int[] nObsTypes)
- **void load** (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId)  
throws Exception

### Additional Inherited Members

---

#### Detailed Description

Parses and creates **Obs** CSV files created by the MLP traffic processes

#### *Author*

Federal Highway Administration

---

#### Constructor & Destructor Documentation

*imrcp.store.MLPCsv.MLPCsv (int[] nObsTypes)*

Constructor a new **MLPCsv** with the given observation types

##### *Parameters*

<i>nObsTypes</i>	array of observation type ids this file provides
------------------	--

---

#### Member Function Documentation

*void imrcp.store.MLPCsv.load (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId) throws Exception*

Parses the MLP CSV prediction file and creates **ObsType#SPDLNK Obs** from the predictions and **ObsType#TRFLNK** by looking up the speed limit of the associated roadway segment and determine what percent of the speed limit the speed prediction is.

Reimplemented from **imrcp.store.CsvWrapper** (*p.125*).

---

*The documentation for this class was generated from the following file:*

- store/MLPCsv.java
- 

#### imrcp.forecast.mlp.MLPHurricane Class Reference

##### Classes

- class **Delegate**
- class **HurData**
- class **HurWork**

##### Public Member Functions

- void **reset** ()
- boolean **start** () throws Exception
- void **process** (String[] sMessage)
- long **getHurWarningTime** (String sFile) throws Exception
- void **execute** ()
- boolean **buildData** (String sNhcFile)

##### Additional Inherited Members

---

#### Detailed Description

#### *Author*

Federal Highway Administration

---

## Member Function Documentation

### `boolean imrcp.forecast.mlp.MLPHurricane.buildData (String sNhcFile)`

Sets up the necessary data structures and queries ObsView for speed observations that will be used by all of the threads used to process the MLP Hurricane model for the given file. Then executes the `imrcp.forecast.mlp.MLPHurricane.Delegate` that handles the multi-threaded execution of the model.

#### Parameters

<code>sNhcFile</code>	NHC hurricane forecast file to process
-----------------------	--

#### Returns

false if the situations occur that allow the file to not be processed, true if a `imrcp.forecast.mlp.MLPHurricane.Delegate` is created and executed.

### `void imrcp.forecast.mlp.MLPHurricane.execute ()`

If the file is not empty and a file is not currently being processed, gets the first file out of the queue and calls `buildData (java.lang.String)` with that file as input. After `buildData (java.lang.String)` returns, Checks the working dir for any stale data files that can be deleted.

Reimplemented from `imrcp.system.BaseBlock` (p.95).

### `long imrcp.forecast.mlp.MLPHurricane.getHurWarningTime (String sFile) throws Exception`

Determines the first time a hurricane warning was issued for the NHC storm the given file is associated with.

#### Parameters

<code>sFile</code>	NHC hurricane forecast file
--------------------	-----------------------------

#### Returns

Timestamp in milliseconds since Epoch when the first hurricane warning was issued for the associated storm. If no hurricane warning has been issued return `Long.MIN_VALUE`

#### Exceptions

<code>Exception</code>
------------------------

### `void imrcp.forecast.mlp.MLPHurricane.process (String[] sMessage)`

Called when a message from `imrcp.system.Directory` is received. If the message is "file download" adds the files to the queue and if the queue is then not empty calls `execute ()`

#### Parameters

<code>sMessage</code>
-----------------------

Reimplemented from `imrcp.system.BaseBlock` (p.97).

### `void imrcp.forecast.mlp.MLPHurricane.reset ()`

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.system.BaseBlock** (p.98).

**boolean imrcp.forecast.mlp.MLPHurricane.start () throws Exception**

Sets a schedule to execute on a fixed interval.

*Returns*

true if no Exceptions are thrown

*Exceptions*

<i>Exception</i>	
------------------	--

Reimplemented from **imrcp.system.BaseBlock** (p.99).

---

*The documentation for this class was generated from the following file:*

- forecast/mlp/MLPHurricane.java
- 

## imrcp.forecast.mlp.MLPMetadata Class Reference

### Public Attributes

- String **m\_sId**
  - String **m\_sSpdLimit**
  - int **m\_nHOV**
  - int **m\_nDirection**
  - int **m\_nCurve**
  - int **m\_nOffRamps**
  - int **m\_nOnRamps**
  - int **m\_nPavementCondition**
  - String **m\_sRoad**
  - int **m\_nSpecialEvents**
  - int **m\_nLanes**
- 

### Detailed Description

Contains metadata for a roadway segment used in the MLP models

### Author

Federal Highway Administration

---

### Member Data Documentation

**int imrcp.forecast.mlp.MLPMetadata.m\_nCurve**

Curve flag. 0 = no curve 1 = curve

*See also*

**imrcp.geosrv.osm.OsmWay::getCurve()**

**int imrcp.forecast.mlp.MLPMetadata.m\_nDirection**

Direction of travel enumeration. 1 = east 2 = south 3 = west 4 = north

**See also**

[imrcp.geosrv.osm.OsmWay::getDirection\(\)](#)

*int imrcp.forecast.mlp.MLPMetadata.m\_nHOV*

HOV flag. 0 = no HOV lane 1 = the segment contains an HOV lane

*int imrcp.forecast.mlp.MLPMetadata.m\_nLanes*

Number of lanes

*int imrcp.forecast.mlp.MLPMetadata.m\_nOffRamps*

Number of off ramps the segment has

*int imrcp.forecast.mlp.MLPMetadata.m\_nOnRamps*

Number of on ramps the segment has

*int imrcp.forecast.mlp.MLPMetadata.m\_nPavementCondition*

Pavement Condition enumeration 1 = good condition 2 = average condition 3 = poor condition

*int imrcp.forecast.mlp.MLPMetadata.m\_nSpecialEvents*

Special event flag. 0 = no special event present 1 = special event present

*String imrcp.forecast.mlp.MLPMetadata.m\_sId*

IMRCP Id as a String.

**See also**

[imrcp.system.Id::toString\(\)](#)

*String imrcp.forecast.mlp.MLPMetadata.m\_sRoad*

Name of the road

*String imrcp.forecast.mlp.MLPMetadata.m\_sSpdLimit*

Speed limit in mph

---

*The documentation for this class was generated from the following file:*

- forecast/mlp/MLPMetadata.java
- 

## [imrcp.forecast.mlp.MLPredict Class Reference](#)

### Public Member Functions

- void **reset** ()
- boolean **start** () throws Exception
- void **createWork** (ArrayList<**WorkObject**> oWorkObjects, ArrayList<**Work**> oRWorks, **Network** oNetwork) throws Exception
- void **process** (String[] sMessage)
- void **run** ()

## Protected Member Functions

- void **buildData** (int nThread, long lStartTime, long lEndTime, **ImrcpObsResultSet** oSpeedData, **ImrcpResultSet** oIncidentData, **Work** oWorkObjs) throws Exception
- void **save** (long lTimestamp)
- void **processWork** (**Work** oRWork)
- void **finishWork** (long lTimestamp)

## Protected Attributes

- String Builder **m\_oROutput** = new String Builder()
- ArrayList<**WorkObject**> **m\_oOutputs** = new ArrayList()
- ArrayList<**Prediction**> **m\_oPreds** = new ArrayList()
- **FilenameFormatter** **m\_ofilenameFormat**
- int **m\_nForecastMinutes**
- int **m\_nA**
- int **m\_nB**
- double **m\_dBTol**
- long **m\_lLastRun**
- long **m\_lPredStep**

## Additional Inherited Members

---

### Detailed Description

Manages running the real time online MLP model for traffic speed predictions.

#### Author

Federal Highway Administration

---

### Member Function Documentation

`void imrcp.forecast.mlp.MLPPredict.buildData (int nThread, long lStartTime, long lEndTime, ImrcpObsResultSet oSpeedData, ImrcpResultSet oIncidentData, Work oWorkObjs) throws Exception [protected]`

Creates the input histdat file for the given parameters

#### Parameters

<i>nThread</i>	thread number
<i>lStartTime</i>	time in milliseconds since Epoch representing start time of the data in two ResultSets
<i>lEndTime</i>	time in milliseconds since Epoch representing end time of the data in the two ResultSets. This is also the run time of the model
<i>oSpeedData</i>	ResultSet containing speed observations
<i>oIncidentData</i>	ResultSet containing incident, workzone, and flooded road data
<i>oWorkObjs</i>	Contains objects to run the model on

#### Exceptions

<i>Exception</i>	
------------------	--

`void imrcp.forecast.mlp.MLPPredict.createWork (ArrayList< WorkObject > oWorkObjects, ArrayList< Work > oRWorks, Network oNetwork) throws Exception`

Fills the given lists with the necessary objects to run the MLP online real time prediction model for the given Network.

*Parameters*

<code>oWorkObjects</code>	List to fill with WorkObjects
<code>oRWorks</code>	List to fill with Works
<code>oNetwork</code>	Network the model is being ran on

*Exceptions*

<code>Exception</code>
------------------------

`void imrcp.forecast.mlp.MLPPredict.finishWork (long lTimestamp) [protected]`

Wrapper for `save (long)` then handles setting the status of the block

*Parameters*

<code>lTimestamp</code>
-------------------------

Reimplemented from `imrcp.forecast.mlp.MLPBlock` (p.346).

`void imrcp.forecast.mlp.MLPPredict.process (String[] sMessage)`

Called when a message from `imrcp.system.Directory` is received. If the message is "files ready", queues up a week's worth of reruns using the time in the message as the starting point. This is intended to be used only if `m_bRealTime` is false

*Parameters*

<code>sMessage</code>	[BaseBlock message is from, message name, timestamp in milliseconds since Epoch of the run time, directory of the long_ts files]
-----------------------	--

Reimplemented from `imrcp.system.BaseBlock` (p.97).

`void imrcp.forecast.mlp.MLPPredict.processWork (Work oRWork) [protected]`

Runs the MLP real time online prediction model for the segments in the given `Work` object by calling R code through the `org.rosuda.REngine.Rserve.RConnection` and Rserve interfaces.

*Parameters*

<code>oRWork</code>	Contains the data to run the model on
---------------------	---------------------------------------

Reimplemented from `imrcp.forecast.mlp.MLPBlock` (p.349).

`void imrcp.forecast.mlp.MLPPredict.reset ()`

Sets configurable member variables from the `BlockConfig` object, should be overridden by child classes.

Reimplemented from `imrcp.forecast.mlp.MLPBlock` (p.349).

`void imrcp.forecast.mlp.MLPPredict.run ()`

Executes one run of the MLP real time online prediction model for the configured Network. If `m_bRealTime` is true, the current time is put into the queue. The model is then ran for the first time in the queue.

Reimplemented from `imrcp.system.BaseBlock` (p.98).

`void imrcp.forecast.mlp.MLPPredict.save (long lTimestamp) [protected]`

Runs "MLP B" (applies speed predictions to close downstream segments that did not get a prediction for whatever reason) then writes predictions from the MLP model to disk.

*Parameters*

<code>lTimestamp</code>	MLP run time in milliseconds since Epoch
-------------------------	--

Reimplemented from `imrcp.forecast.mlp.MLPBlock` (p.349).

`boolean imrcp.forecast.mlp.MLPPredict.start () throws Exception`

Ensures the necessary directories exist. Reads any values in the queue file and adds them to the queue. If a value was added then the histdat files for that time are created. If `m_bRealTime` is true an `InitDelegate` is scheduled to run in 10 seconds, otherwise sets a schedule to execute on a fixed interval.

*Returns*

true if no Exceptions are thrown

*Exceptions*

<code>Exception</code>	
------------------------	--

Reimplemented from `imrcp.system.BaseBlock` (p.99).

---

## Member Data Documentation

`double imrcp.forecast.mlp.MLPPredict.m_dBTol [protected]`

Distance in decimal degrees scaled to 7 decimal places to determine if a downstream segment is "close" when applying predictions for MLPB

`long imrcp.forecast.mlp.MLPPredict.m_lLastRun [protected]`

Keeps track of the last time the model was ran

`long imrcp.forecast.mlp.MLPPredict.m_lPredStep [protected]`

Time in milliseconds between each speed prediction

`int imrcp.forecast.mlp.MLPPredict.m_nA [protected]`

Number of segments that the online prediction R code successfully ran for in a single run of the model

`int imrcp.forecast.mlp.MLPPredict.m_nB [protected]`

Number of segments that the online prediction R code did not successfully run for but got a prediction applied to it from a close upstream segment that did have a successful online prediction.

`int imrcp.forecast.mlp.MLPPredict.m_nForecastMinutes [protected]`

Length of forecasts generated in minutes

*FilenameFormatter*

`imrcp.forecast.mlp.MLPPredict.m_oFilenameFormat [protected]`

Object used to create time dependent file names to save on disk

*ArrayList<WorkObject> imrcp.forecast.mlp.MLPPredict.m\_oOutputs = new ArrayList() [protected]*

Stores the WorkObjects for the current run

*ArrayList<Prediction> imrcp.forecast.mlp.MLPPredict.m\_oPreds = new ArrayList() [protected]*

Stores the Predictions for the current run

*StringBuilder imrcp.forecast.mlp.MLPPredict.m\_oROutput = new StringBuilder() [protected]*

Contains the predicted speed values computed in R

---

*The documentation for this class was generated from the following file:*

- forecast/mlp/MLPPredict.java
- 

## imrcp.forecast.mlp.MLPProcess Class Reference

### Public Member Functions

- **MLPProcess ()**
- **void process (Scenario oScenario, String sScenariosDir, Logger oLogger) throws Exception**

### Public Attributes

- **HashMap< String, double[]> m\_oOutputs**
- 

### Detailed Description

Manages running the oneshot MLP model for traffic speed predictions used by Scenarios.

#### Author

Federal Highway Administration

---

### Constructor & Destructor Documentation

*imrcp.forecast.mlp.MLPProcess.MLPProcess ()*

Default Constructor. Does nothing.

---

### Member Function Documentation

*void imrcp.forecast.mlp.MLPProcess.process (Scenario oScenario, String sScenariosDir, Logger oLogger) throws Exception*

Runs the oneshot MLP model for the given scenario. To do this it generates the historic speed data to run the long time series update MLP model which is used as input to the oneshot model.

#### Parameters

<i>oScenario</i>	Scenario to generate speed predictions for
<i>sScenariosDir</i>	Base directory for scenarios
<i>oLogger</i>	Logger

#### Exceptions

<i>Exception</i>
------------------

---

## Member Data Documentation

### *HashMap<String, double[]> imrcp.forecast.mlp.MLPProcess.m\_oOutputs*

Stores the speed predictions from the oneshot model by roadway segment id.

---

*The documentation for this class was generated from the following file:*

- forecast/mlp/MLPProcess.java
- 

## imrcp.forecast.mlp.MLPRecord Class Reference

### Public Member Functions

- **MLPRecord ()**
- **MLPRecord (MLPRecord oRecord)**
- **MLPRecord (CsvReader oIn, SimpleDateFormat oSdf) throws Exception**
- **void writeRecord (Writer oOut, SimpleDateFormat oSdf) throws Exception**
- **int compareTo (MLPRecord o)**

### Public Attributes

- String **m\_sId**
- int **m\_nPrecipitation**
- int **m\_nVisibility**
- int **m\_nDirection**
- int **m\_nTemperature**
- int **m\_nWindSpeed**
- int **m\_nDayOfWeek**
- int **m\_nTimeOfDay**
- int **m\_nLanes**
- String **m\_sSpeedLimit**
- int **m\_nCurve**
- int **m\_nHOV**
- int **m\_nPavementCondition**
- int **m\_nOnRamps**
- int **m\_nOffRamps**
- int **m\_nIncidentDownstream**
- int **m\_nIncidentOnLink**
- int **m\_nLanesClosedOnLink**
- int **m\_nLanesClosedDownstream**
- int **m\_nWorkzoneOnLink**
- int **m\_nWorkzoneDownstream**
- int **m\_nSpecialEvents**

- int **m\_nFlow** = Integer.MIN\_VALUE
  - int **m\_nSpeed** = Integer.MIN\_VALUE
  - int **m\_nOccupancy** = Integer.MIN\_VALUE
  - long **m\_lTimestamp**
  - String **m\_sRoad**
  - int **m\_nContraflow** = -1
- 

### Detailed Description

Contains the data needed for a single record of the histdat files used in the MLP models

#### *Author*

Federal Highway Administration

---

### Constructor & Destructor Documentation

*imrcp.forecast.mlp.MLPRecord.MLPRecord ()*

Default constructor. Does nothing.

*imrcp.forecast.mlp.MLPRecord.MLPRecord (MLPRecord oRecord)*

Copy constructor. Copies the values of the given **MLPRecord**

#### *Parameters*

<i>oRecord</i>	<b>MLPRecord</b> to copy
----------------	--------------------------

*imrcp.forecast.mlp.MLPRecord.MLPRecord (CsvReader oIn, SimpleDateFormat oSdf)*  
*throws Exception*

Constructs a **MLPRecord** from a line from a histdat file using the given CsvReader which has already called **imrcp.system.CsvReader#readLine()** and is ready to parse the current line of the file.

#### *Parameters*

<i>oIn</i>	CsvReader used to parse lines of the histdat file
<i>oSdf</i>	Object used to parse date/time string

#### *Exceptions*

<i>Exception</i>	
------------------	--

---

### Member Function Documentation

*int imrcp.forecast.mlp.MLPRecord.compareTo (MLPRecord o)*

Compares MLPRecords by Id then timestamp.

#### *See also*

`java.lang.Comparable::compareTo(java.lang.Object)`

`void imrcp.forecast.mlp.MLPRecord.writeRecord (Writer oOut, SimpleDateFormat oSdf) throws Exception`

Writes the record to the given Writer by calling `format(java.text.SimpleDateFormat)`

*Parameters*

<code>oOut</code>	Writer to write the record to
<code>oSdf</code>	Date/time formatter object

*Exceptions*

<code>Exception</code>
------------------------

---

## Member Data Documentation

`long imrcp.forecast.mlp.MLPRecord.m_lTimestamp`

Timestamp in milliseconds since Epoch of the speed, volume, and occupancy observations

`int imrcp.forecast.mlp.MLPRecord.m_nContraflow = -1`

Contraflow enumeration. -1 = no contraflow data 0 = contraflow is not active 1 = contraflow is active

`int imrcp.forecast.mlp.MLPRecord.m_nCurve`

Curve flag. 0 = no curve 1 = curve

`int imrcp.forecast.mlp.MLPRecord.m_nDayOfWeek`

Day of week enumeration 1 = weekend 2 = weekday

`int imrcp.forecast.mlp.MLPRecord.m_nDirection`

Direction of travel enumeration. 1 = east 2 = south 3 = west 4 = north

`int imrcp.forecast.mlp.MLPRecord.m_nHOV`

HOV flag. 0 = no HOV lane 1 = the segment contains an HOV lane

`int imrcp.forecast.mlp.MLPRecord.m_nIncidentDownstream`

Incident downstream flag. 0 = no incidents downstream of the associated roadway segment 1 = incident present downstream of the associated roadway segment

`int imrcp.forecast.mlp.MLPRecord.m_nIncidentOnLink`

Incident on link flag. 0 = no incidents on the associated roadway segment 1 = incident present on the associated roadway segment

`int imrcp.forecast.mlp.MLPRecord.m_nLanes`

Number of lanes of the associated roadway segment

`int imrcp.forecast.mlp.MLPRecord.m_nLanesClosedDownstream`

Number of lanes closed downstream of the associated roadway segment due to incidents or work zones

`int imrcp.forecast.mlp.MLPRecord.m_nLanesClosedOnLink`

Number of lanes closed on the associated roadway segment due to incidents or work zones

*int imrcp.forecast.mlp.MLPRecord.m\_nOffRamps*

Number of off ramps of the associated roadway segment

*int imrcp.forecast.mlp.MLPRecord.m\_nOnRamps*

Number of on ramps of the associated roadway segment

*int imrcp.forecast.mlp.MLPRecord.m\_nPavementCondition*

Pavement Condition enumeration 1 = good condition 2 = average condition 3 = poor condition

*int imrcp.forecast.mlp.MLPRecord.m\_nPrecipitation*

Precipitation category enumeration 1 = no precip 2 = light rain 3 = moderate rain 4 = heavy rain 5 = light snow 6 = moderate snow 7 = heavy snow

*int imrcp.forecast.mlp.MLPRecord.m\_nSpecialEvents*

Special event flag. 0 = no special event present 1 = special event present

*int imrcp.forecast.mlp.MLPRecord.m\_nTemperature*

Air temperature in F

*int imrcp.forecast.mlp.MLPRecord.m\_nTimeOfDay*

Time of day enumeration 1 = morning 2 = am peak 3 = offpeak 4 = pm peak 5 = night

*int imrcp.forecast.mlp.MLPRecord.m\_nVisibility*

Visibility category enumeration 1 = clear visibility 2 = reduced visibility 3 = low visibility

*int imrcp.forecast.mlp.MLPRecord.m\_nWindSpeed*

Wind speed in mph

*int imrcp.forecast.mlp.MLPRecord.m\_nWorkzoneDownstream*

**Work** zone downstream flag. 0 = no work zone downstream of the associated roadway segment 1 = work zone present downstream of the associated roadway segment

*int imrcp.forecast.mlp.MLPRecord.m\_nWorkzoneOnLink*

**Work** zone on link flag. 0 = no work zone on the associated roadway segment 1 = work zone present on the associated roadway segment

*String imrcp.forecast.mlp.MLPRecord.m\_sId*

IMRCP Id of the associated segment. Created from `imrcp.system.Id#toString()`

*String imrcp.forecast.mlp.MLPRecord.m\_sRoad*

Name of the associated roadway segment

*String imrcp.forecast.mlp.MLPRecord.m\_sSpeedLimit*

Speed limit of the associated roadway segment

---

*The documentation for this class was generated from the following file:*

- forecast/mlp/MLPRecord.java
-

## imrcp.store.MLPStore Class Reference

### Protected Member Functions

- **FileWrapper getNewFileWrapper ()**

### Additional Inherited Members

---

#### Detailed Description

**FileCache** that manages CSV files produced by the different MLP traffic processes

#### Author

Federal Highway Administration

---

#### Member Function Documentation

**FileWrapper imrcp.store.MLPStore.getNewFileWrapper () [protected]**

##### Returns

a new **MLPCsv** with the configured observation types

Reimplemented from **imrcp.store.CsvStore** (*p.124*).

---

*The documentation for this class was generated from the following file:*

- store/MLPStore.java
- 

## imrcp.forecast.mlp.MLPUpdate Class Reference

### Public Member Functions

- void **reset ()**
- boolean **start ()** throws Exception
- void **queue** (String sDate, StringBuilder sBuffer) throws Exception
- void **execute ()**
- void **run ()**
- long **createWork** (ArrayList< **WorkObject** > oWorkObjects, ArrayList< **Work** > oRWorks) throws Exception

### Static Public Member Functions

- static void **updateMonthlyHistDat** (long lEarliestTs, ArrayList< **Work** > oRWorks, long lEndTime, String sTimeZone, int[] nBb, String sFilenameFormat, String sDir) throws Exception

### Public Attributes

- int **m\_nWeeksBack**

### Protected Member Functions

- void **save** (long lTimestamp)
- void **processWork** (**Work** oRWork)

- void **finishWork** (long lTimestamp)

## Additional Inherited Members

---

### Detailed Description

Manages running the long time series update MLP model for traffic speed predictions once a week

#### Author

Federal Highway Administration

---

### Member Function Documentation

*long imrcp.forecast.mlp.MLPUpdate.createWork (ArrayList< WorkObject > oWorkObjects, ArrayList< Work > oRWorks) throws Exception*

Fills the given lists with the necessary objects to run the MLP long time series update model and determines earliest time that needs to be used to start generating the historic speed records needed.

#### Parameters

<i>oWorkObjects</i>	List to fill with WorkObjects
<i>oRWorks</i>	List to fill with Works

#### Returns

time in milliseconds since Epoch to start generating historic speed records

#### Exceptions

<i>Exception</i>	
------------------	--

*void imrcp.forecast.mlp.MLPUpdate.execute ()*

If **m\_oQueue** is not empty, gets the first object from it and runs the long time series update MLP model.

Reimplemented from **imrcp.system.BaseBlock** (p.95).

*void imrcp.forecast.mlp.MLPUpdate.finishWork (long lTimestamp) [protected]*

If **m\_bRealTime** is false, notify the the block that will rerun the online speed prediction model the long\_ts files are ready. If there are more times to process, schedule it to run in a separate thread in 1 second, otherwise update the status to **IDLE**

#### Parameters

<i>lTimestamp</i>	run time of MLP model in milliseconds since Epoch
-------------------	---

Reimplemented from **imrcp.forecast.mlp.MLPBlock** (p.346).

*void imrcp.forecast.mlp.MLPUpdate.processWork (Work oRWork) [protected]*

Runs the MLP long time series update model for the segments in the given **Work** object by calling R code through the **org.rosuda.REngine.Rserve.RConnection** and Rserve interfaces.

#### Parameters

<i>oRWork</i>	Contains the data to run the model on
Reimplemented from <b>imrcp.forecast.mlp.MLPBlock</b> (p.349).	

**void imrcp.forecast.mlp.MLPUpdate.queue (String sDate, StringBuilder sBuffer)**  
**throws Exception**

Attempts to queue the time parsed from the given date string.

#### Parameters

<i>sDate</i>	date to queue in "yyyy-MM-dd" format
<i>sBuffer</i>	Buffer to append status messages to

#### Exceptions

<i>Exception</i>
------------------

**void imrcp.forecast.mlp.MLPUpdate.reset ()**

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.forecast.mlp.MLPBlock** (p.349).

**void imrcp.forecast.mlp.MLPUpdate.run ()**

Since most of the processing happens in other threads than the one that calls this method, changing the status of the block is handled differently than the base case. If **m\_bRealTime** is true the current week's Monday is added to the queue. Then if the block is **IDLE** (not processing another time in other threads) **execute ()** is called

Reimplemented from **imrcp.system.BaseBlock** (p.98).

**void imrcp.forecast.mlp.MLPUpdate.save (long lTimestamp) [protected]**

Does nothing since files are saved in a different part of the process for this block

#### Parameters

<i>lTimestamp</i>	MLP model run time in milliseconds since Epoch
Reimplemented from <b>imrcp.forecast.mlp.MLPBlock</b> (p.349).	

**boolean imrcp.forecast.mlp.MLPUpdate.start () throws Exception**

Determine the next Monday and sets a schedule to execute on a fixed interval starting on that Monday and then every Monday after that. Also schedules this block to execute once now to ensure the data files are available and up to date.

#### Returns

true if no Exceptions are thrown

#### Exceptions

<i>Exception</i>
Reimplemented from <b>imrcp.system.BaseBlock</b> (p.99).

**static void imrcp.forecast.mlp.MLPUpdate.updateMonthlyHistDat (long lEarliestTs, ArrayList< Work > oRWorks, long lEndTime, String sTimeZone, int[] nBb, String sFilenameFormat, String sDir) throws Exception [static]**

Creates the historic speed files needed for inputs to the R code.

#### Parameters

<i>lEarliestTs</i>	Time in milliseconds since Epoch to start generating records
<i>oRWorks</i>	List of <b>Work</b> to be processed
<i>lEndTime</i>	Time in milliseconds since Epoch to generate records to up
<i>sTimeZone</i>	TimeZone ID String used by <code>java.util.TimeZone#getTimeZone(java.lang.String)</code>
<i>nBb</i>	bounding box of the Network that is being processed. [min lon, min lat, max lon, max lat] all the lon/lats are in decimal degrees scaled to 7 decimal places
<i>sFilenameFormat</i>	Format String used to generate file names per thread
<i>sDir</i>	Directory where files are written to

#### Exceptions

<i>Exception</i>	
------------------	--

---

## Member Data Documentation

*int imrcp.forecast.mlp.MLPUpdate.m\_nWeeksBack*

Number of weeks of historic speed data needed

---

*The documentation for this class was generated from the following file:*

- forecast/mlp/MLPUpdate.java
- 

## imrcp.web.MonitorServlet Class Reference

### Public Member Functions

- void **init** (ServletConfig oSConfig) throws ServletException
- void **reset** ()
- boolean **start** () throws Exception
- void **execute** ()
- void **doGet** (HttpServletRequest iReq, HttpServletResponse iRep)

### Additional Inherited Members

---

#### Detailed Description

This servlet manages requests for system monitor information.

#### Author

Federal Highway Administration

---

## Member Function Documentation

`void imrcp.web.MonitorServlet.doGet (HttpServletRequest iReq, HttpServletResponse iRep)`

If the correct token is part of the request, the monitor file is added to the response.

### Parameters

<code>iReq</code>	object that contains the request the client has made of the servlet
<code>iRep</code>	object that contains the response the servlet sends to the client

`void imrcp.web.MonitorServlet.execute ()`

For each BaseBlock configured in `m_oTimeouts` it checks the status of the block and determines if the block is executing in a normal/expected fashion.

Reimplemented from **imrcp.system.BaseBlock** (p.95).

`void imrcp.web.MonitorServlet.init (ServletConfig oSConfig) throws ServletException`

Initializes the block. Wrapper for `setName (java.lang.String)`, `setLogger ()`, `setConfig ()`, `register ()`, and `startService ()`

### Parameters

<code>oSConfig</code>	object containing configuration parameters in Tomcat's
-----------------------	--

### Exceptions

<code>ServletException</code>	
<code>n</code>	

`void imrcp.web.MonitorServlet.reset ()`

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.system.BaseBlock** (p.98).

`boolean imrcp.web.MonitorServlet.start () throws Exception`

Writes the current time to the monitor file and ts a schedule to execute on a fixed interval.

### Returns

### Exceptions

<code>Exception</code>	
<code>n</code>	

Reimplemented from **imrcp.system.BaseBlock** (p.99).

---

*The documentation for this class was generated from the following file:*

- `web/MonitorServlet.java`
-

## imrcp.store.NcfEntryData Class Reference

### Public Member Functions

- **NcfEntryData** (int nObsTypeId, ProjectionImpl oProj, VariableDS oVar, Array oArray, double[] dHz, String sHzName, double[] dVrt, String sVrtName, double[] dTime, String sTimeName, int nContrib)
- void **setHzVrtDim** (int nHz, int nVrt)
- void  **setTimeDim** (int nTime)
- boolean **isFillValue** (double dVal)
- boolean **isMissing** (double dVal)
- boolean **isValidData** (double dVal)
- double **getValue** (int nHz, int nVrt)

### Public Attributes

- VariableDS **m\_oVar**
- Array **m\_oArray**
- Index **m\_oIndex**
- int **m\_nHzIndex**
- int **m\_nVrtIndex**
- int **m\_nTimeIndex**
- double[] **m\_dTime**

### Protected Attributes

- boolean **m\_bUseTimeIndex** = true

### Additional Inherited Members

---

#### Detailed Description

**EntryData** for files that are opened using the NetCDF library from NCAR.

#### See also

ucar.nc2.NetcdfFile::open(java.lang.String)

#### Author

Federal Highway Administration

---

#### Constructor & Destructor Documentation

**imrcp.store.NcfEntryData.NcfEntryData** (int nObsTypeId, ProjectionImpl oProj, VariableDS oVar, Array oArray, double[] dHz, String sHzName, double[] dVrt, String sVrtName, double[] dTime, String sTimeName, int nContrib)

Constructs a new **NcfEntryData** with the given parameters

#### Parameters

<i>nObsTypeId</i>	IMRCP observation type id provided by this <b>EntryData</b>
<i>oProj</i>	Project Coordinate System implementation
<i>oVar</i>	NetCDF Variable object
<i>oArray</i>	NetCDF Array object
<i>dHz</i>	values of the horizontal axis
<i>sHzName</i>	label of the horizontal axis

<i>dVrt</i>	values of the vertical axis
<i>sVrtName</i>	label of the vertical axis
<i>dTime</i>	values of the time axis
<i>sTimeName</i>	label of the time axis
<i>nContrib</i>	IMRCP contributor Id

---

## Member Function Documentation

*double imrcp.store.NcfEntryData.getValue (int nHz, int nVrt)*

Gets the value of the grid at the given x and y coordinates

### Parameters

<i>nHz</i>	x coordinate
<i>nVrt</i>	y coordinate

### Returns

value of the grid at the given x and y coordinates, if the coordinates are out of range,  
Double.NaN is returned

Reimplemented from **imrcp.store.EntryData** (p.202).

*boolean imrcp.store.NcfEntryData.isFillValue (double dVal)*

Wrapper for **ucar.nc2.dataset.VariableDS#isFillValue (double)**

### Parameters

<i>dVal</i>	value to test
-------------	---------------

### Returns

true if the value is the fill value defined by the file, otherwise false

*boolean imrcp.store.NcfEntryData.isInvalidData (double dVal)*

Wrapper for **ucar.nc2.dataset.VariableDS#isInvalidData (double)**

### Parameters

<i>dVal</i>	value to test
-------------	---------------

### Returns

true if the value is the invalid as defined by the file, otherwise false

*boolean imrcp.store.NcfEntryData.isMissing (double dVal)*

Wrapper for **ucar.nc2.dataset.VariableDS#isMissing (double)**

### Parameters

<i>dVal</i>	value to test
-------------	---------------

### Returns

true if the value is the missing value defined by the file, otherwise false

`void imrcp.store.NcfEntryData.setHrzVrtDim (int nHz, int nVrt)`

Sets the horizontal and vertical dimensions (axes) to the given values

*Parameters*

<code>nHz</code>	position to set the horizontal axis to
<code>nVrt</code>	position to set the vertical axis to

`void imrcp.store.NcfEntryData.setTimeDim (int nTime)`

If the time index is used, sets the time dimension (axis) to the given value.

*Parameters*

<code>nTime</code>	position to set the time axis to
--------------------	----------------------------------

Reimplemented from **imrcp.store.EntryData** (p.203).

---

## Member Data Documentation

`boolean imrcp.store.NcfEntryData.m_bUseTimeIndex = true [protected]`

Flag indicating if the time index needs to be set

`double [] imrcp.store.NcfEntryData.m_dTime`

Contains the values of the time axis

`int imrcp.store.NcfEntryData.m_nHzIndex`

Index used to access the horizontal axis

`int imrcp.store.NcfEntryData.m_nTimeIndex`

Index used to access the time axis

`int imrcp.store.NcfEntryData.m_nVrtIndex`

Index used to access the vertical axis

`Array imrcp.store.NcfEntryData.m_oArray`

Array object from the NetCDF library

`Index imrcp.store.NcfEntryData.m_oIndex`

Index object from the NetCDF library

`VariableDS imrcp.store.NcfEntryData.m_oVar`

Variable object from the NetCDF library

---

*The documentation for this class was generated from the following file:*

- store/NcfEntryData.java
-

## imrcp.web.tiles.NcfTileCache Class Reference

### Public Member Functions

- void **reset** ()

### Protected Member Functions

- **GriddedFileWrapper getDataWrapper** (int nFormatIndex)

### Protected Attributes

- int[] **m\_nObsTypes**
- String[] **m\_sObsTypes**
- String[] **m\_sHz**
- String[] **m\_sVrt**
- String[] **m\_sTime**
- boolean[] **m\_bUseUVWind**

### Additional Inherited Members

---

### Detailed Description

#### Author

Federal Highway Administration

---

### Member Function Documentation

**GriddedFileWrapper imrcp.web.tiles.NcfTileCache.getDataWrapper (int nFormatIndex) [protected]**

#### Returns

If **m\_bUseUVWind** is not null and for the given format index is true, a new **RapNcfWrapper** otherwise a new **NcfWrapper** with the configured values

Reimplemented from **imrcp.web.tiles.TileCache** (p.521).

**void imrcp.web.tiles.NcfTileCache.reset ()**

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.web.tiles.TileCache** (p.522).

---

### Member Data Documentation

**boolean [] imrcp.web.tiles.NcfTileCache.m\_bUseUVWind [protected]**

Array that contains flags indicated if wind speeds are calculated from u and v component vectors corresponding to the different FilenameFormatters in **m\_oFormatters**

**int [] imrcp.web.tiles.NcfTileCache.m\_nObsTypes [protected]**

Contains the observation type ids that the file provides.

***String [] imrcp.web.tiles.NcfTileCache.m\_sHz [protected]***

Array that contains the horizontal labels in the files corresponding to the different  
FilenameFormatters in **m\_oFormatters**

***String [] imrcp.web.tiles.NcfTileCache.m\_sObsTypes [protected]***

Contains the observation type labels that correspond to the observation type id in  
**m\_nObsTypes**

***String [] imrcp.web.tiles.NcfTileCache.m\_sTime [protected]***

Array that contains the time labels in the files corresponding to the different  
FilenameFormatters in **m\_oFormatters**

***String [] imrcp.web.tiles.NcfTileCache.m\_sVrt [protected]***

Array that contains the vertical labels in the files corresponding to the different  
FilenameFormatters in **m\_oFormatters**

---

*The documentation for this class was generated from the following file:*

- web/tiles/NcfTileCache.java
- 

## imrcp.store.NcfWrapper Class Reference

### Public Member Functions

- **NcfWrapper** (int[] nObsTypes, String[] sObsTypes, String sHz, String sVrt, String sTime)
- int **getTimeIndex** (**EntryData** oData, long lTimestamp)
- void **load** (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId)  
throws Exception
- void **cleanup** (boolean bDelete)
- ArrayList< **Obs** > **getData** (int nObsTypeId, long lTimestamp, int nLat1, int nLon1, int nLat2, int nLon2)
- double **getReading** (int nObsTypeId, long lTimestamp, int nLat, int nLon, Date oTimeRecv)
- String **toString** ()
- void **getIndices** (int nLon, int nLat, int[] nIndices)
- double **getReading** (int nObsType, long lTimestamp, int[] nIndices)

### Public Attributes

- NetcdfFile **m\_oNcFile**

### Static Protected Member Functions

- static double[] **fromArray** (NetcdfFile oNcFile, String sName) throws Exception

### Protected Attributes

- String[] **m\_sObsTypes**
- String **m\_sHz**
- String **m\_sVrt**
- String **m\_sTime**

### Additional Inherited Members

---

## Detailed Description

Parses and creates **Obs** from .grb2 received from different National Weather Service products. UCAR's NetCDF library is used to load the files into memory

### Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

*imrcp.store.NcfWrapper.NcfWrapper (int[] nObsTypes, String[] sObsTypes, String sHz, String sVrt, String sTime)*

Constructs a new **NcfWrapper** with the given parameters

#### Parameters

<i>nObsTypes</i>	IMRCP observation types provided in the file
<i>sObsTypes</i>	Label of the observation types found in the file
<i>sHz</i>	horizontal axis label
<i>sVrt</i>	vertical axis label
<i>sTime</i>	time axis label

---

## Member Function Documentation

*void imrcp.store.NcfWrapper.cleanup (boolean bDelete)*

If the delete flag is true, deletes the index files created by the NetCDF library.

Reimplemented from **imrcp.store.FileWrapper** (p.223).

*static double[] imrcp.store.NcfWrapper.fromArray (NetcdfFile oNcFile, String sName) throws Exception [static], [protected]*

Creates a double[] that contains the values stored in the **ucar.ma2.Array** in the given NetcdfFile with the given label/name.

#### Parameters

<i>oNcFile</i>	File to get the Array from
<i>sName</i>	label of the Array

#### Returns

double[] with the values store in the Array in the NetcdfFile with the given name.

#### Exceptions

<i>Exception</i>	
------------------	--

*ArrayList<Obs> imrcp.store.NcfWrapper.getData (int nObsTypId, long lTimestamp, int nLat1, int nLon1, int nLat2, int nLon2)*

Iterates gridded data that corresponds to the bounding box defined by the given longitudes and latitudes and create **Obs** for the values that match the query.

#### Parameters

<i>nObsTypeId</i>	IMRCP observation type
<i>lTimestamp</i>	query time in milliseconds since Epoch
<i>nLat1</i>	minimum latitude of the query in decimal degrees scaled to 7 decimal places
<i>nLon1</i>	minimum longitude of the query in decimal degrees scaled to 7 decimal places
<i>nLat2</i>	maximum latitude of the query in decimal degrees scaled to 7 decimal places
<i>nLon2</i>	maximum longitude of the query in decimal degrees scaled to 7 decimal places

#### Returns

ArrayList filled with **Obs** that match the query

Reimplemented in **imrcp.store.NDFDQpfWrapper** (p.378), and **imrcp.store.RapNcfWrapper** (p.469).

**void imrcp.store.NcfWrapper.getIndices (int nLon, int nLat, int[] nIndices)**

Fills the given int array with the x and y coordinates/indices of the grid that correspond to the given longitude and latitude.

#### Parameters

<i>nLon</i>	longitude in decimal degrees scaled to 7 decimal places
<i>nLat</i>	latitude in decimal degrees scaled to 7 decimal places
<i>nIndices</i>	array to fill with x and y coordinates of the grid that correspond to the longitude and latitude [x,y]

Reimplemented from **imrcp.store.GriddedFileWrapper** (p.258).

**double imrcp.store.NcfWrapper.getReading (int nObsType, long lTimestamp, int[] nIndices)**

Get the value of the cell of the grid for the given observation type at the given indices/coordinates

#### Parameters

<i>nObsType</i>	desired observation type
<i>lTimestamp</i>	query time in milliseconds since Epoch
<i>nIndices</i>	[x coordinate, y coordinate]

#### Returns

Reimplemented from **imrcp.store.GriddedFileWrapper** (p.259).

Reimplemented in **imrcp.store.RapNcfWrapper** (p.469).

**double imrcp.store.NcfWrapper.getReading (int nObsType, long lTimestamp, int nLat, int nLon, Date oTimeRecv)**

Gets the value of the cell of the grid for the given observation type at the given lon/lat and time.

*Parameters*

<i>nObsType</i>	desired observation type
<i>lTimestamp</i>	query time in milliseconds since Epoch
<i>nLat</i>	latitude of query in decimal degrees scaled to 7 decimal places
<i>nLon</i>	longitude of query in decimal degrees scaled to 7 decimal places
<i>oTimeRecv</i>	Date object to have its time set to the time the observation was received for some implementation

*Returns*

the value of the cell of the grid for the given observation type at the given lon/lat and time,  
Double.NaN if a valid value does not exists for the query

Reimplemented from **imrcp.store.GriddedFileWrapper** (p.258).

Reimplemented in **imrcp.store.RapNcfWrapper** (p.469).

*int imrcp.store.NcfWrapper.getTimeIndex (EntryData oData, long lTimestamp)*

Gets the time index to use for the given **EntryData** and query time.

*Parameters*

<i>oData</i>	<b>EntryData</b> being queried
<i>lTimestamp</i>	query time in milliseconds since Epoch

*Returns*

index corresponding to the query time for the time dimension or -1 if the time is outside of the **EntryData**'s range

Reimplemented from **imrcp.store.GriddedFileWrapper** (p.259).

*void imrcp.store.NcfWrapper.load (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId) throws Exception*

Loads the gridded data of the file into memory using the UCAR NetCDF library.

Reimplemented from **imrcp.store.FileWrapper** (p.224).

*String imrcp.store.NcfWrapper.toString ()*

*Returns*

The value of **m\_oNcFile** calling **ucar.nc2.NetcdfFile#getLocation()** which should be the path of the file.

---

## Member Data Documentation

*NetcdfFile imrcp.store.NcfWrapper.m\_oNcFile*

Original file loaded by the NetCDF library

*String imrcp.store.NcfWrapper.m\_sHz [protected]*

Horizontal axis label in the file

*String [] imrcp.store.NcfWrapper.m\_sObsTypes [protected]*

Contains the observation type labels that correspond to the observation type id in **m\_nObsTypes**

*String imrcp.store.NcfWrapper.m\_sTime [protected]*

Time axis label in the file

*String imrcp.store.NcfWrapper.m\_sVrt [protected]*

Vertical axis label in the file

---

*The documentation for this class was generated from the following file:*

- store/NcfWrapper.java
- 

## imrcp.store.NDFDQpfStore Class Reference

### Public Member Functions

- **NDFDQpfStore ()**
- **GriddedFileWrapper getNewFileWrapper ()**

### Additional Inherited Members

---

#### Detailed Description

**FileCache** that manages National Digital Forecast Database (NDFD) Quantitative Precipitation Forecast files (QPF).

#### Author

Federal Highway Administration

---

#### Constructor & Destructor Documentation

*imrcp.store.NDFDQpfStore.NDFDQpfStore ()*

Default constructor. Does nothing.

---

#### Member Function Documentation

*GriddedFileWrapper imrcp.store.NDFDQpfStore.getNewFileWrapper ()*

#### Returns

a new **NDFDQpfWrapper** with the configured parameters

Reimplemented from **imrcp.store.WeatherStore** (p.564).

---

*The documentation for this class was generated from the following file:*

- store/NDFDQpfStore.java
-

## imrcp.store.NDFDQpfWrapper Class Reference

### Public Member Functions

- **NDFDQpfWrapper** (int[] nObsTypes, String[] sObsTypes, String sHrz, String sVrt, String sTime)
- ArrayList< Obs > **getData** (int nObsTypeId, long lTimestamp, int nLat1, int nLon1, int nLat2, int nLon2)

### Additional Inherited Members

---

### Detailed Description

Parses and creates **Obs** for National Digital Forecast Database (NDFD) Quantitative Precipitation Forecast (QPF) files.

#### Author

Federal Highway Administration

---

### Constructor & Destructor Documentation

*imrcp.store.NDFDQpfWrapper.NDFDQpfWrapper (int[] nObsTypes, String[] sObsTypes, String sHrz, String sVrt, String sTime)*

Wrapper for **NcfWrapper#NcfWrapper(int[], java.lang.String[], java.lang.String, java.lang.String, java.lang.String)**

#### Parameters

<i>nObsTypes</i>	IMRCP observation types the file provides
<i>sObsTypes</i>	Label that corresponds to the observation types in nObsTypes found in the file
<i>sHrz</i>	label of the x axis
<i>sVrt</i>	label of the y axis
<i>sTime</i>	label of the time axis

---

### Member Function Documentation

*ArrayList< Obs > imrcp.store.NDFDQpfWrapper.getData (int nObsTypeld, long lTimestamp, int nLat1, int nLon1, int nLat2, int nLon2)*

Converts the 6 hour predictions in to 6 1 hour predictions for precipitation rate.

Reimplemented from **imrcp.store.NcfWrapper** (p.374).

---

*The documentation for this class was generated from the following file:*

- store/NDFDQpfWrapper.java
-

## imrcp.geosrv.Network Class Reference

### Public Member Functions

- **Network** (JSONObject oFeature, ArrayList< OsmWay > oWays, ArrayList< OsmNode > oNodes)
- JSONObject **toGeoJsonFeature** ()
- int[] **getBoundingBox** ()
- Iterator< int[]> **getPointIterator** ()
- int **getCoordinateCount** ()
- boolean **obsInside** (Obs oObs)
- boolean **pointInside** (int nLon, int nLat)
- boolean **wayInside** (OsmWay oWay)
- boolean **includeId** (Id oId)
- int **compareTo** (Network o)

### Public Attributes

- String **m\_sLabel**
- int **m\_nLoaded**
- boolean **m\_bFinalized**
- String **m\_sNetworkId**
- String[] **m\_sFilter**
- String[] **m\_sOptions**
- ArrayList< Id > **m\_oSegmentIds**
- String[] **m\_sStates**

---

### Detailed Description

This class represents the roadway networks used by the system. They are created by using the web UI that interface with **imrcp.web.NetworkGeneration**.

#### Author

Federal Highway Administration

---

### Constructor & Destructor Documentation

*imrcp.geosrv.Network.Network (JSONObject oFeature, ArrayList< OsmWay > oWays, ArrayList< OsmNode > oNodes)*

Constructs a **Network** from a JSONObject created from the Networks JSON file and the list of Ways and Nodes contained in the **Network**

#### Parameters

<i>oFeature</i>	JSON representation of the <b>Network</b>
<i>oWays</i>	a list of the roadway segments in the <b>Network</b>
<i>oNodes</i>	a list of the nodes that make up all of the roadway segments int the <b>Network</b>

---

### Member Function Documentation

*int imrcp.geosrv.Network.compareTo (Network o)*

Compares **Network** objects by network id

*int[] imrcp.geosrv.Network.getBoundingBox ()*

Gets a new int array that contains the bounding box of the polygon

*Returns*

a new array with format [min lon, min lat, max lon, max lat]

*int imrcp.geosrv.Network.getCoordinateCount ()*

Returns the number of coordinates (x,y pairs times 2) of the polygon.

*Returns*

The number of coordinates of the polygon

*Iterator< int[]> imrcp.geosrv.Network.getPointIterator ()*

Gets an Iterator for the points of the polygon. Wrapper for **imrcp.system.Arrays#iterator(int[], int[], int, int)**

*Returns*

Iterator for the points of the polygon

*boolean imrcp.geosrv.Network.includeld (Id old)*

Determines if the given Id is included in the **Network**. Any Id that does not represent a roadway segment is by default included.

*Parameters*

<i>oId</i>	Id to test
------------	------------

*Returns*

true if the Id does not represent a roadway segment or if the Id represents a roadway segment and is in **m\_oSegmentIds**, otherwise false

*boolean imrcp.geosrv.Network.obsInside (Obs oObs)*

Wrapper for **GeoUtil#obsInside(java.awt.geom.Area, imrcp.store.Obs)** passing **m\_oArea** and the given Obs

*Parameters*

<i>oObs</i>	Obs to test if it is inside the Area defining the polygon
-------------	---

*Returns*

true if the Obs is inside the Area, otherwise false

*boolean imrcp.geosrv.Network.pointInside (int nLon, int nLat)*

Determines if the given longitude/latitude coordinates are inside the **Network** polygon. Wrapper for **GeoUtil#isInside(int, int, int, int, int, int, int)** and **GeoUtil#isInsidePolygon(int[], double, double, int)**

*Parameters*

<i>nLon</i>	longitude in decimal degrees scaled to 7 decimal places
<i>nLat</i>	latitude in decimal degrees scaled to 7 decimal places

*Returns*

true if the point is inside the **Network** polygon, otherwise false

*JSONObject imrcp.geosrv.Network.toGeoJsonFeature ()*

Creates a new GeoJson Feature object representation of the **Network**

*Returns*

GeoJson Feature object representing the **Network**

*boolean imrcp.geosrv.Network.wayInside (OsmWay oWay)*

Determines if the given roadway segment is a part of and inside the **Network**.

*Parameters*

<i>oWay</i>	roadway segment to test
-------------	-------------------------

*Returns*

true if the OsmWay's Id is in **m\_oSegmentIds** and at least one of its Nodes is inside the **Network** polygon, otherwise false

---

## Member Data Documentation

*boolean imrcp.geosrv.Network.m\_bFinalized*

Flag indicating if the network has been finalized or not

*int imrcp.geosrv.Network.m\_nLoaded*

Flag indicating the status of loading and processing the segments. 0 - finished loading all segments 1 - in the process of loading segments 2 - an error occurred when loading segments

*ArrayList<Id> imrcp.geosrv.Network.m\_oSegmentIds*

List of the roadway segment Ids included in the network

*String [] imrcp.geosrv.Network.m\_sFilter*

Road classification filters used to create the network. Valid values come from the highway tag of OSM definitions, including: motorway, motorway\_link, trunk, trunk\_link, primary, primary\_link, secondary, secondary\_link, tertiary, tertiary\_link, residential, unclassified

*String imrcp.geosrv.Network.m\_sLabel*

User given label of the network

*String imrcp.geosrv.Network.m\_sNetworkId*

A Base64 encoded String of a 16 byte id generated by the coordinates of the geometry of the network.

*String [] imrcp.geosrv.Network.m\_sOptions*

Road options used to create the network. Valid values are motorway, connector, trunk, ramp, primary, primary\_link, secondary, secondary\_link, tertiary, and tertiary\_link

*String [] imrcp.geosrv.Network.m\_sStates*

The United States the bounding polygon intersects. This is needed because the system downloads OSM definitions by state.

---

*The documentation for this class was generated from the following file:*

- geosrv/Network.java
- 

## [imrcp.web.NetworkGeneration Class Reference](#)

### Classes

- class **EditList**

### Public Member Functions

- boolean **start** () throws Exception
- void **reset** ()
- int **doList** (HttpServletRequest oReq, HttpServletResponse oRes, **Session** oSession) throws ServletException, IOException
- int **doHash** (HttpServletRequest oReq, HttpServletResponse oRes, **Session** oSession) throws ServletException
- int **doDelete** (HttpServletRequest oReq, HttpServletResponse oRes, **Session** oSession) throws IOException, ServletException
- int **doReprocess** (HttpServletRequest oReq, HttpServletResponse oRes, **Session** oSession) throws IOException, ServletException
- int **doCreate** (HttpServletRequest oReq, HttpServletResponse oRes, **Session** oSession) throws IOException, ServletException
- void **execute** ()
- int **doSave** (HttpServletRequest oReq, HttpServletResponse oRes, **Session** oSession) throws IOException, ServletException
- int **doGeo** (HttpServletRequest oReq, HttpServletResponse oRes, **Session** oSession) throws IOException, ServletException
- int **doAdd** (HttpServletRequest oReq, HttpServletResponse oRes, **Session** oSession) throws IOException, ServletException
- int **doRemove** (HttpServletRequest oReq, HttpServletResponse oRes, **Session** oSession) throws IOException, ServletException
- int **doMerge** (HttpServletRequest oReq, HttpServletResponse oRes, **Session** oSession) throws IOException, ServletException
- int **doSplit** (HttpServletRequest oReq, HttpServletResponse oRes, **Session** oSession) throws IOException, ServletException
- int **doDetectors** (HttpServletRequest oReq, HttpServletResponse oRes, **Session** oSession) throws IOException, ServletException
- int **doDetsave** (HttpServletRequest oReq, HttpServletResponse oRes, **Session** oSession) throws IOException, ServletException
- int **doUpload** (HttpServletRequest oReq, HttpServletResponse oRes, **Session** oSession) throws IOException, ServletException
- int **doOsm** (HttpServletRequest oReq, HttpServletResponse oRes, **Session** oSession) throws IOException, ServletException
- int **doFinalize** (HttpServletRequest oReq, HttpServletResponse oRes, **Session** oSession) throws IOException, ServletException
- void **removeList** (EditList oList)
- EditList **getList** (String sSessionId, String sNetworkId)

### Additional Inherited Members

---

## Detailed Description

This servlet manages and processes all the necessary data and requests for the IMRCP Road Network Generation tool.

### Author

Federal Highway Administration

---

## Member Function Documentation

*int imrcp.web.NetworkGeneration.doAdd (HttpServletRequest oReq,  
HttpServletResponse oRes, Session oSession) throws IOException, ServletException*

Adds the requested Id to the add list of the EditList for the requested network and session.

### Parameters

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client
<i>oSession</i>	object that contains information about the user that made the request

### Returns

HTTP status code to be included in the response.

### Exceptions

<i>IOException</i>	
<i>ServletException</i>	
<i>n</i>	

*int imrcp.web.NetworkGeneration.doCreate (HttpServletRequest oReq,  
HttpServletResponse oRes, Session oSession) throws IOException, ServletException*

Creates a new roadway network with the requested parameters and queues it to process.

### Parameters

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client
<i>oSession</i>	object that contains information about the user that made the request

### Returns

HTTP status code to be included in the response.

### Exceptions

<i>IOException</i>	
<i>ServletException</i>	
<i>n</i>	

*int imrcp.web.NetworkGeneration.doDelete (HttpServletRequest oReq, HttpServletResponse oRes, Session oSession) throws IOException, ServletException*

Deletes the requested network, removing it from the system. If the network is currently processing then **m\_bSave** is set to false.

#### Parameters

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client
<i>oSession</i>	object that contains information about the user that made the request

#### Returns

HTTP status code to be included in the response.

#### Exceptions

<i>IOException</i>	
<i>ServletException</i>	
<i>n</i>	

*int imrcp.web.NetworkGeneration.doDetectors (HttpServletRequest oReq, HttpServletResponse oRes, Session oSession) throws IOException, ServletException*

If a detector file exists for the requested network, adds the detectors to the response.

#### Parameters

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client
<i>oSession</i>	object that contains information about the user that made the request

#### Returns

HTTP status code to be included in the response.

#### Exceptions

<i>IOException</i>	
<i>ServletException</i>	
<i>n</i>	

*int imrcp.web.NetworkGeneration.doDetsave (HttpServletRequest oReq, HttpServletResponse oRes, Session oSession) throws IOException, ServletException*

Saves the way mapping for the requested detector.

#### Parameters

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client
<i>oSession</i>	object that contains information about the user that made the request

*Returns*

HTTP status code to be included in the response.

*Exceptions*

<i>IOException</i>	
<i>ServletException</i>	
<i>n</i>	

*int imrcp.web.NetworkGeneration.doFinalize (HttpServletRequest oReq,  
HttpServletResponse oRes, Session oSession) throws IOException, ServletException*

Queues the requested network to be finalized.

*Parameters*

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client
<i>oSession</i>	object that contains information about the user that made the request

*Returns*

HTTP status code to be included in the response.

*Exceptions*

<i>IOException</i>	
<i>ServletException</i>	
<i>n</i>	

*int imrcp.web.NetworkGeneration.doGeo (HttpServletRequest oReq,  
HttpServletResponse oRes, Session oSession) throws IOException, ServletException*

Adds the geometry of all the roadway segments of the requested network to the response.

*Parameters*

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client
<i>oSession</i>	object that contains information about the user that made the request

*Returns*

HTTP status code to be included in the response.

*Exceptions*

<i>IOException</i>	
<i>ServletException</i>	
<i>n</i>	

*int imrcp.web.NetworkGeneration.doHash (HttpServletRequest oReq,  
HttpServletResponse oRes, Session oSession) throws ServletException*

Adds roadway segments in the cell of the requested hash index to the response.

*Parameters*

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client
<i>oSession</i>	object that contains information about the user that made the request

*Returns*

HTTP status code to be included in the response.

*Exceptions*

<i>ServletException</i>	
<i>n</i>	

*int imrcp.web.NetworkGeneration.doList (HttpServletRequest oReq,  
HttpServletResponse oRes, Session oSession) throws ServletException, IOException*

Adds a list of all of the networks in the system to the response.

*Parameters*

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client
<i>oSession</i>	object that contains information about the user that made the request

*Returns*

HTTP status code to be included in the response.

*Exceptions*

<i>ServletException</i>	
<i>n</i>	

<i>IOException</i>	
--------------------	--

*int imrcp.web.NetworkGeneration.doMerge (HttpServletRequest oReq,  
HttpServletResponse oRes, Session oSession) throws IOException, ServletException*

Attempts to merge the roadway segments with the requested ids. If the merge is successful, the geometry of the newly merged roadway segment is added to the response.

*Parameters*

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client
<i>oSession</i>	object that contains information about the user that made the request

*Returns*

HTTP status code to be included in the response.

*Exceptions*

<i>IOException</i>	
--------------------	--

<i>ServletExceptio n</i>	
------------------------------	--

*int imrcp.web.NetworkGeneration.doOsm (HttpServletRequest oReq,  
HttpServletResponse oRes, Session oSession) throws IOException, ServletException*

Adds an OSM .xml.bz2 representation of the requested network to the response.

#### Parameters

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client
<i>oSession</i>	object that contains information about the user that made the request

#### Returns

HTTP status code to be included in the response.

#### Exceptions

<i>IOException</i>	
<i>ServletExceptio n</i>	
<i>n</i>	

*int imrcp.web.NetworkGeneration.doRemove (HttpServletRequest oReq,  
HttpServletResponse oRes, Session oSession) throws IOException, ServletException*

Adds the requested Id to the remove list of the EditList for the requested network and session.

#### Parameters

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client
<i>oSession</i>	object that contains information about the user that made the request

#### Returns

HTTP status code to be included in the response.

#### Exceptions

<i>IOException</i>	
<i>ServletExceptio n</i>	
<i>n</i>	

*int imrcp.web.NetworkGeneration.doReprocess (HttpServletRequest oReq,  
HttpServletResponse oRes, Session oSession) throws IOException, ServletException*

Queues the requested network to be reprocessed. The network's label is over written by the requested label.

#### Parameters

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client

<i>oSession</i>	object that contains information about the user that made the request
-----------------	---

*Returns*

HTTP status code to be included in the response.

*Exceptions*

<i>IOException</i>	
<i>ServletException</i>	
<i>n</i>	

*int imrcp.web.NetworkGeneration.doSave (HttpServletRequest oReq,  
HttpServletResponse oRes, Session oSession) throws IOException, ServletException*

Saves any edits that have been performed for the requested network and session.

*Parameters*

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client
<i>oSession</i>	object that contains information about the user that made the request

*Returns*

HTTP status code to be included in the response.

*Exceptions*

<i>IOException</i>	
<i>ServletException</i>	
<i>n</i>	

*int imrcp.web.NetworkGeneration.doSplit (HttpServletRequest oReq,  
HttpServletResponse oRes, Session oSession) throws IOException, ServletException*

Splits the requested roadway segment at the index specified in the request. The geometry of the subsequent ways is added to the response.

*Parameters*

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client
<i>oSession</i>	object that contains information about the user that made the request

*Returns*

HTTP status code to be included in the response.

*Exceptions*

<i>IOException</i>	
<i>ServletException</i>	
<i>n</i>	

`int imrcp.web.NetworkGeneration.doUpload (HttpServletRequest oReq, HttpServletResponse oRes, Session oSession) throws IOException, ServletException`

Saves the detector file specified in the request and creates the mapping file for the detectors.

*Parameters*

<code>oReq</code>	object that contains the request the client has made of the servlet
<code>oRes</code>	object that contains the response the servlet sends to the client
<code>oSession</code>	object that contains information about the user that made the request

*Returns*

HTTP status code to be included in the response.

*Exceptions*

<code>IOException</code>	
<code>ServletException</code>	
<code>n</code>	

`void imrcp.web.NetworkGeneration.execute ()`

Processes the first network in the process queue. This includes determining which US states the network intersects, downloading any necessary OSM data files, and converting OSM xml.bz2 files into IMRCP OSM binary files.

Reimplemented from **imrcp.system.BaseBlock** (*p.95*).

`EditList imrcp.web.NetworkGeneration.getList (String sSessionId, String sNetworkId)`

Gets the active EditList for the given session id and network id

*Parameters*

<code>sSessionId</code>	session token for the user editing the network
<code>sNetworkId</code>	network id

*Returns*

active EditList associated with the session id and network id.

`void imrcp.web.NetworkGeneration.removeList (EditList oList)`

Removes the given list from the active session list.

*Parameters*

<code>oList</code>	edit list to remove
--------------------	---------------------

`void imrcp.web.NetworkGeneration.reset ()`

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.web.SecureBaseBlock** (*p.492*).

`boolean imrcp.web.NetworkGeneration.start () throws Exception`

Sets a schedule to execute on a fixed interval.

*Returns*

true if no Exceptions are thrown

### *Exceptions*

<i>Exception</i>	
Reimplemented from <b>imrcp.system.BaseBlock</b> (p.99).	

*The documentation for this class was generated from the following file:*

- web/NetworkGeneration.java

## imrcp.collect.NHCKmz Class Reference

### Public Member Functions

- void **reset** ()
- boolean **start** () throws Exception
- void **execute** ()
- void **checkYear** (int nYear, boolean bNotify) throws Exception

### Static Public Member Functions

- static String **getStormNumber** (String sFile, String[] sParts)

### Additional Inherited Members

### Detailed Description

**Collector** used to download hurricane forecasts from the National Hurricane Center.

### Author

Federal Highway Administration

## Member Function Documentation

### *void imrcp.collect.NHCKmz.checkYear (int nYear, boolean bNotify) throws Exception*

Check for and downloads any hurricane forecast in the NHC archive that is not stored on disk for the given year.

#### Parameters

<i>nYear</i>	Year to check
<i>bNotify</i>	when true, notifies subscribing blocks of new files

### *Exceptions*

<i>Exception</i>	
------------------	--

### *void imrcp.collect.NHCKmz.execute ()*

Wrapper for **NHCKmz#checkYear (int, boolean)** . Calls it for the current year.

Reimplemented from **imrcp.system.BaseBlock** (p.95).

`static String imrcp.collect.NHCKmz.getStormNumber (String sFile, String[] sParts) [static]`

Parses the file name of the .zip files create by **NHCKmz** to extract metadata about the storm forecast the file contains

*Parameters*

<code>sFile</code>	IMRCP NHC .zip file to parse
<code>sParts</code>	Gets filled with the 2 parts of the storm metadata string defined below in the format [BB##yyyy, %% % ?adv]

*Returns*

The string that represents the storm metadata in the format BB##yyyy\_%% % ?adv where BB is the 2 char basin abbreviation, ## is the 2 digit storm number for that basin, yyyy is the 4 digit year, %% % is the 3 digit (with leading 0s) forecast number for that storm, ? is an optional 1 char used for intermediate forecasts (starts at A and increments in alphabetical order) and adv is the constant NHC use at the end of storm numbers. An example is if sFile = "2022/nhc\_EP032022\_019adv\_202206210900\_202206261800\_202206210844.zip" EP032022\_019adv is returned and sParts = [EP032022, 019adv]

`void imrcp.collect.NHCKmz.reset ()`

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.system.BaseBlock** (p.98).

`boolean imrcp.collect.NHCKmz.start () throws Exception`

Calls **NHCKmz#checkYear(int, boolean)** starting at **NHCKmz#m\_nStartYear** and going until the current year. Then sets a schedule to execute on a fixed interval.

*Returns*

true if no exceptions are thrown

*Exceptions*

<code>Exception</code>
Reimplemented from <b>imrcp.system.BaseBlock</b> (p.99).

---

*The documentation for this class was generated from the following file:*

- collect/NHCKmz.java
- 

## imrcp.store.NHCStore Class Reference

### Public Member Functions

- `ArrayList<NHCWrapper> getFiles (long lTimestamp, long lRefTime)`
- `boolean loadFileToMemory (String sFullPath, int nFormatIndex)`

### Protected Member Functions

- `FileWrapper getNewFileWrapper ()`
- `void getData (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)`

## Additional Inherited Members

---

### Detailed Description

**FileCache** that manages hurricane forecast zipped .kmz files from the National Hurricane Center

### Author

Federal Highway Administration

---

### Member Function Documentation

**void imrcp.store.NHCStore.getData (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime) [protected]**

Determines the files valid for the given time query and then calls **NHCWrapper#getData(int, long, long, int, int, int, int)** for each of those files.

Reimplemented from **imrcp.system.BaseBlock** (p.96).

**ArrayList< NHCWrapper > imrcp.store.NHCStore.getFiles (long lTimestamp, long lRefTime)**

Gets a list of hurricane forecast files that match the given time query

#### Parameters

<i>lTimestamp</i>	query time in milliseconds since Epoch
<i>lRefTime</i>	reference time in milliseconds since Epoch

#### Returns

a list containing all the hurricane forecast files valid for the given time query

**FileWrapper imrcp.store.NHCStore.getNewFileWrapper () [protected]**

#### Returns

a new **NHCWrapper**

Reimplemented from **imrcp.store.FileCache** (p.212).

**boolean imrcp.store.NHCStore.loadFileToMemory (String sFullPath, int nFormatIndex)**

Attempts to load the given file path into memory and places it in the cache

#### Parameters

<i>sFullPath</i>	File path to load
<i>nFormatIndex</i>	index of <b>m_oFormatters</b> to use

#### Returns

true if the file is successfully loaded into the cache

Reimplemented from **imrcp.store.FileCache** (p.213).

---

*The documentation for this class was generated from the following file:*

- store/NHCStore.java
- 

## imrcp.web.tiles.NHCTiles Class Reference

### Public Member Functions

- void **init** (ServletConfig oSConfig)
- boolean **start** () throws Exception

### Protected Member Functions

- void **doGet** (HttpServletRequest oRequest, HttpServletResponse oResponse) throws ServletException, IOException

### Additional Inherited Members

---

### Detailed Description

This class handles requests for and generates VectorTiles for the hurricane forecasts received from the National Hurricane Center.

#### Author

Federal Highway Administration

---

### Member Function Documentation

**void imrcp.web.tiles.NHCTiles.doGet (HttpServletRequest oRequest, HttpServletResponse oResponse) throws ServletException, IOException [protected]**

Creates and sends VectorTiles for hurricane forecasts that match the request as the response.

#### Parameters

<i>oRequest</i>	object that contains the request the client has made of the servlet
<i>oResponse</i>	object that contains the response the servlet sends to the client

#### Exceptions

<i>ServletException</i>	
<i>n</i>	

<i>IOException</i>
--------------------

**void imrcp.web.tiles.NHCTiles.init (ServletConfig oSConfig)**

Initializes the block.Wrapper for **setName (java.lang.String)**, **setLogger ()**, **setConfig ()**, **register ()**, and **startService ()**

*Parameters*

<i>oSConfig</i>	object containing configuration parameters in Tomcat's web.xml
-----------------	--

*boolean imrcp.web.tiles.NHCTiles.start () throws Exception*

Sets the reference for **m\_oNHC**

*Returns*

true if no Exceptions are thrown

Reimplemented from **imrcp.system.BaseBlock** (p.99).

---

*The documentation for this class was generated from the following file:*

- web/tiles/NHCTiles.java
- 

## imrcp.store.NHCWrapper Class Reference

### Classes

- class **ConeParser**
- class **TrackParser**

### Public Member Functions

- void **load** (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId) throws Exception
- ArrayList< **Obs** > **getData** (int nObsTypeId, long lTimestamp, long lRefTime, int nLat1, int nLon1, int nLat2, int nLon2)
- void **cleanup** (boolean bDelete)

### Static Public Member Functions

- static int **getSSHWS** (double dWindSpeed)

### Public Attributes

- ArrayList< **Obs** > **m\_oObs** = new ArrayList()
- int[] **m\_oPolygon** = Arrays.newIntArray()
- ArrayList< int[] > **m\_oTracks** = new ArrayList()
- int[] **m\_nBB** = new int[]{Integer.MAX\_VALUE, Integer.MAX\_VALUE, Integer.MIN\_VALUE, Integer.MIN\_VALUE}
- String **m\_sStormName** = null
- int **m\_nMinPressure** = Integer.MIN\_VALUE
- int[] **m\_nMaxWindSpeeds** = Arrays.newIntArray()
- ArrayList< **HurricaneCenter** > **m\_oCenters** = new ArrayList()

### Static Public Attributes

- static final HashMap< String, Integer > **STORMTYPES** = new HashMap()

### Additional Inherited Members

---

### Detailed Description

Parses and creates **Obs** from zipped .kmz files received from the National Hurricane Center

## Author

Federal Highway Administration

---

## Member Function Documentation

`void imrcp.store.NHCWrapper.cleanup (boolean bDelete)`

Does nothing for this implementing class.

### Parameters

<code>bDelete</code>	not used
----------------------	----------

Reimplemented from **imrcp.store.FileWrapper** (*p.223*).

`ArrayList< Obs > imrcp.store.NHCWrapper.getData (int nObsTypeId, long lTimestamp, long lRefTime, int nLat1, int nLon1, int nLat2, int nLon2)`

Creates a new list and fills it will **Obs** that match the given query.

### Parameters

<code>nObsTypeId</code>	observation type of the query
<code>lTimestamp</code>	timestamp of the query in milliseconds since Epoch
<code>lRefTime</code>	reference time of the query in milliseconds since Epoch
<code>nLat1</code>	minimum latitude of the query in decimal degrees scaled to 7 decimal places
<code>nLon1</code>	minimum longitude of the query in decimal degrees scaled to 7 decimal places
<code>nLat2</code>	maximum latitude of the query in decimal degrees scaled to 7 decimal places
<code>nLon2</code>	maximum longitude of the query in decimal degrees scaled to 7 decimal places

### Returns

`ArrayList` filled with **Obs** that match the query

`static int imrcp.store.NHCWrapper.getSSHWS (double dWindSpeed) [static]`

Determines the category of the storm based off of its wind speed in knots using the Saffir–Simpson hurricane wind scale

### Parameters

<code>dWindSpeed</code>	Maximum wind speed in knots.
-------------------------	------------------------------

### Returns

The category number on the Saffir–Simpson hurricane wind scale

`void imrcp.store.NHCWrapper.load (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId) throws Exception`

Opens the zipped file and parses the cone and track files to create **Obs**

Reimplemented from **imrcp.store.FileWrapper** (*p.224*).

---

## Member Data Documentation

`int [] imrcp.store.NHCWrapper.m_nBB = new int[]{Integer.MAX_VALUE, Integer.MAX_VALUE, Integer.MIN_VALUE, Integer.MIN_VALUE}`

Bounding box of the hurricane cone. [minx, miny, maxx, maxy]

`int [] imrcp.store.NHCWrapper.m_nMaxWindSpeeds = Arrays.newIntArray()`

Stores the predicted max wind speed for each of the hurricane track segments

`int imrcp.store.NHCWrapper.m_nMinPressure = Integer.MIN_VALUE`

Minimum predicted atmospheric pressure for the storm

`ArrayList<HurricaneCenter> imrcp.store.NHCWrapper.m_oCenters = new ArrayList()`

Stores the predicted location of the hurricane center throughout the entire forecast

`ArrayList<Obs> imrcp.store.NHCWrapper.m_oObs = new ArrayList()`

Stores the point, track, and cone hurricane predictions

`int [] imrcp.store.NHCWrapper.m_oPolygon = Arrays.newIntArray()`

Growable array to store the lon/lat coordinates in decimal degrees scaled to 7 decimal places of the hurricane cone of probability.

*See also*

`imrcp.system.Arrays`

`ArrayList<int[]> imrcp.store.NHCWrapper.m_oTracks = new ArrayList()`

List of growable arrays that store the polylines that represent the predicted hurricane track

`String imrcp.store.NHCWrapper.m_sStormName = null`

National Hurricane Center name issued for the storm defined in this file.

`final HashMap<String, Integer> imrcp.store.NHCWrapper.STORMTYPES = new HashMap() [static]`

Maps storm type abbreviations to their IMRCP integer value

---

*The documentation for this class was generated from the following file:*

- `store/NHCWrapper.java`
- 

## imrcp.comp.Notifications Class Reference

### Public Member Functions

- `void reset ()`
- `void process (String[] sMessage)`
- `void createNotifications (FileCache oStore)`

### Additional Inherited Members

---

## Detailed Description

**Notifications** modules subscribe to **Alerts** modules to create Notification observations for configured alert types. The goal is of **Notifications** are to filter sets of **Alerts** to smaller sets focusing on specific alert types that are useful to operators and can pop up on the map UI.

## Author

Federal Highway Administration

---

## Member Function Documentation

### `void imrcp.comp.Notifications.createNotifications (FileCache oStore)`

Queries the given FileCache for current alerts and generates a set of notifications from the result set.

#### Parameters

<code>oStore</code>	Filecahe to query
---------------------	-------------------

### `void imrcp.comp.Notifications.process (String[] sMessage)`

Called when a message from `imrcp.system.Directory` is received. If the message is "new data" `Notifications#createNotifications (imrcp.store.FileCache)` is called with the FileCache that sent the message.

#### Parameters

<code>sMessage</code>	[BaseBlock message is from, message name]
-----------------------	---

Reimplemented from `imrcp.system.BaseBlock` (p.97).

### `void imrcp.comp.Notifications.reset ()`

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from `imrcp.system.BaseBlock` (p.98).

---

*The documentation for this class was generated from the following file:*

- comp/Notifications.java
- 

## imrcp.web.NotificationServlet Class Reference

### Public Member Functions

- `void reset ()`
- `int doNotify (HttpServletRequest iReq, HttpServletResponse iRep, Session oSession)`

### Additional Inherited Members

---

## Detailed Description

This servlet manages requests for getting Notifications on the IMRCP Map UI.

*Author*

Federal Highway Administration

---

**Member Function Documentation**

*int imrcp.web.NotificationServlet.doNotify (HttpServletRequest iReq,  
HttpServletResponse iRep, Session oSession)*

Queries the data stores for notifications and adds them to the response.

*Parameters*

<i>iReq</i>	object that contains the request the client has made of the servlet
<i>iRep</i>	object that contains the response the servlet sends to the client
<i>oSession</i>	object that contains information about the user that made the request

*Returns*

HTTP status code to be included in the response.

*void imrcp.web.NotificationServlet.reset ()*

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.web.SecureBaseBlock** (*p.492*).

---

*The documentation for this class was generated from the following file:*

- web/NotificationServlet.java
- 

**imrcp.comp.NotificationSet Class Reference**

**Public Member Functions**

- boolean **add** (int[] nAdd)
  - int **compareTo** (**NotificationSet** o)
- 

**Detailed Description**

Represents a set of locations for a Notification that have the same start time, end time and value (event type). The int[] in the list are bounding boxes in the format [min lat, max lat, min lon, max lon]

*Author*

Federal Highway Administration

---

## Member Function Documentation

*boolean imrcp.comp.NotificationSet.add (int[] nAdd)*

If `java.util.ArrayList#add(java.lang.Object)` returns true, updates the min and max lon/lat values for the **NotificationSet**

### Parameters

<i>nAdd</i>	location represented by [min lat, max lat, min lon, max lon]
-------------	--

### Returns

value returned by `java.util.ArrayList#add(java.lang.Object)`

*int imrcp.comp.NotificationSet.compareTo (NotificationSet o)*

Compares NotificationSets by value, then start time, then end time.

### Parameters

<i>o</i>	the <b>NotificationSet</b> to be compared
----------	---

### Returns

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

### See also

`java.lang.Comparable::compareTo(java.lang.Object)`

---

*The documentation for this class was generated from the following file:*

- `comp/NotificationSet.java`
- 

## imrcp.store.Obs Class Reference

### Public Member Functions

- **Obs ()**
- **Obs (CsvReader oIn)**
- **Obs (int nObsTypeId, int nContribId, Id oObjId, long lObsTime1, long lObsTime2, long lTimeRecv, int nLat1, int nLon1, int nLat2, int nLon2, short tElev, double dValue)**
- **Obs (int nObsTypeId, int nContribId, Id oObjId, long lObsTime1, long lObsTime2, long lTimeRecv, int nLat1, int nLon1, int nLat2, int nLon2, short tElev, double dValue, short tConf)**
- **Obs (int nObsTypeId, int nContribId, Id oObjId, long lObsTime1, long lObsTime2, long lTimeRecv, int nLat1, int nLon1, int nLat2, int nLon2, short tElev, double dValue, short tConf, String sDetail)**
- **Obs (int nObsTypeId, int nContribId, Id oObjId, long lObsTime1, long lObsTime2, long lTimeRecv, int nLat1, int nLon1, int nLat2, int nLon2, short tElev, double dValue, short tConf, String sDetail, long lClearedTime)**
- void **writeCsv (BufferedWriter oOut)** throws Exception
- boolean **matches (int nObsType, long lStartTime, long lEndTime, long lRefTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon)**
- boolean **matchesPoint (int nObsType, long lStartTime, long lEndTime, long lRefTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon)**

## Public Attributes

- int **m\_nObsTypeId**
- int **m\_nContribId**
- Id **m\_oObjId**
- long **m\_lObsTime1**
- long **m\_lObsTime2**
- long **m\_lTimeRecv**
- int **m\_nLat1**
- int **m\_nLon1**
- int **m\_nLat2**
- int **m\_nLon2**
- short **m\_tElev**
- double **m\_dValue**
- short **m\_tConf**
- String **m\_sDetail** = null
- long **m\_lClearedTime** = Long.MIN\_VALUE

## Static Public Attributes

- static String **CSVHEADER** = "ObsType,Source,ObjId,ObsTime1,ObsTime2,TimeRecv,Lat1,Lon1,Lat2,Lon2,Elev,Value,Conf,Cleared,Detail\n"
- static final Comparator< Obs > **g\_oCompObsByTimeType**
- static final Comparator< Obs > **g\_oCompObsByTimeLonLat**
- static final Comparator< Obs > **g\_oCompObsByTime** = (Obs o1, Obs o2) -> Long.compare(o1.m\_lObsTime1, o2.m\_lObsTime1)
- static final Comparator< Obs > **g\_oCompObsByObjId** = (Obs o1, Obs o2) -> Id.COMPARATOR.compare(o1.m\_oObjId, o2.m\_oObjId)
- static final Comparator< Obs > **g\_oCompByTimeTypeContribLatLon**
- static final Comparator< Obs > **g\_oCompObsByTimeTypeContribObj**
- static final Comparator< Obs > **g\_oCompObsByContrib**
- static final Comparator< Obs > **g\_oCompObsByValue**
- static final Comparator< Obs > **g\_oCompObsByIdTime**
- static final Comparator< Obs > **g\_oCompObsByContribLocation**

---

## Detailed Description

Class used to represent a generic Observation

### Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

### *imrcp.store.Obs.Obs ()*

Default constructor. Does nothing.

### *imrcp.store.Obs.Obs (CsvReader oIn)*

Constructs an **Obs** from the given CsvReader which wraps an InputStream of an IMRCP CSV observation file

#### Parameters

<i>oIn</i>	CsvReader ready to parse the current line of the file, meaning <b>CsvReader#readLine()</b> has already been called
------------	--

*imrcp.store.Obs.Obs (int nObsTypeId, int nContribId, Id oObjId, long lObsTime1, long lObsTime2, long lTimeRecv, int nLat1, int nLon1, int nLat2, int nLon2, short tElev, double dValue)*

Constructs an **Obs** with the given parameters

*Parameters*

<i>nObsTypeId</i>	IMRCP observation id
<i>nContribId</i>	IMRCP contributor id
<i>oObjId</i>	associated object id
<i>lObsTime1</i>	start time in milliseconds since Epoch
<i>lObsTime2</i>	end time in milliseconds since Epoch
<i>lTimeRecv</i>	received time in milliseconds since Epoch
<i>nLat1</i>	minimum latitude in decimal degrees scaled to 7 decimal places
<i>nLon1</i>	minimum longitude in decimal degrees scaled to 7 decimal places
<i>nLat2</i>	maximum latitude in decimal degrees scaled to 7 decimal places, Integer.MIN_VALUE if the obs represents a point
<i>nLon2</i>	maximum longitude in decimal degrees scaled to 7 decimal places, Integer.MIN_VALUE if the obs represents a point
<i>tElev</i>	elevation of obs in meters
<i>dValue</i>	value of the obs

*imrcp.store.Obs.Obs (int nObsTypeId, int nContribId, Id oObjId, long lObsTime1, long lObsTime2, long lTimeRecv, int nLat1, int nLon1, int nLat2, int nLon2, short tElev, double dValue, short tConf)*

Constructs an **Obs** with the given parameters

*Parameters*

<i>nObsTypeId</i>	IMRCP observation id
<i>nContribId</i>	IMRCP contributor id
<i>oObjId</i>	associated object id
<i>lObsTime1</i>	start time in milliseconds since Epoch
<i>lObsTime2</i>	end time in milliseconds since Epoch
<i>lTimeRecv</i>	received time in milliseconds since Epoch
<i>nLat1</i>	minimum latitude in decimal degrees scaled to 7 decimal places
<i>nLon1</i>	minimum longitude in decimal degrees scaled to 7 decimal places
<i>nLat2</i>	maximum latitude in decimal degrees scaled to 7 decimal places, Integer.MIN_VALUE if the obs represents a point
<i>nLon2</i>	maximum longitude in decimal degrees scaled to 7 decimal places, Integer.MIN_VALUE if the obs represents a point
<i>tElev</i>	elevation of obs in meters
<i>dValue</i>	value of the obs
<i>tConf</i>	confidence value

*imrcp.store.Obs.Obs (int nObsTypeId, int nContribId, Id oObjId, long lObsTime1, long lObsTime2, long lTimeRecv, int nLat1, int nLon1, int nLat2, int nLon2, short tElev, double dValue, short tConf, String sDetail)*

Constructs an **Obs** with the given parameters

*Parameters*

<i>nObsTypeId</i>	IMRCP observation id
<i>nContribId</i>	IMRCP contributor id
<i>oObjId</i>	associated object id
<i>lObsTime1</i>	start time in milliseconds since Epoch
<i>lObsTime2</i>	end time in milliseconds since Epoch
<i>lTimeRecv</i>	received time in milliseconds since Epoch
<i>nLat1</i>	minimum latitude in decimal degrees scaled to 7 decimal places
<i>nLon1</i>	minimum longitude in decimal degrees scaled to 7 decimal places
<i>nLat2</i>	maximum latitude in decimal degrees scaled to 7 decimal places, <code>Integer.MIN_VALUE</code> if the obs represents a point
<i>nLon2</i>	maximum longitude in decimal degrees scaled to 7 decimal places, <code>Integer.MIN_VALUE</code> if the obs represents a point
<i>tElev</i>	elevation of obs in meters
<i>dValue</i>	value of the obs
<i>tConf</i>	confidence value
<i>sDetail</i>	additional obs detail

`imrcp.store.Obs.Obs (int nObsTypeId, int nContribId, Id oObjId, long lObsTime1, long lObsTime2, long lTimeRecv, int nLat1, int nLon1, int nLat2, int nLon2, short tElev, double dValue, short tConf, String sDetail, long lClearedTime)`

Constructs an **Obs** with the given parameters

*Parameters*

<i>nObsTypeId</i>	IMRCP observation id
<i>nContribId</i>	IMRCP contributor id
<i>oObjId</i>	associated object id
<i>lObsTime1</i>	start time in milliseconds since Epoch
<i>lObsTime2</i>	end time in milliseconds since Epoch
<i>lTimeRecv</i>	received time in milliseconds since Epoch
<i>nLat1</i>	minimum latitude in decimal degrees scaled to 7 decimal places
<i>nLon1</i>	minimum longitude in decimal degrees scaled to 7 decimal places
<i>nLat2</i>	maximum latitude in decimal degrees scaled to 7 decimal places, <code>Integer.MIN_VALUE</code> if the obs represents a point
<i>nLon2</i>	maximum longitude in decimal degrees scaled to 7 decimal places, <code>Integer.MIN_VALUE</code> if the obs represents a point
<i>tElev</i>	elevation of obs in meters
<i>dValue</i>	value of the obs
<i>tConf</i>	confidence value
<i>sDetail</i>	additional obs detail
<i>lClearedTime</i>	time the obs expires in milliseconds since Epoch, if the obs is not expired use <code>java.lang.Long#MIN_VALUE</code>

## Member Function Documentation

`boolean imrcp.store.Obs.matches (int nObsType, long lStartTime, long lEndTime, long lRefTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon)`

Checks if this **Obs** is valid for the given parameters meaning it has the same observation type id, the start and end time intersect the given time range, the received time is before or equal to the given reference time, if the cleared time is set it must be after the reference time, and the spatial extents of the obs intersect the given bounding box.

### Parameters

<code>nObsType</code>	IMRCP observation id
<code>lStartTime</code>	start time in milliseconds since Epoch
<code>lEndTime</code>	end time in milliseconds since Epoch
<code>lRefTime</code>	reference time in milliseconds since Epoch
<code>nStartLat</code>	minimum latitude in decimal degrees scaled to 7 decimal places
<code>nEndLat</code>	maximum latitude in decimal degrees scaled to 7 decimal places
<code>nStartLon</code>	minimum longitude in decimal degrees scaled to 7 decimal places
<code>nEndLon</code>	maximum longitude in decimal degrees scaled to 7 decimal places

### Returns

true if the **Obs** is valid for the requested parameters, otherwise false

Reimplemented in `imrep.store.CAPObs` (*p.109*).

`boolean imrcp.store.Obs.matchesPoint (int nObsType, long lStartTime, long lEndTime, long lRefTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon)`

Checks if this **Obs** is valid for the given parameters meaning it has the same observation type id, the start and end time intersect the given time range, the received time is before or equal to the given reference time, if the cleared time is set it must be after the reference time, and the spatial extents of the obs intersect the given bounding box.

### Parameters

<code>nObsType</code>	IMRCP observation id
<code>lStartTime</code>	start time in milliseconds since Epoch
<code>lEndTime</code>	end time in milliseconds since Epoch
<code>lRefTime</code>	reference time in milliseconds since Epoch
<code>nStartLat</code>	minimum latitude in decimal degrees scaled to 7 decimal places
<code>nEndLat</code>	maximum latitude in decimal degrees scaled to 7 decimal places
<code>nStartLon</code>	minimum longitude in decimal degrees scaled to 7 decimal places
<code>nEndLon</code>	maximum longitude in decimal degrees scaled to 7 decimal places

### Returns

true if the **Obs** is valid for the requested parameters, otherwise false

`void imrcp.store.Obs.writeCsv (BufferedWriter oOut) throws Exception`

Write the observation to the given Writer as a line of a CSV file.

### Parameters

<i>oOut</i>	Writer to write the observation to
-------------	------------------------------------

### Exceptions

<i>Exception</i>
------------------

---

## Member Data Documentation

```
String imrcp.store.Obs.CSVHEADER =
"ObsType,Source,ObjId,ObsTime1,ObsTime2,TimeRecv,Lat1,Lon1,Lat2,Lon2,Elev,Value
,Conf,Cleared,Detail\n" [static]
```

Header for CSV files that contain **Obs**

```
final Comparator<Obs>
imrcp.store.Obs.g_oCompByTimeTypeContribLatLon [static]
```

```
Initial value:= (Obs o1, Obs o2) ->
{
    int nReturn = Long.compare(o1.m_1ObsTime1, o2.m_1ObsTime1);
    if (nReturn == 0)
    {
        nReturn = o1.m_nObsTypeId - o2.m_nObsTypeId;
        if (nReturn == 0)
        {
            nReturn = o1.m_nContribId - o2.m_nContribId;
            if (nReturn == 0)
            {
                nReturn = o1.m_nLat1 - o2.m_nLat1;
                if (nReturn == 0)
                {
                    nReturn = o1.m_nLon1 - o2.m_nLon1;
                    if (nReturn == 0 && o1.m_nLat2 != Integer.MIN_VALUE &&
o2.m_nLat2 != Integer.MIN_VALUE)
                    {
                        nReturn = o1.m_nLat2 - o2.m_nLat2;
                        if (nReturn == 0)
                            nReturn = o1.m_nLon2 - o2.m_nLon2;
                    }
                }
            }
        }
    }
    return nReturn;
}
```

Compares **Obs** by start time, then observation type, then contributor id, then min lat, then min lon, then max lat, then max lon

```
final Comparator<Obs> imrcp.store.Obs.g_oCompObsByContrib [static]
```

```
Initial value:= (Obs o1, Obs o2) ->
{
    return o1.m_nContribId - o2.m_nContribId;
}
```

Compares **Obs** by contributor id

```
final Comparator<Obs> imrcp.store.Obs.g_oCompObsByContribLocation [static]
```

```
Initial value:= (Obs o1, Obs o2) ->
{
    int nReturn = o1.m_nContribId - o2.m_nContribId;
    if (nReturn == 0)
    {
        nReturn = o1.m_nLat1 - o2.m_nLat1;
        if (nReturn == 0)
        {
            nReturn = o1.m_nLon1 - o2.m_nLon1;
```

```

        if (nReturn == 0 && o1.m_nLat2 != Integer.MIN_VALUE && o2.m_nLat2
!= Integer.MIN_VALUE)
            nReturn = Integer.compare(o1.m_nLat2, o2.m_nLat2);
    }

    return nReturn;
}

```

Compares **Obs** by contributor id then min lat, then min lon, then max lat

*final Comparator<Obs> imrcp.store.Obs.g\_oCompObsByldTime [static]*

```

Initial value:= (Obs o1, Obs o2) ->
{
    int nReturn = Id.COMPARATOR.compare(o1.m_oObjId, o2.m_oObjId);
    if (nReturn == 0)
        nReturn = Long.compare(o1.m_lObsTime1, o2.m_lObsTime1);
    return nReturn;
}

```

Compares **Obs** by associated object id then start time

*final Comparator<Obs> imrcp.store.Obs.g\_oCompObsByObjId = (Obs o1, Obs o2) ->
Id.COMPARATOR.compare(o1.m\_oObjId, o2.m\_oObjId) [static]*

Compares **Obs** by associated object id

*final Comparator<Obs> imrcp.store.Obs.g\_oCompObsByTime = (Obs o1, Obs o2) ->
Long.compare(o1.m\_lObsTime1, o2.m\_lObsTime1) [static]*

Compares **Obs** by start time

*final Comparator<Obs> imrcp.store.Obs.g\_oCompObsByTimeLonLat [static]*

```

Initial value:= (Obs o1, Obs o2) ->
{
    int nRet = Long.compare(o1.m_lObsTime1, o2.m_lObsTime1);
    if (nRet == 0)
    {
        nRet = Integer.compare(o1.m_nLon1, o2.m_nLon1);
        if (nRet == 0)
            nRet = Integer.compare(o1.m_nLat1, o2.m_nLat1);
    }

    return nRet;
}

```

Compares **Obs** by start time then min lon, then min lat

*final Comparator<Obs> imrcp.store.Obs.g\_oCompObsByTimeType [static]*

```

Initial value:= (Obs o1, Obs o2) ->
{
    int nRet = Long.compare(o1.m_lObsTime1, o2.m_lObsTime1);
    if (nRet == 0)
        nRet = o1.m_nObsTypeId - o2.m_nObsTypeId;
    return nRet;
}

```

Compares **Obs** by start time the observation type id

*final Comparator<Obs>*

*imrcp.store.Obs.g\_oCompObsByTimeTypeContribObj [static]*

```

Initial value:= (Obs o1, Obs o2) ->
{
    int nReturn = Long.compare(o1.m_lObsTime1, o2.m_lObsTime1);
    if (nReturn == 0)
    {
        nReturn = o1.m_nObsTypeId - o2.m_nObsTypeId;
        if (nReturn == 0)
        {
            nReturn = o1.m_nContribId - o2.m_nContribId;
            if (nReturn == 0)
            {

```

```

        nReturn = Id.COMPARATOR.compare(o1.m_oObjId, o2.m_oObjId);
    }
}
return nReturn;
}

```

Compares **Obs** by start time, observation type id, contributor id, then associated object id

```
final Comparator<Obs> imrcp.store.Obs.g_oCompObsByValue [static]
Initial value:= (Obs o1, Obs o2) ->
{
    return Double.compare(o1.m_dValue, o2.m_dValue);
}
```

Compares **Obs** by value

*double imrcp.store.Obs.m\_dValue*

Value of the **Obs**

*long imrcp.store.Obs.m\_lClearedTime = Long.MIN\_VALUE*

Time in milliseconds since Epoch the **Obs** stops being valid. Used to show that alerts or events have expired. If an **Obs** hasn't expired, use `Long.MIN_VALUE`

*long imrcp.store.Obs.m\_lObsTime1*

Start time of the **Obs** in milliseconds since Epoch

*long imrcp.store.Obs.m\_lObsTime2*

End time of the **Obs** in milliseconds since Epoch

*long imrcp.store.Obs.m\_lTimeRecv*

Received time of the **Obs** in milliseconds since Epoch

*int imrcp.store.Obs.m\_nContribId*

IMRCP contributor Id which is a computed by converting an up to a 6 character alphanumeric string using base 36.

*int imrcp.store.Obs.m\_nLat1*

Minimum latitude in decimal degrees scaled to 7 decimal places

*int imrcp.store.Obs.m\_nLat2*

Maximum latitude in decimal degrees scaled to 7 decimal places. If the **Obs** represent a single point used `Integer.MIN_VALUE`

*int imrcp.store.Obs.m\_nLon1*

Minimum longitude in decimal degrees scaled to 7 decimal places

*int imrcp.store.Obs.m\_nLon2*

Maximum longitude in decimal degrees scaled to 7 decimal places. If the **Obs** represent a single point used `Integer.MIN_VALUE`

*int imrcp.store.Obs.m\_nObsTypeld*

IMRCP observation type id. The observation types are defined in the `ObsType` class. These values are computed by converting an up to 6 character alphanumeric string in to an integer using base 36

*See also*

`imrcp.system.ObsType`

*Id imrcp.store.Obs.m\_oObjId*

Id of the object the **Obs** is associated with. If there is no associated object use **imrcp.system.Id#NULLID**

*String imrcp.store.Obs.m\_sDetail = null*

Any additional detail for the **Obs** needed by the system.

*short imrcp.store.Obs.m\_tConf*

Confidence value for the **Obs**, the system has not implemented using this yet.

*short imrcp.store.Obs.m\_tElev*

Elevation of the **Obs** in meters.

---

*The documentation for this class was generated from the following file:*

- store/Obs.java
- 

## imrcp.web.ObsChartRequest Class Reference

### Public Member Functions

- int **getObstypeId** ()
  - void **setObstypeId** (int m\_nObstypeId)
- 

### Detailed Description

**ObsRequest** implementation used for chart requests from the IMRCP Map UI

#### Author

Federal Highway Administration

---

### Member Function Documentation

*int imrcp.web.ObsChartRequest.getObstypeId ()*

Gets the IMRCP observation type id of the request

#### Returns

The IMRCP observation type id of the request

*void imrcp.web.ObsChartRequest.setObstypeId (int m\_nObstypeId)*

Sets the IMRCP observation type id of the request

#### Parameters

<i>m_nObstypeId</i>	The desired IMRCP observation type id of the request
---------------------	--

---

*The documentation for this class was generated from the following file:*

- web/ObsChartRequest.java

---

## imrcp.web.ObsInfo Class Reference

### Public Member Functions

- **ObsInfo ()**
- **ObsInfo (int nObsType, int nContribId)**
- **int compareTo (ObsInfo o)**

### Public Attributes

- **int m\_nObsType**
- **int m\_nContribId**

### Static Public Attributes

- static final Comparator<ObsInfo> g\_oCOMP

---

### Detailed Description

Object used as a key in Map objects to store Observations by observation type id and contributor id.

### Author

Federal Highway Administration

---

### Constructor & Destructor Documentation

#### *imrcp.web.ObsInfo.ObsInfo ()*

Default constructor. Does nothing.

#### *imrcp.web.ObsInfo.ObsInfo (int nObsType, int nContribId)*

Constructs an **ObsInfo** with the given parameters

##### Parameters

<i>nObsType</i>	IMRCP observation type id
<i>nContribId</i>	IMRCP contributor id

---

### Member Function Documentation

#### *int imrcp.web.ObsInfo.compareTo (ObsInfo o)*

Compares ObsInfos by observation type id then contributor id

---

### Member Data Documentation

#### *final Comparator<ObsInfo> imrcp.web.ObsInfo.g\_oCOMP [static]*

```
Initial value:= (ObsInfo o1, ObsInfo o2) ->
{
    int nReturn = o1.m_nObsType - o2.m_nObsType;
    if (nReturn == 0)
        nReturn = o1.m_nContribId - o2.m_nContribId;
```

```
        return nReturn;
    }
```

Compares ObsInfos by observation type id then contributor id

*int imrcp.web.ObsInfo.m\_nContribId*

IMRCP contributor id

*int imrcp.web.ObsInfo.m\_nObsType*

IMRCP observation type id

---

*The documentation for this class was generated from the following file:*

- web/ObsInfo.java
- 

## imrcp.web.ObsRequest Class Reference

### Public Member Functions

- **LatLngBounds getRequestBounds ()**
  - void **setRequestBounds (LatLngBounds requestBounds)**
  - long **getRequestTimestampStart ()**
  - void **setRequestTimestampStart (long lRequestTimestamp)**
  - long **getRequestTimestampRef ()**
  - void **setRequestTimestampRef (long lRequestTimestamp)**
  - long **getRequestTimestampEnd ()**
  - long **setRequestTimestampEnd (long lRequestTimestampEnd)**
  - int **getSourceId ()**
  - void **setSourceId (int nSourceId)**
- 

### Detailed Description

This class stores the parameters used for observation requests from the IMRCP Map UI

#### Author

Federal Highway Administration

---

### Member Function Documentation

*LatLngBounds imrcp.web.ObsRequest.getRequestBounds ()*

Getter for **m\_oRequestBounds**

#### Returns

The spatial extents of the request

*long imrcp.web.ObsRequest.getRequestTimestampEnd ()*

Getter for **m\_lRequestTimestampEnd**

#### Returns

End time of the request in milliseconds since Epoch

`long imrcp.web.ObsRequest.getRequestTimestampRef ()`

Getter for `m_lRequestTimestampRef`

*Returns*

Reference time of the query in milliseconds since Epoch

`long imrcp.web.ObsRequest.getRequestTimestampStart ()`

Getter for `m_lRequestTimestampStart`

*Returns*

Start time of the request in milliseconds since Epoch

`int imrcp.web.ObsRequest.getSourceId ()`

Getter for `m_nSourceId`

*Returns*

the IMRCP contributor Id of the request

`void imrcp.web.ObsRequest.setRequestBounds (LatLangBounds requestBounds)`

Setter for `m_oRequestBounds`

*Parameters*

<code>requestBounds</code>	The spatial extents of the request
----------------------------	------------------------------------

`long imrcp.web.ObsRequest.setRequestTimestampEnd (long lRequestTimestampEnd)`

Setter for `m_lRequestTimestampEnd`

*Parameters*

<code>lRequestTimestampEnd</code>	End time of the request in milliseconds since Epoch
-----------------------------------	---

`void imrcp.web.ObsRequest.setRequestTimestampRef (long lRequestTimestamp)`

Setter for `m_lRequestTimestampRef`

*Parameters*

<code>lRequestTimestamp</code>	Reference time of the query in milliseconds since Epoch
--------------------------------	---

`void imrcp.web.ObsRequest.setRequestTimestampStart (long lRequestTimestamp)`

Setter for `m_lRequestTimestampStart`

*Parameters*

<code>lRequestTimestamp</code>	Start time of the request in milliseconds since Epoch
--------------------------------	---

`void imrcp.web.ObsRequest.setSourceId (int nSourceId)`

Setter for `m_nSourceId`

#### Parameters

<i>nSourceId</i>	the IMRCP contributor Id of the request
------------------	---

---

*The documentation for this class was generated from the following file:*

- web/ObsRequest.java
- 

## imrcp.system.ObsType Class Reference

### Static Public Member Functions

- static String **getName** (int nObsTypeId)
- static String **getUnits** (int nObsTypeId)
- static String **getUnits** (int nObsTypeId, boolean bMetric)
- static String **getDescription** (int nObsTypeId)
- static boolean **hasLookup** (int nObstypeId)
- static String **lookup** (int nObsTypeId, int nValue)
- static int **lookup** (int nObsTypeId, String sValue)
- static boolean **isInUse** (int nObsTypeId)
- static **RangeRules getRangeRules** (int nObsType)
- static void **getJsonLookupValues** (StringBuilder sBuf)

### Static Public Attributes

- static final int **TAIR** = Integer.valueOf(OBS\_TYPES[1][0], 36)
- static final int **TDEW** = Integer.valueOf(OBS\_TYPES[2][0], 36)
- static final int **TPVT** = Integer.valueOf(OBS\_TYPES[3][0], 36)
- static final int **PRBAR** = Integer.valueOf(OBS\_TYPES[4][0], 36)
- static final int **PRSUR** = Integer.valueOf(OBS\_TYPES[5][0], 36)
- static final int **RH** = Integer.valueOf(OBS\_TYPES[6][0], 36)
- static final int **VIS** = Integer.valueOf(OBS\_TYPES[7][0], 36)
- static final int **CONPVT** = Integer.valueOf(OBS\_TYPES[8][0], 36)
- static final int **STPV** = Integer.valueOf(OBS\_TYPES[9][0], 36)
- static final int **DIRWND** = Integer.valueOf(OBS\_TYPES[10][0], 36)
- static final int **GSTWND** = Integer.valueOf(OBS\_TYPES[11][0], 36)
- static final int **SPDWND** = Integer.valueOf(OBS\_TYPES[12][0], 36)
- static final int **COVCLD** = Integer.valueOf(OBS\_TYPES[13][0], 36)
- static final int **RTEPC** = Integer.valueOf(OBS\_TYPES[14][0], 36)
- static final int **TYPPC** = Integer.valueOf(OBS\_TYPES[15][0], 36)
- static final int **STG** = Integer.valueOf(OBS\_TYPES[16][0], 36)
- static final int **EVT** = Integer.valueOf(OBS\_TYPES[17][0], 36)
- static final int **VOLLNK** = Integer.valueOf(OBS\_TYPES[18][0], 36)
- static final int **GENLNK** = Integer.valueOf(OBS\_TYPES[19][0], 36)
- static final int **VEHLNK** = Integer.valueOf(OBS\_TYPES[20][0], 36)
- static final int **QUELNK** = Integer.valueOf(OBS\_TYPES[21][0], 36)
- static final int **SPDLNK** = Integer.valueOf(OBS\_TYPES[22][0], 36)
- static final int **DNTLNK** = Integer.valueOf(OBS\_TYPES[23][0], 36)
- static final int **SPFLNK** = Integer.valueOf(OBS\_TYPES[24][0], 36)
- static final int **DNFLNK** = Integer.valueOf(OBS\_TYPES[25][0], 36)
- static final int **CTLEFT** = Integer.valueOf(OBS\_TYPES[26][0], 36)
- static final int **DURGRN** = Integer.valueOf(OBS\_TYPES[27][0], 36)
- static final int **CTTHRU** = Integer.valueOf(OBS\_TYPES[28][0], 36)
- static final int **CTMID** = Integer.valueOf(OBS\_TYPES[29][0], 36)
- static final int **TSSRF** = Integer.valueOf(OBS\_TYPES[30][0], 36)

- static final int **DPHLIQ** = Integer.valueOf(OBS\_TYPES[31][0], 36)
- static final int **DPHSN** = Integer.valueOf(OBS\_TYPES[32][0], 36)
- static final int **FLWCAT** = Integer.valueOf(OBS\_TYPES[33][0], 36)
- static final int **SPDCAT** = Integer.valueOf(OBS\_TYPES[34][0], 36)
- static final int **OCCCAT** = Integer.valueOf(OBS\_TYPES[35][0], 36)
- static final int **QPRLNK** = Integer.valueOf(OBS\_TYPES[36][0], 36)
- static final int **DPHLNK** = Integer.valueOf(OBS\_TYPES[37][0], 36)
- static final int **PCCAT** = Integer.valueOf(OBS\_TYPES[38][0], 36)
- static final int **TRFLNK** = Integer.valueOf(OBS\_TYPES[39][0], 36)
- static final int **TDNLNK** = Integer.valueOf(OBS\_TYPES[40][0], 36)
- static final int **TIMERT** = Integer.valueOf(OBS\_TYPES[41][0], 36)
- static final int **RDR0** = Integer.valueOf(OBS\_TYPES[42][0], 36)
- static final int **NOTIFY** = Integer.valueOf(OBS\_TYPES[43][0], 36)
- static final int **TRSCAT** = Integer.valueOf(OBS\_TYPES[44][0], 36)
- static final int **TRSTRK** = Integer.valueOf(OBS\_TYPES[45][0], 36)
- static final int **TRSCNE** = Integer.valueOf(OBS\_TYPES[46][0], 36)
- static final int **KRTPVT** = Integer.valueOf(OBS\_TYPES[47][0], 36)
- static final int **KTSSRF** = Integer.valueOf(OBS\_TYPES[48][0], 36)
- static final int **SSCST** = Integer.valueOf(OBS\_TYPES[49][0], 36)
- static final int **ALL** = Integer.valueOf(OBS\_TYPES[50][0], 36)
- static final int **RTSLDM** = Integer.valueOf(OBS\_TYPES[51][0], 36)
- static final int **RTLIQM** = Integer.valueOf(OBS\_TYPES[52][0], 36)
- static final int **RTPREM** = Integer.valueOf(OBS\_TYPES[53][0], 36)
- static final int **TPSLDM** = Integer.valueOf(OBS\_TYPES[54][0], 36)
- static final int **TPLIQM** = Integer.valueOf(OBS\_TYPES[55][0], 36)
- static final int **TPPREM** = Integer.valueOf(OBS\_TYPES[56][0], 36)
- static final int **MPLOW** = Integer.valueOf(OBS\_TYPES[57][0], 36)
- static final int **WPLOW** = Integer.valueOf(OBS\_TYPES[58][0], 36)
- static final int **TPLOW** = Integer.valueOf(OBS\_TYPES[59][0], 36)
- static final int **SPDVEH** = Integer.valueOf(OBS\_TYPES[60][0], 36)
- static final int[] **ALL\_OBSTYPES** = Arrays.copyOfRange(TYPE\_MAP, 1, TYPE\_MAP.length)
- static final double **m\_dLIGHTRAIN**
- static final double **m\_dLIGHTSNOW**
- static final double **m\_dMEDIUMSNOW**
- static final double **m\_dRAINTEMP**
- static final double **m\_dSNOWTEMP**
- static final double **m\_dLIGHTRAINMMPERHR**
- static final double **m\_dMEDIUMRAINMMPERHR**
- static final double **m\_dLIGHTSNOWMMPERHR**
- static final double **m\_dMEDIUMSNOWMMPERHR**

## Detailed Description

This class contains the defined observation types available in system as well as the enumerated values for those observation types.

### Author

Federal Highway Administration

## Member Function Documentation

**static String imrcp.system.ObsType.getDescription (int nObsTypId) [static]**

Get the description of the given IMRCP observation type id.

*Parameters*

<i>nObsTypeId</i>	IMRCP observation type id
-------------------	---------------------------

*Returns*

the description associated with the IMRCP observation type id

**static void imrcp.system.ObsType.getJsonLookupValues (StringBuilder sBuf) [static]**

Fills the given StringBuilder with a JSON representation of the enumerated values lookup map.

*Parameters*

<i>sBuf</i>	buffer to add the JSON representation of the look up map to
-------------	---

**static String imrcp.system.ObsType.getName (int nObsTypeld) [static]**

Gets the name of the given IMRCP observation type id.

*Parameters*

<i>nObsTypeId</i>	IMRCP observation type id
-------------------	---------------------------

*Returns*

Name associated with the IMRCP observation type id

**static RangeRules imrcp.system.ObsType.getRangeRules (int nObsType) [static]**

Gets the RangeRules associated with the given IMRCP observation type id

*Parameters*

<i>nObsType</i>	IMRCP observation type id
-----------------	---------------------------

*Returns*

The RangeRules associated with the IMRCP observation type id if it exists, otherwise null.

**static String imrcp.system.ObsType.getUnits (int nObsTypeld) [static]**

Gets the English units of the given IMRCP observation type id.

*Parameters*

<i>nObsTypeId</i>	IMRCP observation type
-------------------	------------------------

*Returns*

English units associated with the IMRCP observation type id

**static String imrcp.system.ObsType.getUnits (int nObsTypeld, boolean bMetric) [static]**

Get the English or metric units of the given IMRCP observation type id based on the metric flag.

*Parameters*

<i>nObsTypeId</i>	IMRCP observation type id
-------------------	---------------------------

<i>bMetric</i>	Flag indicating the metric or English units should be returned. true = metric, false = English
----------------	---

*Returns*

If bMetric is true, the metric units associated with the IMRCP observation type id, otherwise the English units associated with the IMRCP observation type id.

**static boolean imrcp.system.ObsType.hasLookup (int nObstypeId) [static]**

Tells if the given IMRCP observation type id has enumerated values to lookup.

*Parameters*

<i>nObstypeId</i>	IMRCP observation type id
-------------------	---------------------------

*Returns*

true if the IMRCP observation type id has an entry in **LOOKUP**, otherwise false.

**static boolean imrcp.system.ObsType.isInUse (int nObsTypeld) [static]**

Tells if the given IMRCP observation type id is currently in use by the system.

*Parameters*

<i>nObsTypeld</i>	IMRCP observation type id
-------------------	---------------------------

*Returns*

true if the IMRCP observation type id is in use by the system, otherwise false.

**static String imrcp.system.ObsType.lookup (int nObsTypeld, int nValue) [static]**

Gets the String lookup value of the given value of the given IMRCP observation type id

*Parameters*

<i>nObsTypeld</i>	IMRCP observation type
<i>nValue</i>	enumerated value to look up

*Returns*

String associated with the value of the IMRCP observation type id. If the observation type does not have a look up for enumerated values, an empty string is returned

**static int imrcp.system.ObsType.lookup (int nObsTypeld, String sValue) [static]**

Gets the enumerated value of the given look up String of the given IMRCP observation type id

*Parameters*

<i>nObsTypeld</i>	IMRCP observation type id
<i>sValue</i>	look up String to get the enumerated value of

*Returns*

the enumerated value associated with the look up String for the IMRCP observation type id. If an enumerated value is not associated with the String, `Integer.MIN_VALUE` is returned

## Member Data Documentation

*final int imrcp.system.ObsType.ALL = Integer.valueOf(OBS\_TYPES[50][0],  
36) [static]*

all

*final int [] imrcp.system.ObsType.ALL\_OBSTYPE = Arrays.copyOfRange(TYPE\_MAP, 1,  
TYPE\_MAP.length) [static]*

Contains all of the defined observation type ids, without the reserved spot for unknown observation types

*final int imrcp.system.ObsType.CONPVT = Integer.valueOf(OBS\_TYPES[8][0],  
36) [static]*

pavement conductivity

*final int imrcp.system.ObsType.COVCLD = Integer.valueOf(OBS\_TYPES[13][0],  
36) [static]*

total cloud cover

*final int imrcp.system.ObsType.CTLEFT = Integer.valueOf(OBS\_TYPES[26][0],  
36) [static]*

number of left-turning vehicles on each link

*final int imrcp.system.ObsType.CTMID = Integer.valueOf(OBS\_TYPES[29][0],  
36) [static]*

cumulative number of vehicles that pass the mid point of links

*final int imrcp.system.ObsType.CTTHRU = Integer.valueOf(OBS\_TYPES[28][0],  
36) [static]*

number of vehicles that pass through the link

*final int imrcp.system.ObsType.DIRWND = Integer.valueOf(OBS\_TYPES[10][0],  
36) [static]*

wind direction

*final int imrcp.system.ObsType.DNFLNK = Integer.valueOf(OBS\_TYPES[25][0],  
36) [static]*

average density of moving vehicles on each link

*final int imrcp.system.ObsType.DNTLNK = Integer.valueOf(OBS\_TYPES[23][0],  
36) [static]*

average density of vehicles on each link

*final int imrcp.system.ObsType.DPHLIQ = Integer.valueOf(OBS\_TYPES[31][0],  
36) [static]*

liquid inundation depth

*final int imrcp.system.ObsType.DPHLNK = Integer.valueOf(OBS\_TYPES[37][0],  
36) [static]*

link depth

```

final int imrcp.system.ObsType.DPHSN = Integer.valueOf(OBS_TYPES[32][0],
36) [static]
    snow inundation depth

final int imrcp.system.ObsType.DURGRN = Integer.valueOf(OBS_TYPES[27][0],
36) [static]
    average green time for each approach

final int imrcp.system.ObsType.EVT = Integer.valueOf(OBS_TYPES[17][0],
36) [static]
    event

final int imrcp.system.ObsType.FLWCAT = Integer.valueOf(OBS_TYPES[33][0],
36) [static]
    predicted flow category

final int imrcp.system.ObsType.GENLNK = Integer.valueOf(OBS_TYPES[19][0],
36) [static]
    vehicles generate on each generation link

final int imrcp.system.ObsType.GSTWND = Integer.valueOf(OBS_TYPES[11][0],
36) [static]
    wind speed gust

final int imrcp.system.ObsType.KRTPVT = Integer.valueOf(OBS_TYPES[47][0],
36) [static]
    kriged pavement temperature

final int imrcp.system.ObsType.KTSSRF = Integer.valueOf(OBS_TYPES[48][0],
36) [static]
    kriged subsurface temperature

final double imrcp.system.ObsType.m_dLIGHTRAIN [static]
    Light rain threshold in kg/(m^2*s)

final double imrcp.system.ObsType.m_dLIGHTRAINMMPERHR [static]
    light rain threshold in mm/hr

final double imrcp.system.ObsType.m_dLIGHTSNOW [static]
    light snow threshold in kg/(m^2*s)

final double imrcp.system.ObsType.m_dLIGHTSNOWMMPERHR [static]
    light snow threshold in mm/hr

final double imrcp.system.ObsType.m_dMEDIUMRAIN [static]
    Medium rain threshold in kg/(m^2*s)

final double imrcp.system.ObsType.m_dMEDIUMRAINMMPERHR [static]
    medium rain threshold in mm/hr

final double imrcp.system.ObsType.m_dMEDIUMSNOW [static]
    medium snow threshold in kg/(m^2*s)

```

```
final double imrcp.system.ObsType.m_dMEDIUMSNOWMMPERHR [static]
    medium snow threshold in mm/hr

final double imrcp.system.ObsType.m_dRAINTEMP [static]
    temperature for rain threshold in K

final double imrcp.system.ObsType.m_dSNOWTEMP [static]
    temperature for snow threshold in K

final int imrcp.system.ObsType.MPLOW = Integer.valueOf(OBS_TYPES[57][0],
36) [static]
    MAC main plow

final int imrcp.system.ObsType.NOTIFY = Integer.valueOf(OBS_TYPES[43][0],
36) [static]
    notification

final int imrcp.system.ObsType.OCCCAT = Integer.valueOf(OBS_TYPES[35][0],
36) [static]
    predicted occupancy category

final int imrcp.system.ObsType.PCCAT = Integer.valueOf(OBS_TYPES[38][0],
36) [static]
    precipitation category

final int imrcp.system.ObsType.PRBAR = Integer.valueOf(OBS_TYPES[4][0],
36) [static]
    barometric pressure

final int imrcp.system.ObsType.PRSUR = Integer.valueOf(OBS_TYPES[5][0],
36) [static]
    surface pressure

final int imrcp.system.ObsType.QPRLNK = Integer.valueOf(OBS_TYPES[36][0],
36) [static]
    queue percentage on link

final int imrcp.system.ObsType.QUELNK = Integer.valueOf(OBS_TYPES[21][0],
36) [static]
    number of queued vehicles on each link

final int imrcp.system.ObsType.RDRO = Integer.valueOf(OBS_TYPES[42][0],
36) [static]
    merged base reflectivity

final int imrcp.system.ObsType.RH = Integer.valueOf(OBS_TYPES[6][0], 36) [static]
    relative humidity

final int imrcp.system.ObsType.RTEPC = Integer.valueOf(OBS_TYPES[14][0],
36) [static]
    precipitation rate surface
```

```
final int imrcp.system.ObsType.RTLIQM = Integer.valueOf(OBS_TYPES[52][0],  
36) [static]  
    liquid material rate  
  
final int imrcp.system.ObsType.RTPREM = Integer.valueOf(OBS_TYPES[53][0],  
36) [static]  
    prewet material rate  
  
final int imrcp.system.ObsType.RTSLDM = Integer.valueOf(OBS_TYPES[51][0],  
36) [static]  
    solid material rate  
  
final int imrcp.system.ObsType.SPDCAT = Integer.valueOf(OBS_TYPES[34][0],  
36) [static]  
    predicted speed category  
  
final int imrcp.system.ObsType.SPDLNK = Integer.valueOf(OBS_TYPES[22][0],  
36) [static]  
    average speed of vehicles on each link  
  
final int imrcp.system.ObsType.SPDVEH = Integer.valueOf(OBS_TYPES[60][0],  
36) [static]  
    Vehicle Speed  
  
final int imrcp.system.ObsType.SPDWND = Integer.valueOf(OBS_TYPES[12][0],  
36) [static]  
    wind speed  
  
final int imrcp.system.ObsType.SPFLNK = Integer.valueOf(OBS_TYPES[24][0],  
36) [static]  
    average speed of moving vehicles on each link  
  
final int imrcp.system.ObsType.SSCST = Integer.valueOf(OBS_TYPES[49][0],  
36) [static]  
    extra tropical storm surge combined surge and tide  
  
final int imrcp.system.ObsType.STG = Integer.valueOf(OBS_TYPES[16][0],  
36) [static]  
    stage  
  
final int imrcp.system.ObsType.STPVT = Integer.valueOf(OBS_TYPES[9][0],  
36) [static]  
    pavement state  
  
final int imrcp.system.ObsType.TAIR = Integer.valueOf(OBS_TYPES[1][0],  
36) [static]  
    air temperature  
  
final int imrcp.system.ObsType.TDEW = Integer.valueOf(OBS_TYPES[2][0],  
36) [static]  
    dew point temperature
```

```
final int imrcp.system.ObsType.TDNLNK = Integer.valueOf(OBS_TYPES[40][0],  
36) [static]  
    traffic density  
  
final int imrcp.system.ObsType.TIMERT = Integer.valueOf(OBS_TYPES[41][0],  
36) [static]  
    route time  
  
final int imrcp.system.ObsType.TPLIQM = Integer.valueOf(OBS_TYPES[55][0],  
36) [static]  
    liquid material type  
  
final int imrcp.system.ObsType.TPLOW = Integer.valueOf(OBS_TYPES[59][0],  
36) [static]  
    MAC tow plow  
  
final int imrcp.system.ObsType.TPPREM = Integer.valueOf(OBS_TYPES[56][0],  
36) [static]  
    prewet material type  
  
final int imrcp.system.ObsType.TPSLDM = Integer.valueOf(OBS_TYPES[54][0],  
36) [static]  
    solid material type  
  
final int imrcp.system.ObsType.TPVT = Integer.valueOf(OBS_TYPES[3][0],  
36) [static]  
    pavement temperature  
  
final int imrcp.system.ObsType.TRFLNK = Integer.valueOf(OBS_TYPES[39][0],  
36) [static]  
    traffic  
  
final int imrcp.system.ObsType.TRSCAT = Integer.valueOf(OBS_TYPES[44][0],  
36) [static]  
    tropical storm category  
  
final int imrcp.system.ObsType.TRSCNE = Integer.valueOf(OBS_TYPES[46][0],  
36) [static]  
    tropical storm cone  
  
final int imrcp.system.ObsType.TRSTRK = Integer.valueOf(OBS_TYPES[45][0],  
36) [static]  
    tropical storm track  
  
final int imrcp.system.ObsType.TSSRF = Integer.valueOf(OBS_TYPES[30][0],  
36) [static]  
    subsurface temperature  
  
final int imrcp.system.ObsType.TYPPC = Integer.valueOf(OBS_TYPES[15][0],  
36) [static]  
    precipitation type
```

*final int imrcp.system.ObsType.VEHLINK = Integer.valueOf(OBS\_TYPES[20][0],  
36) [static]*

number of vehicles on each link

*final int imrcp.system.ObsType.VIS = Integer.valueOf(OBS\_TYPES[7][0], 36) [static]*

surface visibility

*final int imrcp.system.ObsType.VOLLINK = Integer.valueOf(OBS\_TYPES[18][0],  
36) [static]*

link volume

*final int imrcp.system.ObsType.WPLOW = Integer.valueOf(OBS\_TYPES[58][0],  
36) [static]*

MAC wing plow

---

*The documentation for this class was generated from the following file:*

- system/ObsType.java
- 

## imrcp.store.ObsView Class Reference

### Public Member Functions

- ResultSet **getData** (int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)
- ResultSet **getData** (int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime, **Id** oObjId)

### Additional Inherited Members

---

#### Detailed Description

**ObsView** acts as a store of data stores. Most queries for data from the system should access the **ObsView** block.

#### Author

Federal Highway Administration

---

#### Member Function Documentation

*ResultSet imrcp.store.ObsView.getData (int nType, long lStartTime, long lEndTime,  
int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)*

Wrapper for `getData(int, long, long, int, int, int, int, long, imrcp.system.Id)` with `imrcp.system.Id#NULLID` passed as the object id.

#### Parameters

<i>nType</i>	IMRCP observation id
<i>lStartTime</i>	start time in milliseconds since Epoch

<i>lEndTime</i>	end time in milliseconds since Epoch
<i>nStartLat</i>	minimum latitude in decimal degrees scaled to 7 decimal places
<i>nEndLat</i>	maximum latitude in decimal degrees scaled to 7 decimal places
<i>nStartLon</i>	minimum longitude in decimal degrees scaled to 7 decimal places
<i>nEndLon</i>	maximum longitude in decimal degrees scaled to 7 decimal places
<i>lRefTime</i>	reference time in milliseconds since Epoch

#### Returns

a ResultSet with 0 or more **Obs** that are valid for the query

Reimplemented from **imrcp.system.BaseBlock** (p.97).

*ResultSet imrcp.store.ObsView.getData (int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime, Id oObjId)*

#### Parameters

<i>nType</i>	IMRCP observation id
<i>lStartTime</i>	start time in milliseconds since Epoch
<i>lEndTime</i>	end time in milliseconds since Epoch
<i>nStartLat</i>	minimum latitude in decimal degrees scaled to 7 decimal places
<i>nEndLat</i>	maximum latitude in decimal degrees scaled to 7 decimal places
<i>nStartLon</i>	minimum longitude in decimal degrees scaled to 7 decimal places
<i>nEndLon</i>	maximum longitude in decimal degrees scaled to 7 decimal places
<i>lRefTime</i>	reference time in milliseconds since Epoch
<i>oObjId</i>	if this is not <b>imrcp.system.Id#NULLID</b> then only obs that are associated with this Id will be added to the ResultSet

#### Returns

a ResultSet with 0 or more **Obs** that are valid for the query

*The documentation for this class was generated from the following file:*

- store/ObsView.java

## imrcp.collect.OhGo Class Reference

### Public Member Functions

- void **reset** ()
- boolean **start** ()
- void **execute** ()

### Additional Inherited Members

## Detailed Description

Collector for the **OhGo** system.

### Author

Federal Highway Administration

---

## Member Function Documentation

***void imrcp.collect.OhGo.execute ()***

Downloads the configured file from the **OhGo** system.

Reimplemented from **imrcp.system.BaseBlock** (p.95).

***void imrcp.collect.OhGo.reset ()***

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.collect.Collector** (p.114).

***boolean imrcp.collect.OhGo.start ()***

Sets a schedule to execute on a fixed interval.

### Returns

Reimplemented from **imrcp.system.BaseBlock** (p.99).

---

*The documentation for this class was generated from the following file:*

- collect/OhGo.java
- 

## imrcp.comp.OhGoConstruction Class Reference

### Protected Member Functions

- long **getEvents** (String sFile, long lTime) throws Exception

### Additional Inherited Members

---

## Detailed Description

Processes the construction xml files downloaded from OhGo and converts them to IMRCP's incident and work zone .csv file format

### Author

Federal Highway Administration

---

## Member Function Documentation

*long imrcp.comp.OhGoConstruction.getEvents (String sFile, long lTime) throws Exception [protected]*

Parses the given OhGo construction xml file and updates the currently active events in memory.

### Parameters

<i>sFile</i>	OhGo construction xml file to parse
<i>lTime</i>	Timestamp for the file in milliseconds since Epoch

### Returns

Timestamp the file was last updated in milliseconds since Epoch

### Exceptions

<i>Exception</i>
------------------

Reimplemented from **imrcp.comp.EventComp** (*p.204*).

---

*The documentation for this class was generated from the following file:*

- comp/OhGoConstruction.java
- 

## imrcp.comp.OhGoIncidents Class Reference

### Protected Member Functions

- long **getEvents** (String sFile, long lTime) throws Exception

### Additional Inherited Members

---

#### Detailed Description

Processes the incident xml files downloaded from OhGo and converts them to IMRCP's incident and work zone .csv file format

#### Author

Federal Highway Administration

---

## Member Function Documentation

*long imrcp.comp.OhGoIncidents.getEvents (String sFile, long lTime) throws Exception [protected]*

Parses the given OhGo incident xml file and updates the currently active events in memory.

### Parameters

<i>sFile</i>	OhGo incident xml file to parse
<i>lTime</i>	Timestamp for the file in milliseconds since Epoch

#### Returns

Timestamp the file was last updated in milliseconds since Epoch

#### Exceptions

Exception	
Reimplemented from <b>imrcp.comp.EventComp</b> ( <i>p.204</i> ).	

---

*The documentation for this class was generated from the following file:*

- comp/OhGoIncidents.java
- 

## imrcp.geosrv.osm.OsmBinParser Class Reference

### Public Member Functions

- **OsmBinParser ()**
  - void **parseHashes** (String sFile, int nMinLon, int nMinLat, int nMaxLon, int nMaxLat, HashMap< Integer, ArrayList< OsmWay > > oHashes) throws IOException
  - void **parseHash** (String sFile, int nHash, ArrayList< OsmNode > oNodes, ArrayList< OsmWay > oWays) throws Exception
  - int[] **parseFile** (String sFile, ArrayList< OsmNode > oNodes, ArrayList< OsmWay > oWays, StringPool oMainPool) throws IOException
  - void **parseFileWithFilters** (String sFile, ArrayList< OsmNode > oAllNodes, ArrayList< OsmWay > oAllWays, int[] nPolygon, int[] nBoundingBox, StringPool oMainPool, String[] sFilter)
- 

### Detailed Description

The class is used to parse IMRCP OSM binary files which contain the definitions of roadway segments available to the system

#### Author

Federal Highway Administration

---

### Constructor & Destructor Documentation

#### *imrcp.geosrv.osm.OsmBinParser.OsmBinParser ()*

Default constructor. Does nothing.

---

### Member Function Documentation

#### *int[] imrcp.geosrv.osm.OsmBinParser.parseFile (String sFile, ArrayList< OsmNode > oNodes, ArrayList< OsmWay > oWays, StringPool oMainPool) throws IOException*

Parses the given IMRCP OSM binary file, adding all of the roadway segment and node definitions to the given list. This method is primarily used for IMRCP OSM binary files of Networks that have already been created.

*Parameters*

<i>sFile</i>	Path to the IMRCP OSM binary file
<i>oNodes</i>	List to be filled with nodes, should be empty
<i>oWays</i>	List to be filled with roadway segments, should be empty
<i>oMainPool</i>	StringPool to use when parsing key and value Strings

*Returns*

a bounding box of all the roadway segments. [min lon, min lat, max lon, max lat], all the values are in decimal degrees scaled to 7 decimal places

*Exceptions*

<i>IOException</i>	
--------------------	--

```
void imrcp.geosrv.osm.OsmBinParser.parseFileWithFilters (String sFile, ArrayList<OsmNode> oAllNodes, ArrayList<OsmWay> oAllWays, int[] nPolygon, int[] nBoundingBox, StringPool oMainPool, String[] sFilter)
```

Parses the IMRCP OSM binary file, filtering which roadway segments and nodes are added to the list based on the given polygon and highway tag filters. This method is primarily used to aid in the creation of a **Network** by parsing a statewide IMRCP OSM binary file to get the desired roadway segments and node for the **Network**.

*Parameters*

<i>sFile</i>	Path to the IMRCP OSM binary file
<i>oAllNodes</i>	List to be filled with nodes, it may already contain some nodes that have to be sorted by <b>OsmNode#NODEBYTEID</b>
<i>oAllWays</i>	List to be filled with ways, it may already contain some ways that have to be sorted by <b>OsmWay#WAYBYTEID</b>
<i>nPolygon</i>	Array of coordinates describing a polygon, only roadway segments that intersect this polygon will be added to the list. Format is [y0, x0, y1, x1, ..., yn, xn, y0, x0], all coordinates are in decimal degrees scaled to 7 decimal places
<i>nBoundingBox</i>	Array of coordinates describing the bounding box of the polygon. Format is [min lon, min lat, max lon, max lat]
<i>oMainPool</i>	StringPool to use when parsing key and value Strings
<i>sFilter</i>	Array containing the values of the "highway" tag of the roadway segments that are to be included in the list

```
void imrcp.geosrv.osm.OsmBinParser.parseHash (String sFile, int nHash, ArrayList<OsmNode> oNodes, ArrayList<OsmWay> oWays) throws Exception
```

Parses the given file and its index to fill the lists with the nodes and roadway segments that are in the given hash bucket index .

*Parameters*

<i>sFile</i>	IMRCP OSM binary file
<i>nHash</i>	hash bucket index
<i>oNodes</i>	list to be filled with the nodes in the hash bucket, the list is cleared before nodes are added
<i>oWays</i>	list to be filled with the ways in the hash bucket, the list is cleared before ways are added

#### Exceptions

<i>Exception</i>	
------------------	--

`void imrcp.geosrv.osm.OsmBinParser.parseHashes (String sFile, int nMinLon, int nMinLat, int nMaxLon, int nMaxLat, HashMap< Integer, ArrayList< OsmWay > > oHashes) throws IOException`

Parses the hash index file associated with the given IMRCP OSM binary file adding entries for the hash buckets that intersect the given bounding box into the HashMap

#### Parameters

<i>sFile</i>	Path of the IMRCP OSM binary file
<i>nMinLon</i>	minimum longitude of the bounding box in decimal degrees scaled to 7 decimal places
<i>nMinLat</i>	minimum latitude of the bounding box in decimal degrees scaled to 7 decimal places
<i>nMaxLon</i>	maximum longitude of the bounding box in decimal degrees scaled to 7 decimal places
<i>nMaxLat</i>	maximum latitude of the bounding box in decimal degrees scaled to 7 decimal places
<i>oHashes</i>	HashMap to map a hash bucket value to a list of OsmWays in that hash bucket

#### Exceptions

<i>IOException</i>	
--------------------	--

---

*The documentation for this class was generated from the following file:*

- geosrv/osm/OsmBinParser.java
- 

## imrcp.geosrv.osm.OsmBz2ToBin Class Reference

### Public Member Functions

- **OsmBz2ToBin ()**
- **void startElement (String sUri, String sLocalName, String sQname, Attributes iAtt)**
- **void endElement (String sUri, String sLocalName, String sQname)**
- **StringPool convertFile (String sFile, int nTol) throws Exception**

### Static Public Member Functions

- **static void writeBin (String sFile, StringPool oStringPool, ArrayList< OsmNode > oNodes, ArrayList< OsmWay > oWays) throws Exception**
  - **static void writeHashIndex (String sFile, ArrayList< HashBucket > oBuckets) throws Exception**
- 

### Detailed Description

This class is used to convert Open Street Map .xml.bz2 files and convert them to IMRCP OSM binary file which is a more compact form

## Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

*imrcp.geosrv.osm.OsmBz2ToBin.OsmBz2ToBin ()*

Default constructor. Does nothing.

---

## Member Function Documentation

*StringPool imrcp.geosrv.osm.OsmBz2ToBin.convertFile (String sFile, int nTol) throws Exception*

Converts the given OSM .xml.bz file into IMRCP's OSM binary file.

### Parameters

<i>sFile</i>	OSM .xml.bz file to convert
<i>nTol</i>	tolerance in decimal degrees scaled to 7 decimal places used when comparing distance between nodes

### Returns

the StringPool created by parsing the file

### Exceptions

<i>Exception</i>
------------------

*static void imrcp.geosrv.osm.OsmBz2ToBin.writeBin (String sFile, StringPool oStringPool, ArrayList<OsmNode> oNodes, ArrayList<OsmWay> oWays) throws Exception [static]*

Writes the given string pool, list of nodes, and list of ways into the IMRCP OSM binary file format.

### Parameters

<i>sFile</i>	Path of the original OSM file that is being converted
<i>oStringPool</i>	String pool contain all of the tag's keys and values
<i>oNodes</i>	Node to write to the file
<i>oWays</i>	Way to write to the file

### Exceptions

<i>Exception</i>
------------------

*static void imrcp.geosrv.osm.OsmBz2ToBin.writeHashIndex (String sFile, ArrayList<HashBucket> oBuckets) throws Exception [static]*

Writes the index file for the given OSM file which assists in quick look up of node and way definitions via a spatial index using hash buckets.

### Parameters

<i>sFile</i>	Path of the original OSM file that is being converted
--------------	---

<i>oBuckets</i>	List of hash buckets that were generated from the file
-----------------	--

*Exceptions*

<i>Exception</i>
------------------

---

*The documentation for this class was generated from the following file:*

- geosrv/osm/OsmBz2ToBin.java
- 

## imrcp.geosrv.osm.OsmNode Class Reference

### Public Member Functions

- **OsmNode ()**
- **OsmNode (DataInputStream oIn, ArrayList< String > oPool, int[] nFp)** throws IOException
- **OsmNode (long lId, int nLat, int nLon)**
- **OsmNode (int nLat, int nLon)** throws IOException
- **void removeRef (OsmWay oRemove)**
- **void addRef (OsmWay oAdd)**
- **void generateId ()** throws IOException
- **String toString ()**

### Public Attributes

- int **m\_nLat**
- int **m\_nLon**
- ArrayList< **OsmWay** > **m\_oRefs**
- int **m\_nHash**
- ArrayList< **OsmNode** > **m\_oSeps** = null
- boolean **m\_bDrawConnector** = true
- int **m\_nOriginalRefCount** = 0
- int **m\_nOriginalRefForSep** = 0
- double **m\_dHdg**

### Static Public Attributes

- static Comparator< **OsmNode** > **GEOCOMP**
  - static Comparator< **OsmNode** > **NODEBYTEID**
  - static Comparator< **OsmNode** > **LONGID** = (**OsmNode** o1, **OsmNode** o2) -> {return Long.compare(o1.m\_lId, o2.m\_lId);}
- 

### Detailed Description

Represents an Open Street Map Node which make up Open Street Map Ways.

### Author

Federal Highway Administration

---

### Constructor & Destructor Documentation

#### *imrcp.geosrv.osm.OsmNode.OsmNode ()*

Default Constructor. Initializes **m\_oRefs**

*imrcp.geosrv.osm.OsmNode.OsmNode (DataInputStream oIn, ArrayList< String > oPool, int[] nFp) throws IOException*

Constructs an **OsmNode** from the given DataInputStream which is wrapper the input stream of an IMRCP OSM binary file

*Parameters*

<i>oIn</i>	input stream of the IMRCP OSM binary file
<i>oPool</i>	String Pool that contains the possible key and value strings
<i>nFp</i>	current position in the file. An array of size one is used to store this value so it can be updated and used by other methods

*Exceptions*

<i>IOException</i>
--------------------

*imrcp.geosrv.osm.OsmNode.OsmNode (long lId, int nLat, int nLon)*

Constructs an **OsmNode** with the given parameters. Initializes **m\_oRefs** and **m\_nHash**

*Parameters*

<i>lId</i>	OSM id of the node
<i>nLat</i>	latitude in decimal degrees scaled to 7 decimal places
<i>nLon</i>	longitude in decimal degrees scaled to 7 decimal places

*imrcp.geosrv.osm.OsmNode.OsmNode (int nLat, int nLon) throws IOException*

Constructs an **OsmNode** with the given parameters. Initializes **m\_oid**

*Parameters*

<i>nLat</i>	latitude in decimal degrees scaled to 7 decimal places
<i>nLon</i>	longitude in decimal degrees scaled to 7 decimal places

*Exceptions*

<i>IOException</i>
--------------------

---

## Member Function Documentation

*void imrcp.geosrv.osm.OsmNode.addRef (OsmWay oAdd)*

Adds the given **OsmWay** to **m\_oRefs**

*Parameters*

<i>oAdd</i>	The <b>OsmWay</b> to add
-------------	--------------------------

*void imrcp.geosrv.osm.OsmNode.generateId () throws IOException*

Generates the Id for this Node which is based off of its latitude and longitude

*Exceptions*

<i>IOException</i>
--------------------

*void imrcp.geosrv.osm.OsmNode.removeRef (OsmWay oRemove)*

Removes the given **OsmWay** from **m\_oRefs**

#### Parameters

<code>oRemove</code>	The <b>OsmWay</b> to remove
----------------------	-----------------------------

`String imrcp.geosrv.osm.OsmNode.toString ()`

Returns the longitude and latitude concatenated together as a string

#### Returns

---

#### Member Data Documentation

`Comparator<OsmNode> imrcp.geosrv.osm.OsmNode.GEOCOMP [static]`

```
Initial value:= (OsmNode o1, OsmNode o2) ->
{
    int nReturn = Integer.compare(o1.m_nLon, o2.m_nLon);
    if (nReturn == 0)
    {
        nReturn = Integer.compare(o1.m_nLat, o2.m_nLat);
    }
    return nReturn;
}
```

C.compares OsmNodes by longitude then latitude

`Comparator<OsmNode> imrcp.geosrv.osm.OsmNode.LONGID = (OsmNode o1,
OsmNode o2) -> {return Long.compare(o1.m_llid, o2.m_llid);} [static]`

C.compares OsmNodes by OSM id

`boolean imrcp.geosrv.osm.OsmNode.m_bDrawConnector = true`

Not used for anything at the moment

`double imrcp.geosrv.osm.OsmNode.m_dHdg`

Heading in radians to the next node of the **OsmWay** that is currently being processed. This value must be reset for each **OsmWay** it is a part of, usually done by `OsmWay#setHdgs()`

`int imrcp.geosrv.osm.OsmNode.m_nHash`

The hash index of the **HashBucket** this node is in

`int imrcp.geosrv.osm.OsmNode.m_nLat`

Latitude of the node in decimal degrees scaled to 7 decimal places

`int imrcp.geosrv.osm.OsmNode.m_nLon`

Longitude of the node in decimal degrees scaled to 7 decimal places

`int imrcp.geosrv.osm.OsmNode.m_nOriginalRefCount = 0`

The number of OsmWays this Node is a part of in the original Open Street Map definitions

`int imrcp.geosrv.osm.OsmNode.m_nOriginalRefForSep = 0`

The number of OsmWays this Node is a part of at the start of the Separate algorithm

`ArrayList<OsmWay> imrcp.geosrv.osm.OsmNode.m_oRefs`

A list of OsmWays that contain this Node and is sorted using `OsmWay#WAYBYTEID`

*ArrayList<OsmNode> imrcp.geosrv.osm.OsmNode.m\_oSeps = null*

List of nodes that were created from this node during the Separate algorithm

*Comparator<OsmNode> imrcp.geosrv.osm.OsmNode.NODEBYTEID [static]*

```
Initial value:= (OsmNode o1, OsmNode o2) ->
{
    return Id.COMPARATOR.compare(o1.m_oId, o2.m_oId);
}
```

Compare OsmNodes by IMRCP Id

---

*The documentation for this class was generated from the following file:*

- geosrv/osm/OsmNode.java
- 

## imrcp.geosrv.osm.OsmObject Class Reference

### Public Member Functions

- **OsmObject ()**
- **OsmObject (long lId)**

### Public Attributes

- long **m\_lId**
- int **m\_nFp**
- **Id m\_oId**

### Static Public Attributes

- static Comparator< **OsmObject** > **FPCOMP** = (**OsmObject** o1, **OsmObject** o2) -> {return Integer.compare(o1.m\_nFp, o2.m\_nFp);}
  - static Comparator< **OsmObject** > **LONGID** = (**OsmObject** o1, **OsmObject** o2) -> {return Long.compare(o1.m\_lId, o2.m\_lId);}
- 

### Detailed Description

Base class that contains field common to Open Street Map Nodes and Ways. Implemented as a hash map to store the key/value pairs of the nodes and ways

#### Author

Federal Highway Administration

---

### Constructor & Destructor Documentation

*imrcp.geosrv.osm.OsmObject.OsmObject ()*

Default constructor. Does nothing.

*imrcp.geosrv.osm.OsmObject.OsmObject (long lId)*

Constructs an **OsmObject** with the given OSM id.

#### Parameters

<i>lId</i>	Osm id of the object
------------	----------------------

---

## Member Data Documentation

*Comparator<OsmObject> imrcp.geosrv.osm.OsmObject.FPCOMP = (OsmObject o1, OsmObject o2) -> {return Integer.compare(o1.m\_nFp, o2.m\_nFp);} [static]*

Compares OsmObjects by File position

*Comparator<OsmObject> imrcp.geosrv.osm.OsmObject.LONGID = (OsmObject o1, OsmObject o2) -> {return Long.compare(o1.m\_lld, o2.m\_lld);} [static]*

Compare OsmObjects by Osm Id

*long imrcp.geosrv.osm.OsmObject.m\_lld*

Osm id

*int imrcp.geosrv.osm.OsmObject.m\_nFp*

File position. This field is used when creating and parsing IMRCP binary representation of OSM files

*Id imrcp.geosrv.osm.OsmObject.m\_old*

IMRCP Id

---

*The documentation for this class was generated from the following file:*

- geosrv/osm/OsmObject.java

---

## imrcp.geosrv.osm.OsmUtil Class Reference

### Static Public Member Functions

- static double **dist** (**OsmNode** o1, **OsmNode** o2)
- static boolean **nodesEqual** (**OsmNode** o1, **OsmNode** o2, int nTol)
- static **ArrayList< OsmWay >** **merge** (**ArrayList< OsmWay >** oWays) throws **IOException**
- static **OsmWay** **forceMerge** (**OsmWay** oCur, **OsmWay** oCmp) throws **IOException**
- static **OsmWay** **forceSplit** (**OsmWay** oWay, int nSplitIndex) throws **IOException**
- static boolean **merge** (**OsmWay** oCur, **OsmWay** oCmp, int nMergeType, double dAngleThreshold) throws **IOException**
- static **ArrayList< OsmWay >** **split** (**ArrayList< OsmWay >** oWays) throws **Exception**
- static boolean **include** (String sType, String[] sInclude)
- static **ArrayList< OsmNode >** **removeZeroRefs** (**ArrayList< OsmNode >** oNodes)
- static void **determineRamps** (**ArrayList< OsmWay >** oWays, **HashMap< String, ArrayList< int[] > >** oStates, String sOsmDir, String[] sFilter, **HashMap< String, HashMap< Integer, ArrayList< OsmWay > >>** oHashes)
- static void **determineRamps** (**ArrayList< OsmWay >** oWays, **HashMap< String, ArrayList< int[] > >** oStates, String sOsmDir, String[] sFilter)
- static **ArrayList< OsmWay >** **separate** (**ArrayList< OsmWay >** oWays, **ArrayList< OsmNode >** oNodes, int nSepDist) throws **IOException**
- static void **addSeparatedNodes** (**ArrayList< OsmWay >** oWays, double dAngle, **ArrayList< OsmNode >** oSepNodes, **OsmNode** oSepPoint, **OsmWay** oSepWay, int nSepDist, double dHdg, **ArrayList< OsmNode >** oNodes, boolean bPos) throws **IOException**
- static **ArrayList< OsmWay >** **extraSegs** (**ArrayList< OsmWay >** oWays, **ArrayList< OsmNode >** oNodes, int nDistThresh) throws **Exception**

- static void **writeLanesAndSpeeds** (String sGeoFile, ArrayList< OsmWay > oWays, boolean bAppend) throws Exception
- static ArrayList< OsmWay > **finalizeNetwork** (String sGeoFile, String sStateShp, String[] sStates, String[] sFilter, String[] sOptions, String sOsmDir) throws Exception

### Static Public Attributes

- static final String[] **ALL\_ROADS** = {"motorway", "trunk", "primary", "secondary", "tertiary", "motorway\_link", "trunk\_link", "primary\_link", "secondary\_link", "tertiary\_link", "unclassified", "residential"}

### Detailed Description

This class contains utility methods for Open Street Maps(OSM) objects and algorithms used for generating road network models

#### Author

Federal Highway Administration

### Member Function Documentation

*static void imrcp.geosrv.osm.OsmUtil.addSeparatedNodes (ArrayList< OsmWay > oWays, double dAngle, ArrayList< OsmNode > oSepNodes, OsmNode oSepPoint, OsmWay oSepWay, int nSepDist, double dHdg, ArrayList< OsmNode > oNodes, boolean bPos) throws IOException [static]*

Handles the different cases of roadway segments intersecting at the given node to separate. Adds the new node to the overall node list and updates references as needed for the segments that may or may not have been separated already that share the intersection point.

#### Parameters

<i>oWays</i>	the ways that shared an endpoint with the way being separated
<i>dAngle</i>	the calculated angle in radians of way in position 0 of <i>oWays</i>
<i>oSepNodes</i>	the negative or positive tangent node list of the roadway segment being separated
<i>oSepPoint</i>	the node that is being separated
<i>oSepWay</i>	The roadway segment being separated
<i>nSepDist</i>	distance in decimal degrees scaled to 7 decimal places to place nodes of the newly separated roadway segments from the center line of the bi-directional road being separated.
<i>dHdg</i>	the heading in radians of the direction of travel
<i>oNodes</i>	list of all the nodes in the network being processed
<i>bPos</i>	flag indicating if the positive(true) or negative(false) tangent is being processed

#### Exceptions

<i>IOException</i>	
--------------------	--

```
static void imrcp.geosrv.osm.OsmUtil.determineRamps (ArrayList<OsmWay> oWays,
HashMap< String, ArrayList< int[]> > oStates, String sOsmDir, String[]
sFilter) [static]
```

Wrapper for `determineRamps (java.util.ArrayList, java.lang.String, java.lang.String[], java.util.HashMap)`

#### Parameters

<i>oWays</i>	List of roadway segments to process, represents a road network
<i>oStates</i>	maps state names to the geometry of their border. This should contain the states that intersect the road network
<i>sOsmDir</i>	base directory for osm files
<i>sFilter</i>	filter strings used to create the road network

```
static void imrcp.geosrv.osm.OsmUtil.determineRamps (ArrayList<OsmWay> oWays,
HashMap< String, ArrayList< int[]> > oStates, String sOsmDir, String[] sFilter,
HashMap< String, HashMap< Integer, ArrayList< OsmWay >> > oHashes) [static]
```

Determines the type of ramp for each roadway segment in the given list and the number of on and off ramps highway segments have. It does this by checking the "highway" from the OSM database. A "connector" is a segment with a highway tag of "motorway\_link" or "trunk\_link" that is in between two segments with a highway tag of "motorway" or "trunk". A "ramp" is a segment with a highway tag of "motorway\_link" or "trunk\_link" that is in between a segment with a highway of "motorway" or "trunk" and other segment with a highway tag of lower classification (primary, secondary, etc).

#### Parameters

<i>oWays</i>	List of roadway segments to process, represents a road network
<i>oStates</i>	maps state names to the geometry of their border. This should contain the states that intersect the road network
<i>sOsmDir</i>	base directory for osm files
<i>sFilter</i>	filter strings used to create the road network
<i>oHashes</i>	Maps hash indices to the roadway segments contained in that grid.

```
static double imrcp.geosrv.osm.OsmUtil.dist (OsmNode o1, OsmNode o2) [static]
```

Gets the distance between two nodes in decimal degrees scaled to 7 decimal places. This function does not take in account that the earth is spherical, should only be used for points that are relatively close.

#### Parameters

<i>o1</i>	node 1
<i>o2</i>	node 2

#### Returns

Distance between the two nodes in decimal degrees scaled to 7 decimal places.

```
static ArrayList< OsmWay > imrcp.geosrv.osm.OsmUtil.extraSegs (ArrayList< OsmWay > oWays, ArrayList< OsmNode > oNodes, int nDistThresh) throws Exception [static]
```

For each roadway segment in the given list, if the length of that segment is greater than the distance threshold, that segment is split into smaller segments.

#### Parameters

<i>oWays</i>	list of roadway segments to process
<i>oNodes</i>	list of all the nodes making up the roadway segments
<i>nDistThresh</i>	distance in decimal degrees scaled to 7 decimal places that segment's lengths must be less than

#### Returns

a list of all the road segments after processing is finished

#### Exceptions

<i>Exception</i>
------------------

```
static ArrayList< OsmWay > imrcp.geosrv.osm.OsmUtil.finalizeNetwork (String sGeoFile, String sStateShp, String[] sStates, String[] sFilter, String[] sOptions, String sOsmDir) throws Exception [static]
```

Finalizes the network defined by *sGeoFile*. The split, merge, extrasegs, ramp determination, and separate algorithms are ran on the roadway segments in the network.

#### Parameters

<i>sGeoFile</i>	path to the IMRCP OSM binary file defining the network
<i>sStateShp</i>	path to the shapefile containing US state border definitions
<i>sStates</i>	contains the US states the road network intersects
<i>sFilter</i>	contains the filter Strings used to create the network
<i>sOptions</i>	contains the option Strings used to create the network
<i>sOsmDir</i>	base directory for OSM files

#### Returns

a list containing the roadway segments after all of the algorithms have been ran on the network.

#### Exceptions

<i>Exception</i>
------------------

```
static OsmWay imrcp.geosrv.osm.OsmUtil.forceMerge (OsmWay oCur, OsmWay oCmp) throws IOException [static]
```

Merges the two roadway segments regardless of the angle between them and their bridge flags.

#### Parameters

<i>oCur</i>	roadway segment 1
<i>oCmp</i>	roadway segment 2

*Returns*

a new instance of an **OsmWay** with the merged nodes of the original two ways

*Exceptions*

<i>IOException</i>
--------------------

**static OsmWay imrcp.geosrv.osm.OsmUtil.forceSplit (OsmWay oWay, int nSplitIndex)**  
**throws IOException [static]**

Splits the given roadway segment at the given index of its node list. The reference of **oWay** will contain the nodes from 0 to **nSplitIndex** and the returned **OsmWay** with contains the nodes from **nSplitIndex** to the end of the list.

*Parameters*

<i>oWay</i>	roadway segment to split
<i>nSplitIndex</i>	index in the node list to split the way at

*Returns*

the new **OsmWay** created from the nodes from **nSplitIndex** to the end of the list.

*Exceptions*

<i>IOException</i>
--------------------

**static boolean imrcp.geosrv.osm.OsmUtil.include (String sType, String[] sInclude) [static]**

Determines if the given string is in the given string array.

*Parameters*

<i>sType</i>	string to look for
<i>sInclude</i>	array of strings to look in

*Returns*

true if **sType** is in **sInclude** otherwise false.

**static ArrayList< OsmWay > imrcp.geosrv.osm.OsmUtil.merge (ArrayList< OsmWay > oWays) throws IOException [static]**

Attempts to merge any two OsmWays in the list that share a start/end point that is not shared by any other **OsmWay**.

*Parameters*

<i>oWays</i>	contains the OsmWays to merge
--------------	-------------------------------

*Returns*

A list of the OsmWays after merging is complete.

*Exceptions*

<i>IOException</i>
--------------------

`static boolean imrcp.geosrv.osm.OsmUtil.merge (OsmWay oCur, OsmWay oCmp, int nMergeType, double dAngleThreshold) throws IOException [static]`

Attempts to merge the new roadway segments. The ways will not be merged if the angle formed between them is less than dAngleThreshold or if an invalid nMergeType is used. The reference of oCur will contain the nodes of the both segment and represents the merged roadway.

#### Parameters

<i>oCur</i>	roadway segment 1
<i>oCmp</i>	roadway segment 2
<i>nMergeType</i>	a value from 1 to 4: 1 = oCur start node is the same as oCmp start node 2 = oCur start node is the same as oCmp end node 3 = oCur end node is the same as oCmp start node 4 = oCur end node is the same as oCmp end node
<i>dAngleThreshold</i>	the angle that the angle formed between the two roadway segments must be greater than to be merged. This is used to avoid merging segments that have a sharp turn.

#### Returns

true if the segments are merged, otherwise false

#### Exceptions

<i>IOException</i>
--------------------

`static boolean imrcp.geosrv.osm.OsmUtil.nodesEqual (OsmNode o1, OsmNode o2, int nTol) [static]`

Determines if the nodes are "equal" by using their distance. If the distance between the two nodes is less than or equal to the given tolerance, the nodes are considered "equal".

#### Parameters

<i>o1</i>	node 1
<i>o2</i>	node 2
<i>nTol</i>	tolerance in decimal degrees scaled to 7 decimal places

#### Returns

true if the distance between the two nodes is less than or equal to the given tolerance.

`static ArrayList< OsmNode > imrcp.geosrv.osm.OsmUtil.removeZeroRefs (ArrayList< OsmNode > oNodes) [static]`

Creates a new list and adds all of the nodes in the given list that have at least one roadway segment in its reference list. A new list is created instead of removing nodes from the given list to aid in performance since the number of nodes can be a large number.

#### Parameters

<i>oNodes</i>	list of nodes to check for nodes with no roadway segments in their reference list.
---------------	--

#### Returns

a list contains all of the nodes in the list that had at least one roadway segment in their reference list.

```
static ArrayList< OsmWay > imrcp.geosrv.osm.OsmUtil.separate (ArrayList< OsmWay > oWays, ArrayList< OsmNode > oNodes, int nSepDist) throws IOException [static]
```

Attempts to separate bi directional roads into two one directional roads with the given distance from the original center line.

#### Parameters

<i>oWays</i>	List of roadway segments to process
<i>oNodes</i>	List of nodes that make up all the roadway segments. New nodes will be added to this list.
<i>nSepDist</i>	distance in decimal degrees scaled to 7 decimal places to place nodes of the newly separated roadway segments from the center line of the bi-directional road being separated.

#### Returns

A list containing the roadway segments after separating is finished.

#### Exceptions

<i>IOException</i>
--------------------

```
static ArrayList< OsmWay > imrcp.geosrv.osm.OsmUtil.split (ArrayList< OsmWay > oWays) throws Exception [static]
```

Splits the roadway segments in the given list if they have an inner node that is referenced by more than one way.

#### Parameters

<i>oWays</i>	list of roadway segments to split
--------------	-----------------------------------

#### Returns

A list of the OsmWays after splitting is complete.

#### Exceptions

<i>Exception</i>
------------------

```
static void imrcp.geosrv.osm.OsmUtil.writeLanesAndSpeeds (String sGeoFile, ArrayList< OsmWay > oWays, boolean bAppend) throws Exception [static]
```

Writes a metadata files for lanes and speed limits parsing the values from the OSM tags.

#### Parameters

<i>sGeoFile</i>	The path of the file defining the road network of the list of roadway segments
<i>oWays</i>	The roadway segments of the road network
<i>bAppend</i>	flag indicating if the metadata file should be appended to(true) or over written(false).

## Exceptions

<i>Exception</i>	
------------------	--

---

## Member Data Documentation

```
final String [] imrcp.geosrv.osm.OsmUtil.ALL_ROADS = {"motorway", "trunk",
"primary", "secondary", "tertiary", "motorway_link", "trunk_link", "primary_link",
"secondary_link", "tertiary_link", "unclassified", "residential"} [static]
```

Contains all of the "highway" tag values for OSM ways

---

*The documentation for this class was generated from the following file:*

- geosrv/osm/OsmUtil.java
- 

## imrcp.geosrv.osm.OsmWay Class Reference

### Public Member Functions

- **OsmWay ()**
- **OsmWay (int nNodeCap)**
- **OsmWay (DataInputStream oIn, ArrayList< String > oPool, ArrayList< OsmNode > oNodes, int[] nFp) throws IOException**
- **void generateId () throws IOException**
- **OsmWay (long lId)**
- **void setName ()**
- **void setBridge ()**
- **void updateRefs ()**
- **void removeRefs ()**
- **void calcMidpoint ()**
- **void setMinMax ()**
- **void appendLineGeoJson (StringBuilder sLineBuf, WayNetworks oWays)**
- **ArrayList< OsmWay > getFromWays ()**
- **ArrayList< OsmWay > getToWays ()**
- **WaySnapInfo snap (int nTol, int nX, int nY)**
- **int getDirection ()**
- **int getCurve ()**
- **WayIterator iterator ()**
- **void setHdgs ()**
- **Object getLengthInM ()**

### Public Attributes

- **ArrayList< OsmNode > m\_oNodes**
- **int m\_nMaxLat = Integer.MIN\_VALUE**
- **int m\_nMaxLon = Integer.MIN\_VALUE**
- **int m\_nMinLat = Integer.MAX\_VALUE**
- **int m\_nMinLon = Integer.MAX\_VALUE**
- **int m\_nMidLat = Integer.MIN\_VALUE**
- **int m\_nMidLon = Integer.MIN\_VALUE**
- **short m\_tElev = Short.MIN\_VALUE**
- **String m\_sName = ""**
- **boolean m\_bBridge = false**

- byte **m\_yLinkType** = NOTSET
- boolean **m\_bTraversed** = false
- boolean **m\_bSeparated** = false
- **OsmWay m\_oPosSep** = null
- **OsmWay m\_oNegSep** = null
- boolean **m\_bUseStart** = true
- **OsmNode m\_oOriginalStart** = null
- **OsmNode m\_oOriginalEnd** = null
- double **m\_dLength**

#### Static Public Attributes

- static final byte **NOTSET** = -1
  - static final byte **NOTRAMP** = 0
  - static final byte **RAMP** = 1
  - static final byte **CONNECTOR** = 2
  - static Comparator< **OsmWay** > **WAYBYTEID**
- 

#### Detailed Description

This class represents roadway segments in the system which were originally derived from Open Street Map Way objects.

#### Author

Federal Highway Administration

---

#### Constructor & Destructor Documentation

*imrcp.geosrv.osm.OsmWay.OsmWay ()*

Default constructor. Sets **m\_oNodes** to a new ArrayList

*imrcp.geosrv.osm.OsmWay.OsmWay (int nNodeCap)*

Constructs an **OsmWay** with the given capacity for **m\_oNodes**

#### Parameters

<i>nNodeCap</i>	initial capacity for the node list
-----------------	------------------------------------

*imrcp.geosrv.osm.OsmWay.OsmWay (DataInputStream oIn, ArrayList< String > oPool, ArrayList< OsmNode > oNodes, int[] nFp) throws IOException*

Constructs an **OsmWay** by reading its definition from the given DataInputStream which is wrapping an InputStream to an IMRCP OSM binary file.

#### Parameters

<i>oIn</i>	DataInputStream wrapping the InputStream of the IMRCP OSM binary file
<i>oPool</i>	list representing the StringPool of the file
<i>oNodes</i>	list of all the node in the IMRCP OSM binary file, they should be sorted by <b>OsmObject#FPCOMP</b>
<i>nFp</i>	current position in the file, gets updated as bytes are read

*Exceptions*

<i>IOException</i>
--------------------

*imrcp.geosrv.osm.OsmWay.OsmWay (long lId)*

Constructs an **OsmWay** with the given OSM long id

*Parameters*

<i>lId</i>
------------

**Member Function Documentation**

*void imrcp.geosrv.osm.OsmWay.appendLineGeoJson (StringBuilder sLineBuf, WayNetworks oWays)*

Appends a GeoJson Feature representation of this Way to the given StringBuilder.

*Parameters*

<i>sLineBuf</i>	buffer to add the GeoJson Feature to
<i>oWays</i>	Object that contains the metadata definitions for Ways

*void imrcp.geosrv.osm.OsmWay.calcMidpoint ()*

Finds the midpoint of the polyline that makes up this Way. The length of the Way is also set in this method.

*void imrcp.geosrv.osm.OsmWay.generateId () throws IOException*

Generates a unique Id for the Way by adding all of the nodes of the Way to a growable int array and calling **Id#Id(int, int[])**

*Exceptions*

<i>IOException</i>
--------------------

*int imrcp.geosrv.osm.OsmWay.getCurve ()*

Determines if the road is "curved". The algorithm checks if any node of the Way has a perpendicular distance to the line segment created from the first and last node that is greater than 10% of the length of that line segment.

*Returns*

0 = not curved, 1 = curved

*int imrcp.geosrv.osm.OsmWay.getDirection ()*

Returns a value representing the direction of travel of the roadway segment by calculating the heading going from the first node to the second.

*Returns*

1 = east, 2 = south, 3 = west, 4 = north

*ArrayList< OsmWay > imrcp.geosrv.osm.OsmWay.getFromWays ()*

Gets a list of all the Ways whose end node is this Way's start node

*Returns*

A list containing Ways whose end node is this Way's start node

*Object imrcp.geosrv.osm.OsmWay.getLengthInM ()*

Gets the length of the Way in meters using the Haversine formula.

*Returns*

The length in meters of the Way

*ArrayList< OsmWay > imrcp.geosrv.osm.OsmWay.getToWays ()*

Gets a list of all the Ways whose start node is this Way's end node

*Returns*

A list containing Ways whose start node is this Way's end node

*WayIterator imrcp.geosrv.osm.OsmWay.iterator ()*

Gets a **WayIterator** to iterate through the nodes of the Way

*Returns*

a **WayIterator** to iterate through the nodes of the Way

*void imrcp.geosrv.osm.OsmWay.removeRefs ()*

For each node that makes up this Way, remove this Way from its reference list,  
**OsmNode#m\_oRefs**

*void imrcp.geosrv.osm.OsmWay.setBridge ()*

Sets the bridge flag by reading the bridge tag from the OSM definition

*void imrcp.geosrv.osm.OsmWay.setHdgs ()*

Sets the **OsmNode#m\_dHdg** variable for each of the nodes in this Way.

*void imrcp.geosrv.osm.OsmWay.setMinMax ()*

Iterates through the nodes that make up this Way to determine the minimum and maximum latitude and longitudes

*void imrcp.geosrv.osm.OsmWay.setName ()*

Attempts to get the name of the road by checking the ref and name tags from the OSM definition

*WaySnapInfo imrcp.geosrv.osm.OsmWay.snap (int nTol, int nX, int nY)*

Attempts to snap the given point to this Way by trying to snap the point to each line segment of the polyline that defines the Way's geometry.

*Parameters*

<i>nTol</i>	tolerance used for perpendicular distance, the point must be at within this distance of the line segment it gets snapped to
<i>nX</i>	longitude in decimal degrees scaled to 7 decimal places of the point
<i>nY</i>	latitude in decimal degrees scaled to 7 decimal places of the point

### Returns

A **WaySnapInfo** containing the parameters calculated during the  
`GeoUtil#getPerpDist(int, int, int, int, int, int,  
imrcp.geosrv.WaySnapInfo)`

`void imrcp.geosrv.osm.OsmWay.updateRefs ()`

For each node that makes up this Way, ensures that the node contains this Way in its reference list, `OsmNode#m_oRefs`.

---

### Member Data Documentation

`final byte imrcp.geosrv.osm.OsmWay.CONNECTOR = 2 [static]`

Enumeration for `m_yLinkType` indicating this Way's type is a connector.

`boolean imrcp.geosrv.osm.OsmWay.m_bBridge = false`

Flag indicating if the road is a bridge

`boolean imrcp.geosrv.osm.OsmWay.m_bSeparated = false`

Flag indicating if this Way has been separated during the separate algorithm

`boolean imrcp.geosrv.osm.OsmWay.m_bTraversed = false`

Flag indicating if this Way has been traversed during the determine ramps algorithm

`boolean imrcp.geosrv.osm.OsmWay.m_bUseStart = true`

Flag indicating if the first node or end node should be used during parts of the separate algorithm

`double imrcp.geosrv.osm.OsmWay.m_dLength`

Length of the Way in decimal degrees

`int imrcp.geosrv.osm.OsmWay.m_nMaxLat = Integer.MIN_VALUE`

Maximum latitude in decimal degrees scaled to 7 decimal places

`int imrcp.geosrv.osm.OsmWay.m_nMaxLon = Integer.MIN_VALUE`

Maximum longitude in decimal degrees scaled to 7 decimal places

`int imrcp.geosrv.osm.OsmWay.m_nMidLat = Integer.MIN_VALUE`

Latitude in decimal degrees scaled to 7 decimal places of the midpoint of the Way

`int imrcp.geosrv.osm.OsmWay.m_nMidLon = Integer.MIN_VALUE`

Longitude in decimal degrees scaled to 7 decimal places of the midpoint of the Way

`int imrcp.geosrv.osm.OsmWay.m_nMinLat = Integer.MAX_VALUE`

Minimum latitude in decimal degrees scaled to 7 decimal places

`int imrcp.geosrv.osm.OsmWay.m_nMinLon = Integer.MAX_VALUE`

Minimum longitude in decimal degrees scaled to 7 decimal places

`OsmWay imrcp.geosrv.osm.OsmWay.m_oNegSep = null`

The Way generated by separating this Way in the negative tangent direction

*ArrayList<OsmNode> imrcp.geosrv.osm.OsmWay.m\_oNodes*

The list of Nodes that define the geometry and direction of this Way

*OsmNode imrcp.geosrv.osm.OsmWay.m\_oOriginalEnd = null*

Reference of the original end node of this Way, used during the separate algorithm

*OsmNode imrcp.geosrv.osm.OsmWay.m\_oOriginalStart = null*

Reference of the original start node of this Way, used during the separate algorithm

*OsmWay imrcp.geosrv.osm.OsmWay.m\_oPosSep = null*

The Way generated by separating this Way in the positive tangent direction

*String imrcp.geosrv.osm.OsmWay.m\_sName = ""*

Name of the road this Way represents

*short imrcp.geosrv.osm.OsmWay.m\_tElev = Short.MIN\_VALUE*

Elevation of the Way at it's midpoint in meters

*byte imrcp.geosrv.osm.OsmWay.m\_yLinkType = NOTSET*

Enumerated value that tells what type of ramp the Way is, used mainly for Ways with a highway tag ending in "\_link"

*final byte imrcp.geosrv.osm.OsmWay.NOTRAMP = 0 [static]*

Enumeration for **m\_yLinkType** indicating this Way's type is not a ramp

*final byte imrcp.geosrv.osm.OsmWay.NOTSET = -1 [static]*

Enumeration for **m\_yLinkType** indicating this Way's type has not been set

*final byte imrcp.geosrv.osm.OsmWay.RAMP = 1 [static]*

Enumeration for **m\_yLinkType** indicating this Way's type is a ramp

*Comparator<OsmWay> imrcp.geosrv.osm.OsmWay.WAYBYTEID [static]*

```
Initial value:= (OsmWay o1, OsmWay o2) ->
{
    return Id.COMPARATOR.compare(o1.m_oId, o2.m_oId);
}
```

Compares OsmWays by their Id

---

*The documentation for this class was generated from the following file:*

- geosrv/osm/OsmWay.java
- 

## imrcp.web.OutputCsv Class Reference

### Protected Attributes

- String **m\_sHeader** = "Source,**ObsType**,ObstimeStart, ObstimeEnd, Latitude 1, Longitude 1, Latitude 2, Longitude 2, Elevation, **Units**, Observation (Numeric), Observation (**Text**)"
- SimpleDateFormat **m\_oDateFormat** = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")

### Static Protected Attributes

- static DecimalFormat **m\_oDecimal** = new DecimalFormat("0.#")

## Additional Inherited Members

---

### Detailed Description

**OutputFormat** implementation for writing observations in the report CSV format

#### Author

Federal Highway Administration

---

### Member Data Documentation

*SimpleDateFormat imrcp.web.OutputCsv.m\_oDateFormat = new  
SimpleDateFormat("yyyy-MM-dd HH:mm:ss") [protected]*

Format object for timestamps

*DecimalFormat imrcp.web.OutputCsv.m\_oDecimal = new  
DecimalFormat("0.#") [static], [protected]*

Format object used for floating point numbers

*String imrcp.web.OutputCsv.m\_sHeader = "Source,ObsType,ObstimeStart,  
ObstimeEnd,Latitude 1,Longitude 1,Latitude 2,Longitude  
2,Elevation,Units,Observation (Numeric),Observation (Text)" [protected]*

CSV header

---

*The documentation for this class was generated from the following file:*

- web/OutputCsv.java
- 

## imrcp.web.OutputFormat Class Reference

### Public Member Functions

- int **compare** (**Obs** oSubObsL, **Obs** oSubObsR)

### Protected Member Functions

- **OutputFormat** ()

### Protected Attributes

- String **m\_sSuffix**
- 

### Detailed Description

Base class used to write the data files that fulfill subscriptions/reports. Child classes implement **fulfill(java.io.PrintWriter, java.util.List, imrcp.web.ReportSubscription)** to write the data in different formats.

#### Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

*imrcp.web.OutputFormat.OutputFormat () [protected]*

Default constructor. Does nothing

---

## Member Function Documentation

*int imrcp.web.OutputFormat.compare (Obs oSubObsL, Obs oSubObsR)*

C.compares Obs by start time, then contributor id, then observation type id, then end time

---

## Member Data Documentation

*String imrcp.web.OutputFormat.m\_sSuffix [protected]*

File extension of the data files created by this class

---

*The documentation for this class was generated from the following file:*

- web/OutputFormat.java
- 

## imrcp.comp.LAc2cDetectorsComp.Parser Class Reference

### Public Member Functions

- void **characters** (char[] cBuf, int nPos, int nLen)
- void **endElement** (String sUri, String sLocalName, String sQname)

### Protected Member Functions

- **Parser** (HashMap< String, int[]> oObsMap)
- void **parse** (InputStream oIn) throws Exception

### Protected Attributes

- StringBuilder **m\_sBuf** = new StringBuilder()
- 

### Detailed Description

XML parser for LAc2c Detector files

---

## Constructor & Destructor Documentation

*imrcp.comp.LAc2cDetectorsComp.Parser.Parser (HashMap< String, int[]> oObsMap) [protected]*

Creates a new **Parser** with the given HashMap

### Parameters

<i>oObsMap</i>	HashMap to fill observations with
----------------	-----------------------------------

---

## Member Function Documentation

`void imrcp.comp.LAc2cDetectorsComp.Parser.parse (InputStream oIn) throws Exception [protected]`

Wrapper for `XMLReader#parse (org.xml.sax.InputSource)`

*Parameters*

<code>oIn</code>	InputStream of xml file
------------------	-------------------------

*Exceptions*

<code>Exception</code>
------------------------

---

## Member Data Documentation

`StringBuilder imrcp.comp.LAc2cDetectorsComp.Parser.m_sBuf = new StringBuilder() [protected]`

Buffer to store characters in when `DefaultHandler2#characters (char[], int, int)` is called

---

*The documentation for this class was generated from the following file:*

- `comp/LAc2cDetectorsComp.java`
- 

## imrcp.comp.PcCat Class Reference

### Public Member Functions

- `void reset ()`
- `boolean start () throws Exception`
- `void process (String[] sMessage)`
- `void queue (String sStart, String sEnd, StringBuilder sBuffer)`
- `void queueStatus (StringBuilder sBuffer)`
- `void execute ()`
- `boolean stop () throws Exception`

### Protected Attributes

- `int[] m_nObsTypes`
- `String[] m_sObsTypes`
- `String m_sHz`
- `String m_sVrt`
- `String m_sTime`
- `String m_sGfsHz`
- `String m_sGfsVrt`
- `String m_sGfsTime`
- `int[] m_nNdfdTempTypes`
- `int[] m_nNdfdQpfTypes`
- `String[] m_sNdfdTempTypes`
- `String[] m_sNdfdQpfTypes`

## Additional Inherited Members

---

### Detailed Description

Contains methods for creating Precipitation Category observations from different National Weather Service sources.

### Author

Federal Highway Administration

---

### Member Function Documentation

#### `void imrcp.comp.PcCat.execute ()`

Processes up to **PcCat#m\_nRunsPerPeriod** files in the queue.

Reimplemented from **imrcp.system.BaseBlock** (p.95).

#### `void imrcp.comp.PcCat.process (String[] sMessage)`

Called when a message from **imrcp.system.Directory** is received. Determines which method to call depending on which base block sent the message.

##### Parameters

<code>sMessage</code>	[BaseBlock message is from, message name, files to process...]
-----------------------	--

Reimplemented from **imrcp.system.BaseBlock** (p.97).

#### `void imrcp.comp.PcCat.queue (String sStart, String sEnd, StringBuilder sBuffer)`

If this instance is configured to reprocess files, it parses the given start and end times and queues files within that range to be reprocessed. The given StringBuilder gets filled with basic html that contains the files that were queued.

##### Parameters

<code>sStart</code>	start timestamp in the format yyyy-MM-ddTHH:mm
<code>sEnd</code>	end timestamp in the format yyyy-MM-ddTHH:mm
<code>sBuffer</code>	Buffer that gets fills with html containing the queued files

#### `void imrcp.comp.PcCat.queueStatus (StringBuilder sBuffer)`

Adds the current files in the queue to the StringBuilder in basic html

##### Parameters

<code>sBuffer</code>	StringBuilder to fill with current files queued
----------------------	---

#### `void imrcp.comp.PcCat.reset ()`

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.system.BaseBlock** (p.98).

#### `boolean imrcp.comp.PcCat.start () throws Exception`

Checks if the configured queue file has any file names in it and them to the queue. Then if the period is not 0 sets a schedule to execute on a fixed interval.

*Returns*

true if no exceptions are thrown

*Exceptions*

<i>Exception</i>	
------------------	--

Reimplemented from **imrcp.system.BaseBlock** (p.99).

*boolean imrcp.comp.PcCat.stop () throws Exception*

Deletes partial files that have not finished processing

*Returns*

true if no exceptions are thrown

*Exceptions*

<i>Exception</i>	
------------------	--

Reimplemented from **imrcp.system.BaseBlock** (p.100).

---

## Member Data Documentation

*int [] imrcp.comp.PcCat.m\_nNdfdQpfTypes [protected]*

Contains the IMRCP observation types that map to the observation type labels in **PcCat#m\_sNdfdQpfTypes**'s corresponding indices. Used in constructing **imrcp.store.NcfWrapper**s

*int [] imrcp.comp.PcCat.m\_nNdfdTempTypes [protected]*

Contains the IMRCP observation types that map to the observation type labels in **PcCat#m\_sNdfdTempTypes**'s corresponding indices. Used in constructing **imrcp.store.NcfWrapper**s

*int [] imrcp.comp.PcCat.m\_nObsTypes [protected]*

Contains the IMRCP observation types that map to the observation type labels in **PcCat#m\_sObsTypes**'s corresponding indices. Used in constructing **imrcp.store.NcfWrapper**s

*String imrcp.comp.PcCat.m\_sGfsHz [protected]*

Label of the x coordinate in GFS files

*String imrcp.comp.PcCat.m\_sGfsTime [protected]*

Label of the time axis in GFS files

*String imrcp.comp.PcCat.m\_sGfsVrt [protected]*

Label of the y coordinate in GFS files

*String imrcp.comp.PcCat.m\_sHz [protected]*

Label of the x coordinate in NWS files using Lambert Conformal projections

*String [] imrcp.comp.PcCat.m\_sNdfdQpfTypes [protected]*

Contains the observation type labels found in NDFD quantitative precipitation forecast files. Used in constructing **imrcp.store.NcfWrapper**s

*String [] imrcp.comp.PcCat.m\_sNdfdTempTypes [protected]*

Contains the observation type labels found in NDFD temperature files. Used in constructing **imrcp.store.NcfWrapper**s

*String [] imrcp.comp.PcCat.m\_sObsTypes [protected]*

Contains the observation type labels found in NWS files. Used in constructing **imrcp.store.NcfWrapper**s

*String imrcp.comp.PcCat.m\_sTime [protected]*

Label of the time axis in NWS files using Lambert Conformal projections

*String imrcp.comp.PcCat.m\_sVrt [protected]*

Label of the y coordinate in NWS files using Lambert Conformal projections

---

*The documentation for this class was generated from the following file:*

- comp/PcCat.java
- 

## imrcp.store.grib.PngInStream Class Reference

### Public Member Functions

- **PngInStream** (InputStream oIn) throws IOException
- **PngInStream** (DataInputStream oIn) throws IOException
- int **read** (byte[] yBuf, int nOff, int nLen) throws IOException
- int **read** () throws IOException
- int **finish** () throws IOException

### Static Public Attributes

- static final int **IDAT** = 0x49444154
  - static final int **IEND** = 0x49454e44
  - static final int **IHDR** = 0x49484452
  - static final long **PNGHDR** = -8552249625308161526L
- 

### Detailed Description

FilterInputStream used to parse PNG datastreams

#### Author

Federal Highway Administration

---

### Constructor & Destructor Documentation

*imrcp.store.grib.PngInStream.PngInStream (InputStream oIn) throws IOException*

Calls **PngInStream(java.io.DataInputStream)** by wrapping the given InputStream with a DataInputStream.

#### Parameters

<i>oIn</i>	InputStream representing the start of a PNG datastream
------------	--

*Exceptions*

<i>IOException</i>
--------------------

*imrcp.store.grib.PngInStream.PngInStream (DataInputStream oIn) throws IOException*

Constructs a new **PngInStream** with the given DataInputStream. This function prepares the **PngInStream** to be ready to read the image line by line. To do this the IHDR section is read and then the IDAT section is read into **m\_yIDAT** and set as the input of **m\_oInf**

*Parameters*

<i>oIn</i>	DataInputStream wrapping a PNG datastream
------------	---

*Exceptions*

<i>IOException</i>
--------------------

---

## Member Function Documentation

*int imrcp.store.grib.PngInStream.finish () throws IOException*

Reads to the end of the PNG datastream.

*Returns*

0

*Exceptions*

<i>IOException</i>
--------------------

*int imrcp.store.grib.PngInStream.read () throws IOException*

Wrapper for **PngInStream#read(byte[], int, int)** passing **m\_yInfBuf** as the buffer meaning 1 byte from **m\_oInf** is read into **m\_yInfBuf**

*int imrcp.store.grib.PngInStream.read (byte[] yBuf, int nOff, int nLen) throws IOException*

Reads bytes from **m\_oInf** into yBuf

---

## Member Data Documentation

*final int imrcp.store.grib.PngInStream.IDAT = 0x49444154 [static]*

Marker for IDAT section

*final int imrcp.store.grib.PngInStream.IEND = 0x49454e44 [static]*

Marker for IEND section

*final int imrcp.store.grib.PngInStream.IHDR = 0x49484452 [static]*

Marker for IHDR section

*final long imrcp.store.grib.PngInStream.PNGHDR = -8552249625308161526L [static]*

The first eight bytes of a PNG datastream

---

*The documentation for this class was generated from the following file:*

- store/grib/PngInStream.java
- 

## imrcp.system.shp.Point Class Reference

### Public Member Functions

- **Point** (double dX, double dY)
- **Point** (DataInputStream oDataInputStream) throws Exception
- int **compareTo** (**Point** o)

### Public Attributes

- double **m\_dX**
  - double **m\_dY**
- 

### Constructor & Destructor Documentation

*imrcp.system.shp.Point.Point (double dX, double dY)*

#### Parameters

<i>dX</i>	
<i>dY</i>	

*imrcp.system.shp.Point.Point (DataInputStream oDataInputStream) throws Exception*

#### Parameters

<i>oDataInputStream</i>	
-------------------------	--

#### Exceptions

<i>Exception</i>	
------------------	--

---

*The documentation for this class was generated from the following file:*

- system/shp/Point.java
- 

## imrcp.web.layers.PointsLayerServlet Class Reference

### Protected Member Functions

- void **buildObsResponseContent** (JsonGenerator oOutputGenerator, **ObsRequest** oObsRequest) throws Exception

- void **buildObsChartResponseContent** (JsonGenerator oOutputGenerator, **ObsChartRequest** oObsRequest) throws Exception

### Additional Inherited Members

---

#### Detailed Description

Handles requests from the IMRCP Map UI when Points(sensors, CVs, alerts) layer objects are clicked or when a chart for Point observations is requested

#### Author

Federal Highway Administration

---

#### Member Function Documentation

*void imrcp.web.layers.PointsLayerServlet.buildObsChartResponseContent  
(JsonGenerator oOutputGenerator, ObsChartRequest oObsRequest) throws  
Exception [protected]*

Add the response to the given JSON stream for requests made from the IMRCP Map UI to create a chart for points observations

Reimplemented from **imrcp.web.layers.LayerServlet** (*p.322*).

*void imrcp.web.layers.PointsLayerServlet.buildObsResponseContent (JsonGenerator  
oOutputGenerator, ObsRequest oObsRequest) throws Exception [protected]*

Add the response to the given JSON stream for requests made from the IMRCP Map UI when a Points Layer object is clicked.

Reimplemented from **imrcp.web.layers.LayerServlet** (*p.323*).

---

*The documentation for this class was generated from the following file:*

- web/layers/PointsLayerServlet.java
- 

## imrcp.system.shp.Polyline Class Reference

#### Public Member Functions

- **Polyline** (DataInputStream oDataInputStream, boolean bMicroDegrees) throws Exception
- boolean **contextSearch** (double dMaxDistance, int nX, int nY)

#### Protected Member Functions

- **Polyline** ()

### Additional Inherited Members

---

#### Detailed Description

Holds information associated with a geo-coordinate polyline. A polyline can be used to represent roads, rivers, rail lines, or other linear map features.

*Author*

Federal Highway Administration

*Version*

1.0 (April 27, 2007)

---

**Constructor & Destructor Documentation**

*imrcp.system.shp.Polyline () [protected]*

Creates a new "blank" instance of **Polyline**

*imrcp.system.shp.Polyline (DataInputStream oDataStream, boolean bMicroDegrees) throws Exception*

Creates a new instance of **Polyline** with name, identifier, data input, and math transform specified.

*Parameters*

<i>bMicroDegrees</i>	
<i>oDataStream</i>	data stream containing information used to build the polyline object

*Exceptions*

<i>java.lang.Exception</i>	
----------------------------	--

---

**Member Function Documentation**

*boolean imrcp.system.shp.Polyline.contextSearch (double dMaxDistance, int nX, int nY)*

Determines if the specified coordinate is within the specified distance of the polyline.

*Parameters*

<i>dMaxDistance</i>	maximum distance for point to be considered "in" the polyline
<i>nX</i>	longitudinal coordinate
<i>nY</i>	latitudinal coordinate

*Returns*

true if the point is "in" the polyline, false otherwise

Reimplemented from **imrcp.system.shp.Polyshape** (p.456).

---

*The documentation for this class was generated from the following file:*

- system/shp/Polyline.java
-

## imrcp.system.shp.Polyshape Class Reference

### Public Member Functions

- int[] **getPoints** ()
- **PolyshapeIterator iterator (PolyshapeIterator oShapeIter)**
- abstract boolean **contextSearch** (double dMaxDistance, int nX, int nY)
- boolean **isInsideBounds** (int nLat, int nLon, int nTol)
- int **compareTo (Polyshape oRhs)**
- void **printPoints** (PrintWriter oPrinter)

### Static Public Member Functions

- static int **toIntDegrees** (double dValue)
- static double **fromIntDegrees** (int nValue)

### Public Attributes

- final int **m\_nXmin**
- final int **m\_nYmin**
- final int **m\_nXmax**
- final int **m\_nYmax**

### Protected Member Functions

- **Polyshape ()**
- **Polyshape (DataInputStream oDataInputStream, boolean bMicroDegrees)** throws Exception

### Static Protected Member Functions

- static void **encodeSignedNumber** (PrintWriter oPrinter, int nNumber)

### Protected Attributes

- int[] **m\_nParts**
- int[] **m\_nPoints**

---

### Detailed Description

Holds information associated with a geo-coordinate polyshape. A polyshape can be used to represent roads, rivers, rail lines, city boundaries, state boundaries, or any other 2-D map shape. A polyshape is defined by "parts" and geo-coordinate points. Each "part" is composed of at least one set of longitude and latitude coordinates. The points are broken into parts in order to allow for discontiguous shapes like dotted lines or "donuts".

In the array of "parts", each entry points to the index of the points array where the part begins. The first entry in the parts array is always 0.

#### Author

Federal Highway Administration

#### Version

1.0 (April 27, 2007)

---

### Constructor & Destructor Documentation

#### *imrcp.system.shp.Polyshape.Polyshape () [protected]*

Creates a new "blank" instance of **Polyshape**.

*imrcp.system.shp.Polyshape.Polyshape (DataInputStream oDataStream, boolean bMicroDegrees) throws Exception [protected]*

Creates a new instance of **Polyshape** with name, data input, and math transform specified.

*Parameters*

<i>oDataStream</i>	data stream containing information used to build the polyshape object
<i>bMicroDegrees</i>	tells whether or not to store the coordinates in micro degrees or not

*Exceptions*

<i>java.lang.Exception</i>	
----------------------------	--

---

## Member Function Documentation

*int imrcp.system.shp.Polyshape.compareTo (Polyshape oRhs)*

Compares this polysyape to the specified polyshape for order by minimum longitude. Returns a negative integer, zero, or a positive integer as this object's minimum longitude is less than, equal to, or greater than the specified object's minimum longitude.

*Parameters*

<i>oRhs</i>	the polyshape object to be compared.
-------------	--------------------------------------

*Returns*

a negative integer, zero, or a positive integer as this object's minimum longitude is less than, equal to, or greater than the specified object's minimum longitude.

*abstract boolean imrcp.system.shp.Polyshape.contextSearch (double dMaxDistance, int nX, int nY) [abstract]*

Abstract class to determine if the specified coordinate is within the specified distance of a polyshape.

*Parameters*

<i>dMaxDistance</i>	maximum distance for point to be considered "in" the polyshape
<i>nX</i>	longitudinal coordinate
<i>nY</i>	latitudinal coordinate

*Returns*

true if the point is "in" the polyshape, false otherwise

Reimplemented in **imrcp.system.shp.Polyline** (*p.454*).

*static void imrcp.system.shp.Polyshape.encodeSignedNumber (PrintWriter oPrintWriter, int nNumber) [static], [protected]*

Encodes the specified integer and prints it to the specified PrintWriter.

*Parameters*

<i>oPrintWriter</i>	output destination
---------------------	--------------------

<i>nNumber</i>	integer to encode
----------------	-------------------

**static double imrcp.system.shp.Polyshape.fromIntDegrees (int nValue) [static]**

Converts the specified scaled integer to a double precision number.

#### Parameters

<i>nValue</i>	the scaled integer to convert
---------------	-------------------------------

#### Returns

double precision number

**boolean imrcp.system.shp.Polyshape.isInsideBounds (int nLat, int nLon, int nTol)**

Determines if the specified point is within the bounds of this polyshape. This method does not take into account any padding distance, the point must either be within the actual area enclosed by polyshape or on the polyshape itself.

#### Parameters

<i>nLat</i>	latitudinal coordinate
<i>nLon</i>	longitudinal coordinate
<i>nTol</i>	

#### Returns

ture if the point is strictly in the polyshape, false otherwise

**PolyshapeIterator imrcp.system.shp.Polyshape.iterator (PolyshapeIterator oShapeIter)**

Initializes an iterator for this polyshape.

#### Parameters

<i>oShapeIter</i>	a <b>PolyshapeIterator</b> object to initialize
-------------------	---

#### Returns

the initialized iterator for this polyshape

**void imrcp.system.shp.Polyshape.printPoints (PrintWriter oPrinter)**

Prints the points that define this polyshape in JSON form to the specified PrintWriter.

#### Parameters

<i>oPrinter</i>	output destination
-----------------	--------------------

**static int imrcp.system.shp.Polyshape.toIntDegrees (double dValue) [static]**

Converts the specified double precision number to an integer scaled to seven decimal places.

#### Parameters

<i>dValue</i>	the number to convert
---------------	-----------------------

#### Returns

integer scaled to seven decimal places

---

## Member Data Documentation

*int [] imrcp.system.shp.Polyshape.m\_nParts [protected]*

Array of polyshape "parts".

*int [] imrcp.system.shp.Polyshape.m\_nPoints [protected]*

Array of polyshape latitude & longitude coordinates.

*final int imrcp.system.shp.Polyshape.m\_nXmax*

Maximum polyshape longitude.

*final int imrcp.system.shp.Polyshape.m\_nXmin*

Minimum polyshape longitude.

*final int imrcp.system.shp.Polyshape.m\_nYmax*

Maximum polyshape latitude.

*final int imrcp.system.shp.Polyshape.m\_nYmin*

Minimum polyshape latitude.

---

*The documentation for this class was generated from the following file:*

- system/shp/Polyshape.java
- 

---

## imrcp.system.shp.Polyshapelterator Class Reference

### Public Member Functions

- boolean **nextPart ()**
  - boolean **nextPoint ()**
  - int **getX ()**
  - int **getY ()**
- 

### Detailed Description

Provides a standard iterator for polyshape objects. A polyshape is defined by "parts" and geo-coordinate points. Each "part" is composed of at least one set of longitude and latitude coordinates. The points are broken into parts in order to allow for discontiguous shapes like dotted lines or "donuts". For more information on the architecture of polyshape objects, see the **Polyshape** documentation.

#### *Author*

Federal Highway Administration

#### *Version*

1.0 (August 1, 2008)

## Member Function Documentation

*int imrcp.system.shp.PolyshapeIterator.getX ()*

Returns the longitude of the current coordinate.

*Returns*

integer longitude scaled to six decimal places

*int imrcp.system.shp.PolyshapeIterator.getY ()*

Returns the latitude of the current coordinate.

*Returns*

integer latitude scaled to six decimal places

*boolean imrcp.system.shp.PolyshapeIterator.nextPart ()*

Moves this iterator to then next part of the polyshape.

*Returns*

true if there is a next part, false otherwise

*boolean imrcp.system.shp.PolyshapeIterator.nextPoint ()*

Moves this iterator to the next point of the current part of the polyshape.

*Returns*

true if there is a next point, false otherwise

---

*The documentation for this class was generated from the following file:*

- system/shp/PolyshapeIterator.java
- 

## imrcp.forecast.mlp.Prediction Class Reference

### Public Member Functions

- **Prediction ()**
  - **Prediction (Id oId, double[] dVals)**
  - **Prediction (Id oId, double[] dVals, double[] dOneshot)**
  - **int compareTo (Prediction o)**
- 

### Detailed Description

Stores the predicted speed values from MLP model runs.

#### Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

*imrcp.forecast.mlp.Prediction.Prediction ()*

Default constructor. Does nothing.

*imrcp.forecast.mlp.Prediction.Prediction (Id old, double[] dVals)*

Constructs a **Prediction** with the given Id and online prediction values

### Parameters

<i>oId</i>	IMRCP id of segment the speed predictions are applied to
<i>dVals</i>	speed predictions in mph from online prediction model

*imrcp.forecast.mlp.Prediction.Prediction (Id old, double[] dVals, double[] dOneshot)*

Constructs a **Prediction** with the given Id, online prediction values, and oneshot prediction values

### Parameters

<i>oId</i>	IMRCP id of segment the speed predictions are applied to
<i>dVals</i>	speed predictions in mph from online prediction model
<i>dOneshot</i>	speed predictions in mph from oneshot model

---

## Member Function Documentation

*int imrcp.forecast.mlp.Prediction.compareTo (Prediction o)*

Compares Predictions by Id

### See also

java.lang.Comparable::compareTo(java.lang.Object)

---

*The documentation for this class was generated from the following file:*

- forecast/mlp/Prediction.java
- 

## imrcp.store.grib.Projection Class Reference

### Static Public Member Functions

- static **Projection newProjection** (DataInputStream oIn, int nSecLen) throws IOException

### Public Attributes

- int **m\_nX**
- int **m\_nY**
- int **m\_nScanningMode**
- int **m\_nTemplate**

---

### Detailed Description

Base class for reading Section 3 - **Grid** Definition Section of GRIB2 files.

See

[https://www.nco.ncep.noaa.gov/pmb/docs/grib2/grib2\\_doc/grib2\\_section3.shtml](https://www.nco.ncep.noaa.gov/pmb/docs/grib2/grib2_doc/grib2_section3.shtml)

*Author*

Federal Highway Administration

---

**Member Function Documentation**

*static Projection imrcp.store.grib.Projection.newProjection (DataInputStream oIn, int nSecLen) throws IOException [static]*

Constructs a new **Projection** object based off the information found in Section 3 of a GRIB2 file.

*Parameters*

<i>oIn</i>	DataInputStream of the GRIB2 file. It should be ready to read the 6th byte of Section 3 (meaning the length (4 bytes) of the section and section number (1 byte) has been read)
<i>nSecLen</i>	length of Section 3 in bytes

*Returns*

a new **Projection**, its type depending on the template number. Right now there are 2 implemented: 0 = **LatLonProj** and 30 = **LambertConformalProj**. Any other template will return null

*Exceptions*

<i>IOException</i>
--------------------

---

**Member Data Documentation**

*int imrcp.store.grib.Projection.m\_nScanningMode*

Scanning mode. See Flag Table 3.4 of NCEP GRIB2 documentation

*int imrcp.store.grib.Projection.m\_nTemplate*

Template number, tells what kind of projected coordinate system is used. See GRIB2 - TABLE 3.1

*int imrcp.store.grib.Projection.m\_nX*

Number of points along the x axis

*int imrcp.store.grib.Projection.m\_nY*

Number of points along the y axis

---

*The documentation for this class was generated from the following file:*

- store/grib/Projection.java

## [imrcp.store.ProjProfile Class Reference](#)

### [Public Member Functions](#)

- **ProjProfile ()**
- **ProjProfile (double[] dX, double[] dY)**
- **void initGrid (double[] dX, double[] dY, ProjectionImpl oProj)**
- **void getCell (int nHz, int nVrt, double[] dCorners)**
- **void getPointIndices (double dLon, double dLat, int[] nIndices)**
- **void getIndices (double dLon1, double dLat1, double dLon2, double dLat2, int[] nIndices)**
- **int compareTo (ProjProfile o)**

### [Public Attributes](#)

- int **m\_nVrt**
- int **m\_nHz**
- double **m\_dX1**
- double **m\_dX2**
- double **m\_dY1**
- double **m\_dY2**
- double[] **m\_dXs**
- double[] **m\_dYs**
- double[][] **m\_dGrid**
- boolean **m\_bUseReverseY**
- ProjectionImpl **m\_oProj**

### [Static Public Attributes](#)

- static int **xTL** = 0
- static int **yTL** = 1
- static int **xTR** = 2
- static int **yTR** = 3
- static int **xBR** = 4
- static int **yBR** = 5
- static int **xBL** = 6
- static int **yBL** = 7

---

### [Detailed Description](#)

Represents a Projected Coordinate System that can be used by other system components. The main reason this class was created was to lower the memory footprint by having multiple files that use the same Projected Coordinate System can all access a single object instead of having each file create its own Projected Coordinate System.

#### *Author*

Federal Highway Administration

---

### [Constructor & Destructor Documentation](#)

#### [\*imrcp.store.ProjProfile.ProjProfile \(\)\*](#)

Default constructor. Does nothing.

#### [\*imrcp.store.ProjProfile.ProjProfile \(double\[\] dX, double\[\] dY\)\*](#)

Constructs a new **ProjProfile** with the given x and y coordinate grids

*Parameters*

<i>dX</i>	the values of the Projected Coordinate System's x axis
<i>dY</i>	the values of the Projected Coordinate System's y axis

---

**Member Function Documentation**

*int imrcp.store.ProjProfile.compareTo (ProjProfile o)*

C.compares ProjProfiles by number of rows, number of columns, first x value, last x value, first y value, last y value.

*void imrcp.store.ProjProfile.getCell (int nHrz, int nVrt, double[] dCorners)*

Fills the given double array with the lon/lat coordinates of the corners of the cell of the grid at the given horizontal and vertical indices

*Parameters*

<i>nHrz</i>	horizontal index
<i>nVrt</i>	vertical index
<i>dCorners</i>	double array with length of 8 to be filled the geo coordinates of the corners of the cell.

*void imrcp.store.ProjProfile.getIndices (double dLon1, double dLat1, double dLon2, double dLat2, int[] nIndices)*

Fills the given int array with the indices of the grid that correspond to the given lon/lat points.

*Parameters*

<i>dLon1</i>	first point longitude in decimal degrees
<i>dLat1</i>	first point latitude in decimal degrees
<i>dLon2</i>	second point longitude in decimal degrees
<i>dLat2</i>	second point latitude in decimal degrees
<i>nIndices</i>	array of length 4 that gets filled with the indices of the grid corresponding to the lon/lat points. [min x index, max y index, max x index, min y index]

*void imrcp.store.ProjProfile.getPointIndices (double dLon, double dLat, int[] nIndices)*

*Parameters*

<i>dLon</i>	
<i>dLat</i>	
<i>nIndices</i>	

*void imrcp.store.ProjProfile.initGrid (double[] dX, double[] dY, ProjectionImpl oProj)*

Creates the grid for the given x and y axis for the Projected Coordinate System.

*Parameters*

<i>dX</i>	the values of the Projected Coordinate System's x axis
<i>dY</i>	the values of the Projected Coordinate System's y axis

<i>oProj</i>	Object that converts points from lon/lat to the Projected Coordinate System and vice versa
--------------	--

---

## Member Data Documentation

*boolean imrcp.store.ProjProfile.m\_bUseReverseY*

Flag indicated if the y axis is in ascending or descending order. true = descending, false = ascending

*double [][] imrcp.store.ProjProfile.m\_dGrid*

Stores the longitude and latitudes of the coordinate system. Each row has the format [lon0, lat0, lon1, lat1,... lonn, latn]

*double imrcp.store.ProjProfile.m\_dX1*

First x value

*double imrcp.store.ProjProfile.m\_dX2*

Last x value

*double [] imrcp.store.ProjProfile.m\_dXs*

Values of the x axis

*double imrcp.store.ProjProfile.m\_dY1*

First y value

*double imrcp.store.ProjProfile.m\_dY2*

Last y value

*double [] imrcp.store.ProjProfile.m\_dYs*

Values of the y axis

*int imrcp.store.ProjProfile.m\_nHz*

Number of columns (x values) in the grid

*int imrcp.store.ProjProfile.m\_nVrt*

Number of rows (y values) in the grid

*ProjectionImpl imrcp.store.ProjProfile.m\_oProj*

Object that converts points from lon/lat to the Projected Coordinate System and vice versa

*int imrcp.store.ProjProfile.xBL = 6 [static]*

Index used in arrays that represent the corners of a cell for the bottom left x

*int imrcp.store.ProjProfile.xBR = 4 [static]*

Index used in arrays that represent the corners of a cell for the bottom right x

*int imrcp.store.ProjProfile.xTL = 0 [static]*

Index used in arrays that represent the corners of a cell for the top left x

*int imrcp.store.ProjProfile.xTR = 2 [static]*

Index used in arrays that represent the corners of a cell for the top right x

*int imrcp.store.ProjProfile.yBL = 7 [static]*

Index used in arrays that represent the corners of a cell for the bottom left y

*int imrcp.store.ProjProfile.yBR = 5 [static]*

Index used in arrays that represent the corners of a cell for the bottom right y

*int imrcp.store.ProjProfile.yTL = 1 [static]*

Index used in arrays that represent the corners of a cell for the top left y

*int imrcp.store.ProjProfile.yTR = 3 [static]*

Index used in arrays that represent the corners of a cell for the top right y

---

*The documentation for this class was generated from the following file:*

- store/ProjProfile.java
- 

## imrcp.store.ProjProfiles Class Reference

### Public Member Functions

- **ProjProfile newProfile** (double[] dX, double[] dY, ProjectionImpl oProj, int nContrib)
- **ProjProfile getProfile** (int nHz, int nVrt, double dX1, double dY1, double dX2, double dY2)
- **ProjProfile getProfile** (int nContrib)

### Static Public Member Functions

- static **ProjProfiles getInstance** ()
- 

### Detailed Description

A singleton class that manages the **imrcp.store.ProjProfile**s available in the system.

#### Author

Federal Highway Administration

---

### Member Function Documentation

**static ProjProfiles imrcp.store.ProjProfiles.getInstance () [static]**

Gets the reference to the singleton instance

#### Returns

Reference to the singleton instance

**ProjProfile imrcp.store.ProjProfiles.getProfile (int nContrib)**

Gets the **ProjProfile** associated with the given contributor id.

#### Parameters

<i>nContrib</i>	IMRCP contributor Id
-----------------	----------------------

#### Returns

The **ProjProfile** for the given contributor id or null if one does not exist.

*ProjProfile imrcp.store.ProjProfiles.getProfile (int nHz, int nVrt, double dX1, double dY1, double dX2, double dY2)*

Gets the **ProjProfile** that has the given parameters.

#### Parameters

<i>nHz</i>	number of columns in the grid
<i>nVrt</i>	number of rows in the grid
<i>dX1</i>	first x value of x axis
<i>dY1</i>	last x value of x axis
<i>dX2</i>	first y value of y axis
<i>dY2</i>	last y value of y axis

#### Returns

*ProjProfile imrcp.store.ProjProfiles.newProfile (double[] dX, double[] dY, ProjectionImpl oProj, int nContrib)*

Searches the list **PROFILES** for a **ProjProfile** with the given parameters. If one does not exist, it is created and added to the list. Then the **ProjProfile** that matches the given parameters in the list is returned.

#### Parameters

<i>dX</i>	values of the x axis
<i>dY</i>	values of the y axis
<i>oProj</i>	Object that converts points from lon/lat to the Projected Coordinate System and vice versa
<i>nContrib</i>	Contributor id that uses the given Projected Coordinate System.

#### Returns

The **ProjProfile** that matches the given parameters that is in **PROFILES**

---

*The documentation for this class was generated from the following file:*

- store/ProjProfiles.java
- 

## imrcp.geosrv.RangeRules Class Reference

### Public Member Functions

- **RangeRules** (String sObsType)
- **RangeRules** (int nObsType)
- boolean **shouldDelete** (double dGroupVal)
- double **groupValue** (double dVal)

## Public Attributes

- double **m\_dNaNMapping**
  - double[] **m\_dRanges**
  - double[] **m\_dDeleteRanges**
  - int **m\_nObsType**
- 

## Detailed Description

Class used to bucket ranges of values of a specific observation type to a single value to aid in presentation and grouping similar observations.

### Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

### *imrcp.geosrv.RangeRules.RangeRules (String sObsType)*

Wrapper for **RangeRules (int)** . It converts the given observation type to its integer representation.

#### Parameters

<i>sObsType</i>	up to 6 alphanumeric characters representing an observation type
-----------------	--

### *imrcp.geosrv.RangeRules.RangeRules (int nObsType)*

Constructs a **RangeRules** object for the given observation type by reading values from the Configuration component.

#### Parameters

<i>nObsType</i>	IMRCP observation type
-----------------	------------------------

## Member Function Documentation

### *double imrcp.geosrv.RangeRules.groupValue (double dVal)*

Gets the group/bucket value for the given value.

#### Parameters

<i>dVal</i>	value to group
-------------	----------------

#### Returns

The lower value of the range/group the value falls in or the nan mapping if the given value is Double.NaN or not within one of the configured ranges.

### *boolean imrcp.geosrv.RangeRules.shouldDelete (double dGroupVal)*

Determines if the given group value should be deleted/ignored based on the configured ranges.

#### Parameters

<i>dGroupVal</i>	group value generated by <b>groupValue (double)</b>
------------------	---

### Returns

true if the value is equal to the nan mapping, is Double.NaN, or within one of the delete ranges, otherwise false.

---

## Member Data Documentation

### `double [] imrcp.geosrv.RangeRules.m_dDeleteRanges`

Array that stores the ranges that should be ignored or deleted in pairs [min1, max1, min2, max2,... minn, maxn]

### `double imrcp.geosrv.RangeRules.m_dNaNMapping`

Double value that corresponds to an invalid numerical value

### `double [] imrcp.geosrv.RangeRules.m_dRanges`

Array that stores the ranges in pairs [min1, max1, min2, max2,... minn, maxn]

### `int imrcp.geosrv.RangeRules.m_nObsType`

IMRCP observation type the rules are valid for.

---

*The documentation for this class was generated from the following file:*

- geosrv/RangeRules.java
- 

## imrcp.store.RapNcfWrapper Class Reference

### Public Member Functions

- **RapNcfWrapper** (int[] nObsTypes, String[] sObsTypes, String sHz, String sVrt, String sTime)
- synchronized double **getReading** (int nObsTypeId, long lTimestamp, int nLat, int nLon, Date oTimeRecv)
- double **getReading** (int nObsType, long lTimestamp, int[] nIndices)
- synchronized ArrayList< **Obs** > **getWindSpeedData** (long lTimestamp, int nLat1, int nLon1, int nLat2, int nLon2)
- synchronized ArrayList< **Obs** > **getData** (int nObsTypeId, long lTimestamp, int nLat1, int nLon1, int nLat2, int nLon2)

### Additional Inherited Members

---

### Detailed Description

Parses and creates **Obs** from .grb2 received from National Weather Service's Rapid Refresh product. Has specific implementations of getReading and getData for precipitation type and wind speed observations.

### Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

`imrcp.store.RapNcfWrapper.RapNcfWrapper (int[] nObsTypes, String[] sObsTypes, String sHz, String sVrt, String sTime)`

Constructs a new **RapNcfWrapper** with the given parameters

### Parameters

<code>nObsTypes</code>	IMRCP observation types provided in the file
<code>sObsTypes</code>	Label of the observation types found in the file
<code>sHz</code>	horizontal axis label
<code>sVrt</code>	vertical axis label
<code>sTime</code>	time axis label

---

## Member Function Documentation

`synchronized ArrayList< Obs > imrcp.store.RapNcfWrapper.getData (int nObsTypeld, long lTimestamp, int nLat1, int nLon1, int nLat2, int nLon2)`

Returns a list of observations that match the given query parameters. The implementation in `super#getData(int, long, int, int, int, int)` works for all observation types except wind speed, precipitation type, and precipitation category.

Reimplemented from **imrcp.store.NcfWrapper** (*p.374*).

`double imrcp.store.RapNcfWrapper.getReading (int nObsType, long lTimestamp, int[] nIndices)`

Handles the different logic necessary for getting precipitation type and wind speed data.

Reimplemented from **imrcp.store.NcfWrapper** (*p.375*).

`synchronized double imrcp.store.RapNcfWrapper.getReading (int nObsTypeld, long lTimestamp, int nLat, int nLon, Date oTimeRecv)`

Handles the different logic necessary for getting precipitation type and wind speed data.

Reimplemented from **imrcp.store.NcfWrapper** (*p.375*).

`synchronized ArrayList< Obs > imrcp.store.RapNcfWrapper.getWindSpeedData (long lTimestamp, int nLat1, int nLon1, int nLat2, int nLon2)`

Gets a list of wind speed obs that match the given query parameters. The logic is different for wind speed because the u and v vectors have to be combined to get the magnitude.

### Parameters

<code>lTimestamp</code>	query time in milliseconds since Epoch
<code>nLat1</code>	minimum latitude of query in decimal degrees scaled to 7 decimal places
<code>nLon1</code>	minimum longitude of query in decimal degrees scaled to 7 decimal places
<code>nLat2</code>	maximum latitude of query in decimal degrees scaled to 7 decimal places
<code>nLon2</code>	maximum longitude of query in decimal degrees scaled to 7 decimal places

*Returns*

---

*The documentation for this class was generated from the following file:*

- store/RapNcfWrapper.java
- 

## imrcp.store.RAPStore Class Reference

### Public Member Functions

- **RAPStore ()**
- **GriddedFileWrapper getNewFileWrapper ()**

### Additional Inherited Members

---

#### Detailed Description

**FileCache** that manages .grb2 files downloaded from National Weather Service's Rapid Refresh product.

#### Author

Federal Highway Administration

---

#### Constructor & Destructor Documentation

##### *imrcp.store.RAPStore.RAPStore ()*

Default Constructor. Does nothing.

---

#### Member Function Documentation

##### *GriddedFileWrapper imrcp.store.RAPStore.getNewFileWrapper ()*

*Returns*

a new **RapNcfWrapper** with the configured values.

Reimplemented from **imrcp.store.WeatherStore** (*p.564*).

---

*The documentation for this class was generated from the following file:*

- store/RAPStore.java
-

## imrcp.system.Directory.RegisteredBlock Class Reference

### Public Member Functions

- int **compareTo** (**RegisteredBlock** oBlock)
- 

### Detailed Description

This object is what gets registered into the **Directory** representing all of the active system components.

---

### Member Function Documentation

*int imrcp.system.Directory.RegisteredBlock.compareTo (RegisteredBlock oBlock)*

Compares RegisteredBlocks by instance name of their **BaseBlock**

---

*The documentation for this class was generated from the following file:*

- system/Directory.java
- 

## imrcp.collect.RemoteGrid Class Reference

### Public Member Functions

- **RemoteGrid** ()
- void **reset** ()
- void **execute** ()
- boolean **start** () throws Exception
- boolean **stop** () throws Exception

### Protected Member Functions

- void **setError** ()

### Protected Attributes

- String **m\_sCurrentFile** = null
- int **m\_nDirs**
- int **m\_nDirDiff**
- **FilenameFormatter** **m\_oUrlExt**
- String **m\_sStart**
- String **m\_sEnd**
- String **m\_sInitSkip**
- String **m\_sConSkip**
- String **m\_sPattern**
- boolean **m\_bUseNow**
- int **m\_nRecvOffset**
- int **m\_nDownloadPeriod**
- int **m\_nDownloadOffset**
- int **m\_nFileOffsetStart**
- int **m\_nFileOffsetLimit**
- int **m\_nDefaultBuffer**
- int **m\_nTimeout**
- int **m\_nErrorRetry**

## Additional Inherited Members

---

### Detailed Description

Generic collector used to download gridded weather forecasts from different National Weather Service products.

### Author

Federal Highway Administration

---

### Constructor & Destructor Documentation

#### `imrcp.collect.RemoteGrid.RemoteGrid ()`

Default constructor

---

### Member Function Documentation

#### `void imrcp.collect.RemoteGrid.execute ()`

Attempts to download any available file from the configured product that is not saved to disk.

Reimplemented from **imrcp.system.BaseBlock** (*p.95*).

#### `void imrcp.collect.RemoteGrid.reset ()`

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.collect.Collector** (*p.114*).

#### `void imrcp.collect.RemoteGrid.setError () [protected]`

Set the **imrcp.system.BaseBlock#m\_nStatus** to **imrcp.system.ImrcpBlock#ERROR** and then create a timer to schedule the block to call **RemoteGrid#setIdle()** after the configured time.

Reimplemented from **imrcp.system.BaseBlock** (*p.99*).

#### `boolean imrcp.collect.RemoteGrid.start () throws Exception`

Sets a schedule to execute once in 10 seconds so the block is initialized and started before trying to download files because that can take a long time and we don't want the stores to wait on a bulk download to start, and a schedule to execute on a fixed interval.

### Returns

### Exceptions

<code>Exception</code>
------------------------

Reimplemented from **imrcp.system.BaseBlock** (*p.99*).

#### `boolean imrcp.collect.RemoteGrid.stop () throws Exception`

If a file is currently being downloaded and written to disk, delete the partial file.

*Returns*

true if no exceptions are thrown

*Exceptions*

<i>Exception</i>
Reimplemented from <b>imrcp.system.BaseBlock</b> ( <i>p.100</i> ).

---

## Member Data Documentation

### *boolean imrcp.collect.RemoteGrid.m\_bUseNow [protected]*

Set to true if the product has a single file that gets over written for each forecast. Set to false if the product has time dependent file names

### *int imrcp.collect.RemoteGrid.m\_nDefaultBuffer [protected]*

Default size of the buffer than contains the downloaded file

### *int imrcp.collect.RemoteGrid.m\_nDirDiff [protected]*

Number of milliseconds inbetween two consecutive subdirectories

### *int imrcp.collect.RemoteGrid.m\_nDirs [protected]*

Number of subdirectories to check. This is used for products that keep multiple days of forecasts available

### *int imrcp.collect.RemoteGrid.m\_nDownloadOffset [protected]*

The midnight offset in seconds of when a new forecast is expected to be available

### *int imrcp.collect.RemoteGrid.m\_nDownloadPeriod [protected]*

How often it is expected to download a forecast in seconds

### *int imrcp.collect.RemoteGrid.m\_nErrorRetry [protected]*

Time in millisecond to wait after an error has happened to start trying to download files again.

### *int imrcp.collect.RemoteGrid.m\_nFileOffsetLimit [protected]*

For products that have multiple files for a forecast interval, this is the upper limit of file indices to keep

### *int imrcp.collect.RemoteGrid.m\_nFileOffsetStart [protected]*

For products that have multiple files for a forecast interval, this is the lower limit of file indices to keep

### *int imrcp.collect.RemoteGrid.m\_nRecvOffset [protected]*

Time in milliseconds that needs to be added to the timestamp parsed from source files to get the correct received time

### *int imrcp.collect.RemoteGrid.m\_nTimeout [protected]*

Timeout in milliseconds used for **java.net.URLConnection**

### *FilenameFormatter imrcp.collect.RemoteGrid.m\_oUrlExt [protected]*

Object used to create time dependent paths that extend the base url

***String imrcp.collect.RemoteGrid.m\_sConSkip [protected]***

String to search for after a file has been found in the file index to get in a position to be able to find the next file.

***String imrcp.collect.RemoteGrid.m\_sCurrentFile = null [protected]***

Stores the file currently being downloaded

***String imrcp.collect.RemoteGrid.m\_sEnd [protected]***

String to search for in the file index that ends a file name

***String imrcp.collect.RemoteGrid.m\_sInitSkip [protected]***

String to search for to skip all of the header information of the file index

***String imrcp.collect.RemoteGrid.m\_sPattern [protected]***

Regex string used to validate if a file name found in the index is the correct type to download

***String imrcp.collect.RemoteGrid.m\_sStart [protected]***

String to search for in the file index that starts a file name

---

*The documentation for this class was generated from the following file:*

- collect/RemoteGrid.java
- 

## imrcp.web.ReportSubscription Class Reference

### Classes

- enum **Format**

### Public Member Functions

- **ReportSubscription ()**
- **ReportSubscription (Path oPath)** throws Exception
- **ReportSubscription (HttpServletRequest oReq)** throws Exception
- **void clearAll ()**
- **boolean inRange (double dValue)**
- **boolean inRegion (int nLat1, int nLon1, int nLat2, int nLon2)**
- **boolean isObs (int nObsType)**
- **boolean isReport ()**
- **boolean isSubscription ()**
- **boolean hasObstype ()**
- **boolean hasMin ()**
- **boolean hasMax ()**
- **void writeSubConfig (String sBaseDir, String sFileFormat)** throws Exception
- **void updateFulfillmentTime ()** throws Exception
- **void updateLastAccess ()** throws Exception

### Public Attributes

- **String m\_sUuid**
- **String m\_sUsername**
- **int m\_nCycle**
- **Format m\_sOutputFormat**

- long **m\_lRefTime**
- long **m\_lStartTime**
- long **m\_lEndTime**
- int **m\_nMinLat**
- int **m\_nMinLon**
- int **m\_nMaxLat**
- int **m\_nMaxLon**
- int[] **m\_nObsTypes** = new int[0]
- **Id[] m\_oElementIds** = new **Id[0]**
- double **m\_dMin**
- double **m\_dMax**
- long **m\_lFulfillmentTime**
- long **m\_lLastAccess**
- long **m\_lCreatedTime**
- int **m\_nOffset**
- int **m\_nDuration**
- String **m\_sName**
- String **m\_sDescription**

#### [Static Public Attributes](#)

- static final **Format DEFAULT\_FORMAT = Format.CSV**
- 

#### [Detailed Description](#)

This class represents a report or subscription created by the IMRCP Create Reports UI.

#### *Author*

Federal Highway Administration

---

#### [Constructor & Destructor Documentation](#)

##### *imrcp.web.ReportSubscription.ReportSubscription ()*

Default constructor. Calls **clearAll ()** to set all values to their defaults.

##### *imrcp.web.ReportSubscription.ReportSubscription (Path oPath) throws Exception*

Constructs a **ReportSubscription** from the Path that points to the report/subscription configuration file.

#### *Parameters*

<i>oPath</i>	Path to the report/subscription configuration file
--------------	--

#### *Exceptions*

<i>Exception</i>
------------------

##### *imrcp.web.ReportSubscription.ReportSubscription (HttpServletRequest oReq) throws Exception*

Constructs a **ReportSubscription** from the given Http request

#### *Parameters*

<i>oReq</i>	object that contains the request the client has made of the servlet
-------------	---

## Exceptions

<i>Exception</i>	
------------------	--

---

## Member Function Documentation

*void imrcp.web.ReportSubscription.clearAll ()*

Resets all the member variables to the default values

*boolean imrcp.web.ReportSubscription.hasMax ()*

Determines if this report/subscription has a configured maximum value

*Returns*

true if **is a finite value**

*boolean imrcp.web.ReportSubscription.hasMin ()*

Determines if this report/subscription has a configured minimum value

*Returns*

true if **is a finite value**

*boolean imrcp.web.ReportSubscription.hasObstype ()*

Determines if this report/subscription has a configured observation type id array.

*Returns*

**m\_nObsTypes != null**

*boolean imrcp.web.ReportSubscription.inRange (double dValue)*

Determines if the value is within the min and max value.

*Parameters*

<i>dValue</i>	value to test
---------------	---------------

*Returns*

true if the value satisfies **m\_dMin <= dValue <= m\_dMax** or the length of **m\_nObsTypes** is 0 true, otherwise false.

*boolean imrcp.web.ReportSubscription.inRegion (int nLat1, int nLon1, int nLat2, int nLon2)*

Determines if the given bounding box intersects the bounding box of the report/subscription

*Parameters*

<i>nLat1</i>	minimum latitude in decimal degrees scaled to 7 decimal places
<i>nLon1</i>	maximum longitude in decimal degrees scaled to 7 decimal places
<i>nLat2</i>	minimum latitude in decimal degrees scaled to 7 decimal places
<i>nLon2</i>	maximum longitude in decimal degrees scaled to 7 decimal places

*Returns*

*boolean imrcp.web.ReportSubscription.isObs (int nObsType)*

Determine if the observation type id matches the report/subscription's observation type

*Parameters*

<i>nObsType</i>	IMRCP observation type id
-----------------	---------------------------

*Returns*

true if **m\_nObsTypes** length is 0 or if nObsType equals one of the values in **m\_nObsTypes**

*boolean imrcp.web.ReportSubscription.isReport ()*

Determines if this object represents a report

*Returns*

true if this object is a report, otherwise false

*boolean imrcp.web.ReportSubscription.isSubscription ()*

Determines if this object represents a subscription

*Returns*

true if this object is a subscription, otherwise false.

*void imrcp.web.ReportSubscription.updateFulfillmentTime () throws Exception*

Updates the fulfillment time in the report/subscription configuration file to the current time

*Exceptions*

<i>Exception</i>
------------------

*void imrcp.web.ReportSubscription.updateLastAccess () throws Exception*

Updates the last access time in the report/subscription configuration file to the current time

*Exceptions*

<i>Exception</i>
------------------

*void imrcp.web.ReportSubscription.writeSubConfig (String sBaseDir, String sFileFormat) throws Exception*

Write the parameters of this report/subscription to its configuration file.

*Parameters*

<i>sBaseDir</i>	base directory for report/subscriptions
<i>sFileFormat</i>	Format String used to generate file names

*Exceptions*

<i>Exception</i>
------------------

## Member Data Documentation

*final Format imrcp.web.ReportSubscription.DEFAULT\_FORMAT = Format.CSV [static]*

Default format enumeration, current CSV

*double imrcp.web.ReportSubscription.m\_dMax*

Maximum value of observation to include in the report/subscription

*double imrcp.web.ReportSubscription.m\_dMin*

Minimum value of observations to include in the report/subscription

*long imrcp.web.ReportSubscription.m\_lCreatedTime*

Time in milliseconds since Epoch the report/subscription was created

*long imrcp.web.ReportSubscription.m\_lEndTime*

The end time in milliseconds since Epoch to be used for the query to fulfill a report. Not used for subscriptions

*long imrcp.web.ReportSubscription.m\_lFulfillmentTime*

Time in milliseconds since Epoch the report/subscription was fulfilled

*long imrcp.web.ReportSubscription.m\_lLastAccess*

Time in milliseconds since Epoch the report/subscription was last accessed

*long imrcp.web.ReportSubscription.m\_lRefTime*

The reference time in milliseconds since Epoch to be used for the query to fulfill a report. Not used for subscriptions

*long imrcp.web.ReportSubscription.m\_lStartTime*

The start time in milliseconds since Epoch to be used for the query to fulfill a report. Not used for subscriptions

*int imrcp.web.ReportSubscription.m\_nCycle*

The number of minutes that need to elapse before the subscription creates a new data file. For reports, this value is 0.

*int imrcp.web.ReportSubscription.m\_nDuration*

Time in minutes to add to the start time of a query to get the end time of the query

*int imrcp.web.ReportSubscription.m\_nMaxLat*

Maximum latitude of the roadway segments included in the report/subscription

*int imrcp.web.ReportSubscription.m\_nMaxLon*

Maximum longitude of the roadway segments included in the report/subscription

*int imrcp.web.ReportSubscription.m\_nMinLat*

Minimum latitude of the roadway segments included in the report/subscription

*int imrcp.web.ReportSubscription.m\_nMinLon*

Minimum longitude of the roadway segments included in the report/subscription

*int [] imrcp.web.ReportSubscription.m\_nObsTypes = new int[0]*

Stores the observation types to query for this report/subscription

*int imrcp.web.ReportSubscription.m\_nOffset*

Time in minutes to offset the reference time when creating a data file to get the start time of the query.

*Id [] imrcp.web.ReportSubscription.m\_oElementIds = new Id[0]*

Stores the Ids of the roadway segments included in this report/subscription

*String imrcp.web.ReportSubscription.m\_sDescription*

Description given to the report/subscription by the user

*String imrcp.web.ReportSubscription.m\_sName*

Name given to the report/subscription by the user

*Format imrcp.web.ReportSubscription.m\_sOutputFormat*

The output format to use for data files created by this report/subscription

*String imrcp.web.ReportSubscription.m\_sUsername*

User that created the report/subscription

*String imrcp.web.ReportSubscription.m\_sUuid*

Id of the report/subscription

---

*The documentation for this class was generated from the following file:*

- web/ReportSubscription.java
- 

## imrcp.forecast.mdss.RoadcastData Class Reference

### Public Member Functions

- int **compareTo** (RoadcastData o)

### Public Attributes

- int[] **m\_nStpvt**
  - float[] **m\_fTpvt**
  - float[] **m\_fTssrf**
  - float[] **m\_fDphliq**
  - float[] **m\_fDphsn**
  - long[] **m\_lStartTimes**
  - long[] **m\_lEndTimes**
- 

### Detailed Description

METRo calls its outputs Roadcasts which contain forecasts for the pavement state, pavement temperature, subsurface temperature, depth of accumulated liquid , and depth of accumulated ice/snow

*Author*

Federal Highway Administration

---

**Member Function Documentation**

*int imrcp.forecast.mdss.RoadcastData.compareTo (RoadcastData o)*

C.compares **RoadcastData** by longitude, then latitude

*Parameters*

<i>o</i>	the object to be compared
----------	---------------------------

*Returns*

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

*See also*

[java.lang.Comparable::compareTo\(java.lang.Object\)](#)

---

**Member Data Documentation**

*float [] imrcp.forecast.mdss.RoadcastData.m\_fDphliq*

Stores output depth of accumulated liquid forecasts from METRo

*float [] imrcp.forecast.mdss.RoadcastData.m\_fDphsn*

Stores output depth of accumulated ice/snow forecasts from METRo

*float [] imrcp.forecast.mdss.RoadcastData.m\_fTpvt*

Stores output pavement temperature forecasts from METRo

*float [] imrcp.forecast.mdss.RoadcastData.m\_fTssrf*

Stores output subsurface temperature forecasts from METRo

*long [] imrcp.forecast.mdss.RoadcastData.m\_lEndTimes*

Stores the end times of each forecast

*long [] imrcp.forecast.mdss.RoadcastData.m\_lStartTimes*

Stores the start times of each forecast

*int [] imrcp.forecast.mdss.RoadcastData.m\_nStpvt*

Stores output pavement state forecasts from METRo

---

*The documentation for this class was generated from the following file:*

- [forecast/mdss/RoadcastData.java](#)
-

## imrcp.web.layers.RoadLayerServlet Class Reference

### Public Member Functions

- void **reset** ()

### Protected Member Functions

- void **buildObsResponseContent** (JsonGenerator oOutputGenerator, **ObsRequest** oObsRequest) throws Exception
- void **buildObsChartResponseContent** (JsonGenerator oOutputGenerator, **ObsChartRequest** oObsRequest) throws Exception

### Additional Inherited Members

---

#### Detailed Description

Handles requests from the IMRCP Map UI when Road(line string) layer objects are clicked or when a chart for Road observations is requested

#### Author

Federal Highway Administration

---

#### Member Function Documentation

**void imrcp.web.layers.RoadLayerServlet.buildObsChartResponseContent (JsonGenerator oOutputGenerator, ObsChartRequest oObsRequest) throws Exception [protected]**

Add the response to the given JSON stream for requests made from the IMRCP Map UI to create a chart for road observations

Reimplemented from **imrcp.web.layers.LayerServlet** (*p.322*).

**void imrcp.web.layers.RoadLayerServlet.buildObsResponseContent (JsonGenerator oOutputGenerator, ObsRequest oObsRequest) throws Exception [protected]**

Add the response to the given JSON stream for requests made from the IMRCP Map UI when a Road Layer object is clicked.

Reimplemented from **imrcp.web.layers.LayerServlet** (*p.323*).

**void imrcp.web.layers.RoadLayerServlet.reset ()**

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.web.layers.LayerServlet** (*p.324*).

---

*The documentation for this class was generated from the following file:*

- web/layers/RoadLayerServlet.java
- 
-

## imrcp.web.Scenario Class Reference

### Public Member Functions

- **Scenario** (Path oPath) throws IOException
- **Scenario** (JSONObject oJson) throws IOException
- final void **setValues** (JSONObject oJson, Path oPath) throws IOException
- void **generateId** () throws IOException
- JSONObject **toJSONObject** (boolean bIncludeProcessed, boolean bIncludeUser)

### Public Attributes

- String **m\_sUser**
- String **m\_sName**
- String **m\_sId**
- long **m\_lStartTime**
- SegmentGroup[] **m\_oGroups**
- boolean **m\_bRun**
- boolean **m\_bProcessed** = false
- String **m\_sNetwork**

---

### Detailed Description

This class encapsulates all of the parameters needed to represent a **Scenario**. A **Scenario** consists of groups of roadway segments with an associated time series of actions for each group.

#### Author

Federal Highway Administration

---

### Constructor & Destructor Documentation

#### *imrcp.web.Scenario.Scenario (Path oPath) throws IOException*

Constructs a **Scenario** from the given Path which should point to a JSON file defining the **Scenario**

##### Parameters

<i>oPath</i>	Path of the JSON file
--------------	-----------------------

##### Exceptions

<i>IOException</i>
--------------------

#### *imrcp.web.Scenario.Scenario (JSONObject oJson) throws IOException*

Constructs a **Scenario** from the given JSONObject by calling **setValues (org.json.JSONObject, java.nio.file.Path)**

##### Parameters

<i>oJson</i>	JSONObject defining the <b>Scenario</b>
--------------	---

##### Exceptions

<i>IOException</i>
--------------------

## Member Function Documentation

*void imrcp.web.Scenario.generateId () throws IOException*

Uses a Shake256 squeeze algorithm to create a unique id based off of the parameters of the **Scenario**.

### Exceptions

<i>IOException</i>
--------------------

*final void imrcp.web.Scenario.setValues (JSONObject oJson, Path oPath) throws IOException*

Sets the member variables of this **Scenario** by parsing the given JSON object.

### Parameters

<i>oJson</i>	JSONObject defining the <b>Scenario</b>
<i>oPath</i>	Path of the file containing the JSON definition, can be null.

### Exceptions

<i>IOException</i>
--------------------

*JSONObject imrcp.web.Scenario.toJSONObject (boolean bIncludeProcessed, boolean bIncludeUser)*

Serializes the **Scenario** into a JSONObject.

### Parameters

<i>bIncludeProcessed</i>	flag indicating if the processed field should be included
<i>bIncludeUser</i>	flag indicating if the user field should be included

### Returns

JSONObject that represents the **Scenario**

---

## Member Data Documentation

*boolean imrcp.web.Scenario.m\_bProcessed = false*

Flag indicating if the **Scenario** has finished processing or not

*boolean imrcp.web.Scenario.m\_bRun*

Flag indicating if the **Scenario** is being ran for a specific time or if it is a **Scenario** template

*long imrcp.web.Scenario.m\_lStartTime*

Start time of the **Scenario** in milliseconds since Epoch

*SegmentGroup [] imrcp.web.Scenario.m\_oGroups*

The roadway segment groups

#### *String imrcp.web.Scenario.m\_sId*

**Scenario** Id which is a UUID if this object represents a **Scenario** template or a 32 byte array converted to a String using Base64 if this object is a **Scenario** that is being ran for a specific time

#### *String imrcp.web.Scenario.m\_sName*

The label given to the **Scenario**

#### *String imrcp.web.Scenario.m\_sNetwork*

Network ID of the Network used for this **Scenario**

#### *String imrcp.web.Scenario.m\_sUser*

The user that created the **Scenario**

---

*The documentation for this class was generated from the following file:*

- web/Scenario.java
- 

## imrcp.web.Scenarios Class Reference

### Public Member Functions

- boolean **start** () throws Exception
- void **reset** ()
- int **doSave** (HttpServletRequest oReq, HttpServletResponse oRes, Session oSess) throws IOException, ServletException
- int **doDeleteTemplate** (HttpServletRequest oReq, HttpServletResponse oRes, Session oSess) throws IOException, ServletException
- int **doList** (HttpServletRequest oReq, HttpServletResponse oRes, Session oSess) throws IOException, ServletException
- int **doRun** (HttpServletRequest oReq, HttpServletResponse oRes, Session oSess) throws IOException, ServletException
- int **doData** (HttpServletRequest oReq, HttpServletResponse oRes, Session oSess) throws IOException, ServletException
- ArrayList< **Scenario** > **getScenarios** (String sUserName) throws IOException
- **Scenario getScenario** (String sId) throws IOException
- void **execute** ()

### Additional Inherited Members

---

#### Detailed Description

This servlet manages all of the requests and processing necessary to run the Create and View **Scenarios** UIs.

#### *Author*

Federal Highway Administration

## Member Function Documentation

*int imrcp.web.Scenarios.doData (HttpServletRequest oReq, HttpServletResponse oRes, Session oSess) throws IOException, ServletException*

Adds the data associated with the **Scenario** defined in the request to the response.

### Parameters

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client
<i>oSess</i>	object that contains information about the user that made the request

### Returns

HTTP status code to be included in the response.

### Exceptions

<i>IOException</i>	
<i>ServletException</i>	

*int imrcp.web.Scenarios.doDeleteTemplate (HttpServletRequest oReq, HttpServletResponse oRes, Session oSess) throws IOException, ServletException*

Deletes the **Scenario** Template defined in the request

### Parameters

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client
<i>oSess</i>	object that contains information about the user that made the request

### Returns

HTTP status code to be included in the response.

### Exceptions

<i>IOException</i>	
<i>ServletException</i>	

*int imrcp.web.Scenarios.doList (HttpServletRequest oReq, HttpServletResponse oRes, Session oSess) throws IOException, ServletException*

Adds a list of the **Scenarios** to the response, that if the "processed" parameter exists and is true represent the **Scenarios** that have been queued to process, otherwise the list represents the **Scenario** Templates

### Parameters

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client
<i>oSess</i>	object that contains information about the user that made the request

*Returns*

HTTP status code to be included in the response.

*Exceptions*

<i>IOException</i>	
<i>ServletException</i>	
<i>n</i>	

*int imrcp.web.Scenarios.doRun (HttpServletRequest oReq, HttpServletResponse oRes, Session oSess) throws IOException, ServletException*

Queues the **Scenario** Template defined in the request to process.

*Parameters*

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client
<i>oSess</i>	object that contains information about the user that made the request

*Returns*

HTTP status code to be included in the response.

*Exceptions*

<i>IOException</i>	
<i>ServletException</i>	
<i>n</i>	

*int imrcp.web.Scenarios.doSave (HttpServletRequest oReq, HttpServletResponse oRes, Session oSess) throws IOException, ServletException*

Saves the **Scenario** Template defined in the request.

*Parameters*

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client
<i>oSess</i>	object that contains information about the user that made the request

*Returns*

HTTP status code to be included in the response.

*Exceptions*

<i>IOException</i>	
<i>ServletException</i>	
<i>n</i>	

*void imrcp.web.Scenarios.execute ()*

Iterates through the directories in the **Scenarios** base directory and processes any **Scenarios** that need to be processed.

Reimplemented from **imrcp.system.BaseBlock** (p.95).

*Scenario imrcp.web.Scenarios.getScenario (String sId) throws IOException*

Get the **Scenario** object with the given id. This can be a UUID for **Scenario** Templates or a Shake256 squeeze algorithm id for **Scenarios** that have been processed.

*Parameters*

<i>sId</i>	<b>Scenario</b> id.
------------	---------------------

*Returns*

The **Scenario** with the given id if it exists, otherwise null

*Exceptions*

<i>IOException</i>
--------------------

*ArrayList< Scenario > imrcp.web.Scenarios.getScenarios (String sUserName) throws IOException*

Gets a list of **Scenario** objects that the given user created. If *sUserName* is null then all **Scenarios** are returned.

*Parameters*

<i>sUserName</i>	user name to check against the user name that created the <b>Scenarios</b> .
------------------	--

*Returns*

a list of **Scenario** objects that the given user created. If *sUserName* is null then all **Scenarios** are returned.

*Exceptions*

<i>IOException</i>
--------------------

*void imrcp.web.Scenarios.reset ()*

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.web.SecureBaseBlock** (p.492).

*boolean imrcp.web.Scenarios.start () throws Exception*

If the base directory doesn't exists, it creates the necessary parent directories and the configured base directory. It then iterates through the existing scenarios and loads the templates into memory in *m\_oTemplates*. Then sets a schedule to execute on a fixed interval.

*Returns*

true if no Exceptions are thrown

*Exceptions*

<i>Exception</i>
------------------

Reimplemented from **imrcp.system.BaseBlock** (p.99).

The documentation for this class was generated from the following file:

- web/Scenarios.java
- 

## imrcp.system.Scheduling Class Reference

### Public Member Functions

- synchronized int **createSched** (Runnable iRunnable, int nOffset, int nPeriod)
- synchronized int **createSched** (Runnable iRunnable, Date oTime, int nPeriod)
- void **scheduleOnce** (Runnable iRunnable, int nDelay)
- void **scheduleOnce** (Runnable iRunnable, Date oWhen)
- void **execute** (Runnable iRunnable)
- void **stop** ()
- synchronized boolean **cancelSched** (Runnable iRunnable, int nSchedId)

### Static Public Member Functions

- static **Scheduling getInstance** ()
  - static Calendar **getNextPeriod** (int nOffset, int nPeriod)
  - static Calendar **getLastPeriod** (int nOffset, int nPeriod)
- 

### Detailed Description

Singleton class that contains a thread pool and manages scheduling tasks for the system.

#### Author

Federal Highway Administration

---

### Member Function Documentation

**synchronized boolean imrcp.system.Scheduling.cancelSched (Runnable iRunnable, int nSchedId)**

Attempts to cancel the schedule for the given Runnable and schedule id.

#### Parameters

<i>iRunnable</i>	Runnable to cancel its fixed interval of execution
<i>nSchedId</i>	schedule id associated with the Runnable

#### Returns

true if the task was successfully canceled and removed from the task list, otherwise false.

**synchronized int imrcp.system.Scheduling.createSched (Runnable iRunnable, Date oTime, int nPeriod)**

Creates a Sched object wrapping the given Runnable, adds it to the task list, **m\_oTasks** and uses the Timer, **m\_oTimer** to schedule the task to be executed at a fixed rate, with the first time it is executed is the time in the given Date.

#### Parameters

<i>iRunnable</i>	Runnable to schedule for execution
------------------	------------------------------------

<i>oTime</i>	Time to first execute the task
<i>nPeriod</i>	Period of execution in seconds

*Returns*

The schedule id assigned to the task

*synchronized int imrcp.system.Scheduling.createSched (Runnable iRunnable, int nOffset, int nPeriod)*

Creates a Sched object wrapping the given Runnable, adds it to the task list, **m\_oTasks** and uses the Timer, **m\_oTimer** to schedule the task to be executed at a fixed rate based off of the given offset and period.

*Parameters*

<i>iRunnable</i>	Runnable to schedule for execution
<i>nOffset</i>	Schedule offset from midnight in seconds
<i>nPeriod</i>	Period of execution in seconds

*Returns*

The schedule id assigned to the task

*void imrcp.system.Scheduling.execute (Runnable iRunnable)*

Wrapper for **m\_iExecutor#execute(java.lang.Runnable)**

*Parameters*

<i>iRunnable</i>	The Runnable to execute
------------------	-------------------------

*static Scheduling imrcp.system.Scheduling.getInstance () [static]*

Gets the singleton instance

*Returns*

The singleton instance

*static Calendar imrcp.system.Scheduling.getLastPeriod (int nOffset, int nPeriod) [static]*

Gets a Calendar with its time set to the last period of execution based off of the given offset and period.

*Parameters*

<i>nOffset</i>	Schedule offset from midnight in seconds
<i>nPeriod</i>	Period of execution in seconds

*Returns*

Calendar with the time set as the last period of execution

*static Calendar imrcp.system.Scheduling.getNextPeriod (int nOffset, int nPeriod) [static]*

Gets a Calendar with its time set to the next period of execution based off of the given offset and period.

*Parameters*

<i>nOffset</i>	Schedule offset from midnight in seconds
<i>nPeriod</i>	Period of execution in seconds

*Returns*

Calendar with the time set as the next period of execution

**void imrcp.system.Scheduling.scheduleOnce (Runnable iRunnable, Date oWhen)**

Schedules the given Runnable to execute once at the given time.

*Parameters*

<i>iRunnable</i>	Runnable to schedule for execution
<i>oWhen</i>	Time to execute the Runnable

**void imrcp.system.Scheduling.scheduleOnce (Runnable iRunnable, int nDelay)**

Schedules the given Runnable to execute once after the given delay in milliseconds

*Parameters*

<i>iRunnable</i>	Runnable to schedule for execution
<i>nDelay</i>	time to wait in milliseconds before executing the Runnable

**void imrcp.system.Scheduling.stop ()**

Wrapper for `java.util.concurrent.ExecutorService#shutdownNow()`

---

*The documentation for this class was generated from the following file:*

- system/Scheduling.java
- 

## imrcp.web.SecureBaseBlock Class Reference

### Public Member Functions

- `void init (ServletConfig oSConfig) throws ServletException`
- `void reset ()`
- `void doGet (HttpServletRequest oReq, HttpServletResponse oRes) throws IOException, ServletException`
- `void doPost (HttpServletRequest oReq, HttpServletResponse oRes) throws IOException, ServletException`

### Protected Member Functions

- `SecureBaseBlock ()`

### Additional Inherited Members

---

### Detailed Description

Base class used for system components that receive remote requests from the IMRCP Web Application.

*Author*

Federal Highway Administration

---

**Constructor & Destructor Documentation**

*imrcp.web.SecureBaseBlock.SecureBaseBlock () [protected]*

Default constructor. Wrapper from **BaseBlock#BaseBlock ()**

---

**Member Function Documentation**

*void imrcp.web.SecureBaseBlock.doGet (HttpServletRequest oReq,  
HttpServletResponse oRes) throws IOException, ServletException*

Wrapper for **doPost(HttpServletRequest, HttpServletResponse)**

*Parameters*

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client

*Exceptions*

<i>IOException</i>	
<i>ServletException</i>	
<i>n</i>	

*void imrcp.web.SecureBaseBlock.doPost (HttpServletRequest oReq,  
HttpServletResponse oRes) throws IOException, ServletException*

Processes the given request by first authenticating the request and then calling the appropriate method.

*Parameters*

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client

*Exceptions*

<i>IOException</i>	
<i>ServletException</i>	
<i>n</i>	

Reimplemented in **imrcp.web.Subscriptions** (*p.506*).

*void imrcp.web.SecureBaseBlock.init (ServletConfig oSConfig) throws ServletException*

Initializes the block. Wrapper for **setName(java.lang.String)**, **setLogger()**, **setConfig()**, **register()**, and **startService()**

*Parameters*

<i>oSConfig</i>	object containing configuration parameters in Tomcat's
-----------------	--

### *Exceptions*

<i>ServletException</i>
<i>n</i>

Reimplemented in **imrcp.web.tiles.TileServlet** (p.525).

### *void imrcp.web.SecureBaseBlock.reset ()*

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.system.BaseBlock** (p.98).

Reimplemented in **imrcp.store.BinObsStore** (p.102), **imrcp.store.FileCache** (p.214), **imrcp.store.GribStore** (p.250), **imrcp.store.MetroStore** (p.341), **imrcp.store.WeatherStore** (p.564), **imrcp.web.LaneServlet** (p.317), **imrcp.web.layers.AreaLayerServlet** (p.86), **imrcp.web.layers.LayerServlet** (p.324), **imrcp.web.layers.RoadLayerServlet** (p.481), **imrcp.web.NetworkGeneration** (p.389), **imrcp.web.NotificationServlet** (p.398), **imrcp.web.Scenarios** (p.487), **imrcp.web.Subscriptions** (p.508), **imrcp.web.tiles.GribTileCache** (p.250), **imrcp.web.tiles.NcfTileCache** (p.372), **imrcp.web.tiles.TileCache** (p.522), **imrcp.web.tiles.TileServlet** (p.525), **imrcp.web.USCenPlaceLookup** (p.545), and **imrcp.web.UserSettingsServlet** (p.548).

---

*The documentation for this class was generated from the following file:*

- `web/SecureBaseBlock.java`
- 

## **imrcp.geosrv.Seglterator Class Reference**

### **Public Member Functions**

- `boolean hasNext ()`
  - `int[] next ()`
  - `void remove ()`
- 

### **Detailed Description**

Object used to iterate over an int array that represents a roadway segment (linestring) by line segment.

### **Author**

Federal Highway Administration

---

### **Member Function Documentation**

#### *int[] imrcp.geosrv.Seglterator.next ()*

Copies the points of the next line segment into **m\_nLine** and returns it reference.

#### *void imrcp.geosrv.Seglterator.remove ()*

NOT IMPLEMENTED

---

The documentation for this class was generated from the following file:

- geosrv/SegIterator.java
- 

## imrcp.web.SegmentGroup Class Reference

### Public Member Functions

- **SegmentGroup** (JSONObject oGroup)

### Public Attributes

- String **m\_sLabel**
  - Id[] **m\_oSegments**
  - boolean[] **m\_bPlowed**
  - boolean[] **m\_bTreated**
  - int[] **m\_nVsl**
  - int[] **m\_nLanes**
- 

### Detailed Description

This class represents a group of segments and the actions defined for them in a **Scenario**.

#### Author

Federal Highway Administration

---

### Constructor & Destructor Documentation

*imrcp.web.SegmentGroup.SegmentGroup (JSONObject oGroup)*

Constructs a **SegmentGroup** from the JSON representation of it.

#### Parameters

<i>oGroup</i>	JSON object defining the <b>SegmentGroup</b>
---------------	--

---

### Member Data Documentation

*boolean [] imrcp.web.SegmentGroup.m\_bPlowed*

Array indicating if the segments were plowed for each hour of the **Scenario**

*boolean [] imrcp.web.SegmentGroup.m\_bTreated*

Array indicating if the segments were treated for each hour of the **Scenario**

*int [] imrcp.web.SegmentGroup.m\_nLanes*

Array indicating how many lanes are available for each hour of the **Scenario**

*int [] imrcp.web.SegmentGroup.m\_nVsl*

Array indicating what the Variable Speed Limit is for each hour of the **Scenario**

*Id [] imrcp.web.SegmentGroup.m\_oSegments*

Id of the segments included in this segment group

*String imrcp.web.SegmentGroup.m\_sLabel*

Label of the segment group

---

*The documentation for this class was generated from the following file:*

- web/SegmentGroup.java
- 

## imrcp.web.Session Class Reference

### Public Member Functions

- int **compare** (**Session** oLhs, **Session** oRhs)

### Public Attributes

- String **m\_sName**
- **UserProfile** **m\_oProfile** = null

### Protected Attributes

- long **m\_lLastAccess**
  - long **m\_lReset**
  - byte[] **m\_ySalt**
  - String **m\_sPass**
  - String **m\_sContact**
  - String **m\_sGroup**
  - String **m\_sToken** = ""
- 

### Detailed Description

This class represents a User and their sessions being logged into the IMRCP web application.

#### Author

Federal Highway Administration

---

### Member Function Documentation

*int imrcp.web.Session.compare (Session oLhs, Session oRhs)*

C.compares Sessions by user name

---

### Member Data Documentation

*long imrcp.web.Session.m\_lLastAccess [protected]*

Time in milliseconds since Epoch the **Session** last accessed something from the system.

*long imrcp.web.Session.m\_lReset [protected]*

Time in millisecond since Epoch that a reset password request is valid

*UserProfile imrcp.web.Session.m\_oProfile = null*

**UserProfile** object associated with the user

*String imrcp.web.Session.m\_sContact [protected]*

Email address used to contact the user

*String imrcp.web.Session.m\_sGroup [protected]*

; separated string of the system groups the user belongs too

*String imrcp.web.Session.m\_sName*

User name

*String imrcp.web.Session.m\_sPass [protected]*

Hashed password as a hex string

*String imrcp.web.Session.m\_sToken = "" [protected]*

Stores the active session token

*byte [] imrcp.web.Session.m\_ySalt [protected]*

Random bytes used to hash the password

---

*The documentation for this class was generated from the following file:*

- web/Session.java
- 

## imrcp.web.SessMgr Class Reference

### Public Member Functions

- **SessMgr ()**
- **void init ()**
- **void doPost (HttpServletRequest oReq, HttpServletResponse oRep)**
- **int compare (Session oLhs, Session oRhs)**

### Static Public Member Functions

- **static SessMgr getInstance ()**
- **static void main (String[] sArgs) throws Exception**

### Static Protected Attributes

- **static long LAST\_REFRESH = 0L**
  - **static long TIMEOUT = 1800000**
  - **static MessageDigest DIGEST**
- 

### Detailed Description

This singleton class manages the system Users and their Sessions using the IMRCP Web Application.

### Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

*imrcp.web.SessMgr.SessMgr ()*

Default constructor. Initializes the handlers

---

## Member Function Documentation

*int imrcp.web.SessMgr.compare (Session oLhs, Session oRhs)*

C.compares **Session** by session token

*void imrcp.web.SessMgr.doPost (HttpServletRequest oReq, HttpServletResponse oRep)*

Manages all of the requests sent dealing with user authentication and sessions of the IMRCP Web Application. It does this by parsing the request and calling the appropriate Handler.

### Parameters

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRep</i>	object that contains the response the servlet sends to the client

*static SessMgr imrcp.web.SessMgr.getInstance () [static]*

Gets the instance of the singleton

### Returns

The **SessMgr** singleton instance

*void imrcp.web.SessMgr.init ()*

Initializes the servlet by reading all of the configuration parameters from the ServletConfig object.

*static void imrcp.web.SessMgr.main (String[] sArgs) throws Exception [static]*

Main function that is intended to be used outside of the system to generate lines for the IMRCP user CSV file. It hashes and saved the encrypted password as a hex string. The email address and group list might need to be edited.

### Parameters

<i>sArgs</i>	[user name, password]
--------------	-----------------------

### Exceptions

<i>Exception</i>	
------------------	--

---

## Member Data Documentation

*MessageDigest imrcp.web.SessMgr.DIGEST [static], [protected]*

MessageDigest used for one-way hash algorithms

*long imrcp.web.SessMgr.LAST\_REFRESH = 0L [static], [protected]*

Time in milliseconds since Epoch the IMRCP user CSV file was read to refresh User definitions.

*long imrcp.web.SessMgr.TIMEOUT = 1800000 [static], [protected]*

Time in milliseconds used to determine if a **Session** is stale

---

*The documentation for this class was generated from the following file:*

- web/SessMgr.java
- 

## imrcp.store.SourceUnit Class Reference

### Public Member Functions

- **SourceUnit (CsvReader oIn)**

### Public Attributes

- int **m\_nObsTypeId**
  - int **m\_nContribId**
  - String **m\_sUnit**
- 

### Detailed Description

Contains the units used by data collected different contributors.

#### Author

Federal Highway Administration

---

### Constructor & Destructor Documentation

*imrcp.store.SourceUnit.SourceUnit (CsvReader oIn)*

Constructs a new **SourceUnit** from a line of the SourceUnits CSV file.

#### Parameters

<i>oIn</i>	CsvReader wrapping the InputStream of the SourceUnits CSV file ready to parse the current line.
------------	---

### Member Data Documentation

*int imrcp.store.SourceUnit.m\_nContribId*

IMRCP contributor id of data

*int imrcp.store.SourceUnit.m\_nObsTypeId*

IMRCP observation type id of data

*String imrcp.store.SourceUnit.m\_sUnit*

Units used in the source files

---

*The documentation for this class was generated from the following file:*

- store/SourceUnit.java

---

## *imrcp.store.SpatialFileCache* Class Reference

### Classes

- class **TileXCache**
- class **TileYCache**

### Public Member Functions

- **SpatialFileCache** (int nZoom)
- void **process** (String[] sMessage)
- boolean **start** () throws Exception
- **FileWrapper getFile** (long lTimestamp, long lRefTime, int nLon, int nLat)
- boolean **loadFileToMemory** (String sFullPath, int nFormatIndex, **TileYCache** oYCache)
- void **execute** ()

### Public Attributes

- final int **m\_nZoom**

### Protected Member Functions

- boolean **loadFileToCache** (long lTimestamp, long lRefTime, int nLon, int nLat)
- **TileXCache searchXCache** (int nTileX)

### Protected Attributes

- ArrayList< **TileXCache** > **m\_oSpatialCache** = new ArrayList()

### Additional Inherited Members

---

### Detailed Description

Contains methods for managing the caching of data files that are spatially indexed (which is based on map tiles) on disk into memory for quick access.

#### *Author*

Federal Highway Administration

---

### Constructor & Destructor Documentation

#### *imrcp.store.SpatialFileCache.SpatialFileCache (int nZoom)*

Constructs a new **SpatialFileCache** with the given zoom level used to index the files. **m\_oCache** is set to null since **SpatialFileCache** used a different method to cache files using **m\_oSpatialCache**

#### *Parameters*

<i>nZoom</i>	Zoom level used to spatially index files
--------------	--

---

### Member Function Documentation

#### *void imrcp.store.SpatialFileCache.execute ()*

Iterates through the caches and removes stale files

Reimplemented from **imrcp.store.FileCache** (p.211).

*FileWrapper imrcp.store.SpatialFileCache.getFile (long lTimestamp, long lRefTime, int nLon, int nLat)*

Retrieves the "best" file that is valid for the given query and reference time. If that file is not in the cache, it is loaded into the cache in this process.

#### Parameters

<i>lTimestamp</i>	query time in milliseconds since Epoch
<i>lRefTime</i>	reference time in milliseconds since Epoch
<i>nLon</i>	longitude of query in decimal degrees scaled to 7 decimal places. Used <code>Integer.MIN_VALUE</code> if not a <b>SpatialFileCache</b>
<i>nLat</i>	latitude of query in decimal degrees scaled to 7 decimal places. Used <code>Integer.MIN_VALUE</code> if not a <b>SpatialFileCache</b>

#### Returns

Reimplemented from **imrcp.store.FileCache** (p.212).

*boolean imrcp.store.SpatialFileCache.loadFileToCache (long lTimestamp, long lRefTime, int nLon, int nLat) [protected]*

Attempts to load a file into memory that is valid for the query time and reference time. The longitude and latitude are only used in the implementation of this function for SpatialFileCaches.

#### Parameters

<i>lTimestamp</i>	query time in milliseconds since Epoch
<i>lRefTime</i>	reference time in milliseconds since Epoch
<i>nLon</i>	longitude of query in decimal degrees scaled to 7 decimal places. Used <code>Integer.MIN_VALUE</code> if not a <b>SpatialFileCache</b>
<i>nLat</i>	latitude of query in decimal degrees scaled to 7 decimal places. Used <code>Integer.MIN_VALUE</code> if not a <b>SpatialFileCache</b>

#### Returns

true if a file valid for the query and reference was already in the cache or was loaded into the cache, otherwise false

Reimplemented from **imrcp.store.FileCache** (p.213).

*boolean imrcp.store.SpatialFileCache.loadFileToMemory (String sFullPath, int nFormatIndex, TileYCache oYCache)*

Attempts to load the given file path into memory and places it in the given cache

#### Parameters

<i>sFullPath</i>	File path to load
<i>nFormatIndex</i>	index of <code>m_oFormatters</code> to use
<i>oYCache</i>	

#### Returns

true if the file is successfully loaded into the cache

`void imrcp.store.SpatialFileCache.process (String[] sMessage)`

Called when a message from `imrcp.system.Directory` is received. Attempts to load newly downloaded files into memory.

*Parameters*

<code>sMessage</code>	[BaseBlock message is from, message name, file1, file2, ..., filen]
-----------------------	---

Reimplemented from `imrcp.store.FileCache` (p.214).

`TileXCache imrcp.store.SpatialFileCache.searchXCache (int nTileX) [protected]`

Searches for the given x tile index in the list of caches, if it doesn't exist a new `TileXCache` is add to the cache list and returned.

*Parameters*

<code>nTileX</code>	X tile index to search for
---------------------	----------------------------

*Returns*

The `TileXCache` with the requested x tile index.

`boolean imrcp.store.SpatialFileCache.start () throws Exception`

Sets a schedule to execute on a fixed interval.

*Returns*

true if no Exceptions are thrown

*Exceptions*

<code>Exception</code>	
------------------------	--

Reimplemented from `imrcp.store.FileCache` (p.214).

---

## Member Data Documentation

`final int imrcp.store.SpatialFileCache.m_nZoom`

Zoom level used to spatially index files

`ArrayList<TileXCache> imrcp.store.SpatialFileCache.m_oSpatialCache = new ArrayList() [protected]`

Used to cache data files. This is a list of the x tile indices that are currently cache which contain lists of y tile indices that contain the actual files.

*See also*

`TileXCache`

`TileYCache`

---

*The documentation for this class was generated from the following file:*

- store/SpatialFileCache.java

## imrcp.store.SpatialFileWrapper Class Reference

### Public Member Functions

- abstract void **load** (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId, int nTileX, int nTileY) throws Exception
- void **load** (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId) throws Exception
- void **cleanup** (boolean bDelete)
- int **compareTo** (**FileWrapper** o)

### Public Attributes

- int **m\_nTileX**
- int **m\_nTileY**

### Additional Inherited Members

---

#### Detailed Description

FileWrappers that are spatially indexed (based on map tile indices)

#### Author

Federal Highway Administration

---

#### Member Function Documentation

##### *void imrcp.store.SpatialFileWrapper.cleanup (boolean bDelete)*

Called when a **FileWrapper** is removed from the cache in memory to clean up resources if necessary.

###### Parameters

<i>bDelete</i>	flag used to indicate if index files created when the file is loaded into memory or the file itself should be deleted or not
----------------	--

Reimplemented from **imrcp.store.FileWrapper** (p.223).

Reimplemented in **imrcp.store.MetroWrapper** (p.342).

##### *int imrcp.store.SpatialFileWrapper.compareTo (FileWrapper o)*

Compare SpatialFileWrappers by x tile index, then y tile index, then valid time in descending order, then start time, then end time

##### *void imrcp.store.SpatialFileWrapper.load (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId) throws Exception*

DO NOT USE THIS FUNCTION FOR SPATIALFILEWRAPPERS. Use **load(long, long, long, java.lang.String, int, int, int)**

Reimplemented from **imrcp.store.FileWrapper** (p.224).

##### *abstract void imrcp.store.SpatialFileWrapper.load (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId, int nTileX, int nTileY) throws Exception [abstract]*

Parses and loads observations from the given file into memory.

#### Parameters

<i>lStartTime</i>	time in milliseconds since Epoch that the file starts having observations
<i>lEndTime</i>	time in milliseconds since Epoch that the file stops having observations
<i>lValidTime</i>	time in milliseconds since Epoch that the file starts being valid (usually the time it is received)
<i>sFilename</i>	path of the file being loaded
<i>nContribId</i>	IMRCP contributor Id which is a computed by converting an up to a 6 character alphanumeric string using base 36.
<i>nTileX</i>	tile x index
<i>nTileY</i>	tile y index

#### Exceptions

<i>Exception</i>	
------------------	--

Reimplemented in **imrcp.store.MetroWrapper** (*p.343*).

---

### Member Data Documentation

*int imrcp.store.SpatialFileWrapper.m\_nTileX*

x tile index the file belongs to

*int imrcp.store.SpatialFileWrapper.m\_nTileY*

y tile index the file belongs to

---

*The documentation for this class was generated from the following file:*

- store/SpatialFileWrapper.java
- 

### imrcp.system.StringPool Class Reference

#### Classes

- class **Group**

#### Public Member Functions

- **StringPool ()**
- **String intern (String sVal)**
- **ArrayList< String > toList ()**
- **int length ()**
- **void clear ()**

#### Protected Attributes

- **char[] m\_oSearch = new char[2]**
-

## Detailed Description

Object used to store references of String for commonly used String to lower the amount of memory used.

### Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

`imrcp.system.StringPool.StringPool ()`

Default constructor. Does nothing.

---

## Member Function Documentation

`void imrcp.system.StringPool.clear ()`

Calls `ArrayList#clear()` for each group and then `ArrayList#clear()` on itself

`String imrcp.system.StringPool.intern (String sVal)`

If the given String is not in the String pool, it is added. The reference to the String from the String pool is then returned

### Parameters

<code>sVal</code>	the String to get the reference of
-------------------	------------------------------------

### Returns

The reference of the given String from the String pool

`int imrcp.system.StringPool.length ()`

Gets the number of total Strings in the String pool by adding the size of each group

### Returns

The number of Strings in the String pool

`ArrayList< String > imrcp.system.StringPool.toList ()`

Converts all of the String groups into a single list

### Returns

a list containing all of the Strings in the String pool

---

## Member Data Documentation

`char [] imrcp.system.StringPool.m_oSearch = new char[2] [protected]`

Convenience search object

---

*The documentation for this class was generated from the following file:*

- system/StringPool.java

---

## imrcp.web.Subscriptions Class Reference

### Public Member Functions

- boolean **start** () throws Exception
- final void **reset** ()
- void **doPost** (HttpServletRequest oReq, HttpServletResponse oRes) throws IOException, ServletException
- void **execute** ()
- String[] **getAvailableFiles** (String sId) throws IOException
- Path **getSubscriptionFile** (String sId, String sFileName)
- ReportSubscription **getSubscriptionByUuid** (String sUuid) throws Exception
- boolean **deleteSubscription** (String sUuid)
- int **doList** (HttpServletRequest oReq, HttpServletResponse oRes, Session oSession) throws ServletException, IOException
- int **doAdd** (HttpServletRequest oReq, HttpServletResponse oRes, Session oSession) throws ServletException, IOException
- int **doFiles** (HttpServletRequest oReq, HttpServletResponse oRes, Session oSession) throws ServletException, IOException
- int **doDownload** (HttpServletRequest oReq, HttpServletResponse oRes, Session oSession) throws ServletException, IOException
- void **listSubscriptions** (HttpServletRequest oReq, HttpServletResponse oResp, Session oSess) throws IOException, Exception
- int **retrieveSubscriptionResult** (String sUuid, String sFileName, HttpServletResponse oRes) throws IOException
- void **serializeSubSummaryDetails** (JsonGenerator oOutputGenerator, ReportSubscription oSub, boolean bShort) throws IOException

### Protected Member Functions

- JsonGenerator **createJsonGenerator** (HttpServletResponse oResp) throws IOException

### Protected Attributes

- int **m\_nOffset**
- int **m\_nPeriod**

### Additional Inherited Members

---

#### Detailed Description

This servlet manages all of the requests and processing necessary to run the Create and View Reports UIs.

#### Author

Federal Highway Administration

---

#### Member Function Documentation

*JsonGenerator imrcp.web.Subscriptions.createJsonGenerator (HttpServletResponse oResp) throws IOException [protected]*

Wrapper for **JsonFactory#createJsonGenerator (java.io.Writer)** using the response's writer

*Parameters*

<i>oResp</i>	object that contains the response the servlet sends to the client
--------------	---

*Returns*

A new JsonGenerator wrapping the writer for the response.

*Exceptions*

<i>IOException</i>
--------------------

***boolean imrcp.web.Subscriptions.deleteSubscription (String sUuid)***

Deletes the report/subscription with the given id and all of the files associated with it

*Parameters*

<i>sUuid</i>	report/subscription id
--------------	------------------------

*Returns*

true if the report/subscription is deleted successfully, otherwise false

***int imrcp.web.Subscriptions.doAdd (HttpServletRequest oReq, HttpServletResponse oRes, Session oSession) throws ServletException, IOException***

Adds the report/subscription defined in the request, if valid, into the system.

*Parameters*

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client
<i>oSession</i>	object that contains information about the user that made the request

*Returns*

HTTP status code to be included in the response.

*Exceptions*

<i>ServletException</i>	
<i>IOException</i>	

***int imrcp.web.Subscriptions.doDownload (HttpServletRequest oReq, HttpServletResponse oRes, Session oSession) throws ServletException, IOException***

Adds the report/subscription data file defined in the request to the response.

*Parameters*

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client
<i>oSession</i>	object that contains information about the user that made the request

*Returns*

HTTP status code to be included in the response.

*Exceptions*

<i>ServletExceptio n</i>	
<i>IOException</i>	

*int imrcp.web.Subscriptions.doFiles (HttpServletRequest oReq, HttpServletResponse oRes, Session oSession) throws ServletException, IOException*

Add a list of the files associated with the subscription id given in the request to the response as a JSON array.

*Parameters*

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client
<i>oSession</i>	object that contains information about the user that made the request

*Returns*

HTTP status code to be included in the response.

*Exceptions*

<i>ServletExceptio n</i>	
<i>IOException</i>	

*int imrcp.web.Subscriptions.doList (HttpServletRequest oReq, HttpServletResponse oRes, Session oSession) throws ServletException, IOException*

Adds a list of the active reports/subscriptions for the user making the request to the response as a JSON array of JSON objects.

*Parameters*

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client
<i>oSession</i>	object that contains information about the user that made the request

*Returns*

HTTP status code to be included in the response.

*Exceptions*

<i>ServletExceptio n</i>	
<i>IOException</i>	

*void imrcp.web.Subscriptions.doPost (HttpServletRequest oReq, HttpServletResponse oRes) throws IOException, ServletException*

Handles the specific logic for file download requests since users do not have to be logged in to download subscription files since they can use the direct link given when the subscription is created. If it is not a file download request calls

```
SecureBaseBlock#doPost(javax.servlet.http.HttpServletRequest
, javax.servlet.http.HttpServletResponse)
```

*Parameters*

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client

*Exceptions*

<i>IOException</i>	
<i>ServletException</i>	

Reimplemented from **imrcp.web.SecureBaseBlock** (*p.491*).

*void imrcp.web.Subscriptions.execute ()*

Iterates through the directories in the reports/subscriptions base directory and processes and fulfills any report/subscription that need to be processed.

Reimplemented from **imrcp.system.BaseBlock** (*p.95*).

*String[] imrcp.web.Subscriptions.getAvailableFiles (String sId) throws IOException*

Gets an array that contains all of the available data files for the given id of a report/subscription.

*Parameters*

<i>sId</i>	report/subscription id
------------	------------------------

*Returns*

A String array containing all of the available data files for the report/subscription. The filenames are sorted so that the most recent file is in position 0.

*Exceptions*

<i>IOException</i>	
--------------------	--

*ReportSubscription imrcp.web.Subscriptions.getSubscriptionByUuid (String sUuid)*  
*throws Exception*

Gets the **ReportSubscription** with the given id

*Parameters*

<i>sUuid</i>	report/subscription id
--------------	------------------------

*Returns*

The **ReportSubscription** with the given id

*Exceptions*

<i>Exception</i>	an Exception will be thrown if a <b>ReportSubscription</b> with the given id doesn't exist or if there is an error when reading its configuration file.
------------------	---

*Path imrcp.web.Subscriptions.getSubscriptionFile (String sId, String sFileName)*

Gets the Path of the given filename associated with the given id

*Parameters*

<i>sId</i>	report/subscription id
<i>sFileName</i>	data file name

*Returns*

Path representing the data file name that matches the id and filename, null if the file does not exist

*void imrcp.web.Subscriptions.listSubscriptions (HttpServletRequest oReq, HttpServletResponse oResp, Session oSess) throws IOException, Exception*

Adds the active reports/subscriptions that belong to the user to the response as a JSON array of JSON objects.

*Parameters*

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oResp</i>	object that contains the response the servlet sends to the client
<i>oSess</i>	object that contains information about the user that made the request

*Exceptions*

<i>IOException</i>	
<i>Exception</i>	

*final void imrcp.web.Subscriptions.reset ()*

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.web.SecureBaseBlock** (p.492).

*int imrcp.web.Subscriptions.retrieveSubscriptionResult (String sUuid, String sFileName, HttpServletResponse oRes) throws IOException*

Adds the report/subscription data file with the given id and file name to the response.

*Parameters*

<i>sUuid</i>	report/subscription id
<i>sFileName</i>	file name of the data file
<i>oRes</i>	object that contains the response the servlet sends to the client

*Returns*

HTTP status code to be included in the response.

*Exceptions*

<i>IOException</i>	

*void imrcp.web.Subscriptions.serializeSubSummaryDetails (JsonGenerator oOutputGenerator, ReportSubscription oSub, boolean bShort) throws IOException*

Writes the fields of the report/subscription to the given JSONGenerator.

#### Parameters

<i>oOutputGenerator</i>	JSONGenerator to write the fields to
<i>oSub</i>	report/subscription to serialize
<i>bShort</i>	flag indicating if all fields are written (true) or just the name, description, uuid, format, and if it is a report or subscription (false)

#### Exceptions

<i>IOException</i>
--------------------

*boolean imrcp.web.Subscriptions.start () throws Exception*

Creates the necessary directories for report/subscription files and then sets a schedule to execute on a fixed interval.

#### Returns

true if no Exceptions are thrown

#### Exceptions

<i>Exception</i>
------------------

Reimplemented from **imrcp.system.BaseBlock** (p.99).

---

## Member Data Documentation

*int imrcp.web.Subscriptions.m\_nOffset [protected]*

Schedule offset from midnight in seconds

*int imrcp.web.Subscriptions.m\_nPeriod [protected]*

Period of execution in seconds

---

*The documentation for this class was generated from the following file:*

- web/Subscriptions.java

---

## imrcp.forecast.mlp.SVMRecord Class Reference

### Public Member Functions

- **void write** (Writer oOut) throws IOException
- **int compareTo** (SVMRecord o)

### Public Attributes

- **OsmWay m\_oWay**
- **long m\_lStartDate**
- **long m\_lForecastDate**
- **int m\_nStatusHur**
- **int m\_nLatHur**
- **int m\_nLonHur**

- int **m\_nMaxSpeed**
- int **m\_nMinPressure**
- int **m\_nNewLabel**
- int **m\_nCategory**
- int **m\_nLocation**
- String **m\_sHurricane**
- double **m\_dDistToHur**
- double[] **m\_dDistances**

#### Static Public Attributes

- static String **HEADER** =
   
"linkid,onlydate,t\_start,t\_period,DayOfWeek,direction,ref,lat,lon,length,StatusHur,LatHur,LonHur,MaxSpeed,MinPressure,Timestamp,new\_label,linkid\_date,DayAftWarn,category,location,hurricane,distance,d1,d2,d3,d4,d5,d6,d7"
- 

#### Detailed Description

Contains the data needed for a single record in the input file for the Support Vector Machine Model of the MLP Hurricane model.

#### Author

Federal Highway Administration

---

#### Member Function Documentation

*int imrcp.forecast.mlp.SVMRecord.compareTo (SVMRecord o)*

Compares SVMRecords by Id of the OsmWay and forecast timestamp

#### See also

java.lang.Comparable::compareTo(java.lang.Object)

*void imrcp.forecast.mlp.SVMRecord.write (Writer oOut) throws IOException*

Writes the SMVRecord as a CSV line to the given Writer

#### Parameters

<i>oOut</i>	Writer to write the record to
-------------	-------------------------------

#### Exceptions

<i>IOException</i>
--------------------

---

#### Member Data Documentation

*String imrcp.forecast.mlp.SVMRecord.HEADER =*

*"linkid,onlydate,t\_start,t\_period,DayOfWeek,direction,ref,lat,lon,length,StatusHur,LatHur,LonHur,MaxSpeed,MinPressure,Timestamp,new\_label,linkid\_date,DayAftWarn,category,location,hurricane,distance,d1,d2,d3,d4,d5,d6,d7" [static]*

Header of the CSV file

*double [] imrcp.forecast.mlp.SVMRecord.m\_dDistances*

Distance from the midpoint of **m\_oWay** to the 7 cities identified in the MLP Hurricane Model.

[distance to Lake Charles, distance to Lafayette, distance to Baton Rouge, distance to New Orleans, distance to Alexandria, distance to Shreveport, distance to Monroe]

*double imrcp.forecast.mlp.SVMRecord.m\_dDistToHur*

Distance from the midpoint of **m\_oWay** to the eye of the hurricane in meters

*long imrcp.forecast.mlp.SVMRecord.m\_lForecastDate*

Timestamp of the hurricane forecast in milliseconds since Epoch

*long imrcp.forecast.mlp.SVMRecord.m\_lStartDate*

Timestamp at midnight in milliseconds since Epoch of the first day a hurricane warning was issued for the current storm

*int imrcp.forecast.mlp.SVMRecord.m\_nCategory*

The Saffir-Simpson hurricane wind scale (SSHWS) category of the hurricane

*int imrcp.forecast.mlp.SVMRecord.m\_nLatHur*

Predicted latitude of the hurricane at the timestamp of this record in decimal degrees scaled to 7 decimal places

*int imrcp.forecast.mlp.SVMRecord.m\_nLonHur*

Predicted longitude of the hurricane at the timestamp of this record in decimal degrees scaled to 7 decimal places

*int imrcp.forecast.mlp.SVMRecord.m\_nMaxSpeed*

Maximum predicted wind speed of the hurricane

*int imrcp.forecast.mlp.SVMRecord.m\_nMinPressure*

Minimum predicted pressure of the hurricane

*int imrcp.forecast.mlp.SVMRecord.m\_nNewLabel*

A label that was used in the training of SVM model, not needed in real time predictions

*int imrcp.forecast.mlp.SVMRecord.m\_nStatusHur*

1, 2, or 3 depending on the predicted storm type. 3 = HU or MH (Hurricane or Major Hurricane) 2 = TS (Tropical Storm) 1 = anything else

*OsmWay imrcp.forecast.mlp.SVMRecord.m\_oWay*

Roadway segment associated with the record

*String imrcp.forecast.mlp.SVMRecord.m\_sHurricane*

Storm name of the hurricane

---

*The documentation for this class was generated from the following file:*

- forecast/mlp/SVMRecord.java
-

## imrcp.system.Text Class Reference

### Static Public Member Functions

- static void **removeWhitespace** (StringBuilder sBuffer)
- static long **parseLong** (CharSequence iCharSeq)
- static long **parseLong** (CharSequence iCharSeq, int nPos, int nEndPos) throws NumberFormatException
- static int **parseInt** (CharSequence iCharSeq)
- static int **parseInt** (CharSequence iCharSeq, int nPos, int nEndPos) throws NumberFormatException
- static double **parseDouble** (CharSequence iCharSeq)
- static int **compare** (CharSequence iSeqL, CharSequence iSeqR)
- static int **compareIgnoreCase** (CharSequence iSeqL, CharSequence iSeqR)
- static boolean **startsWith** (CharSequence iSource, CharSequence iPrefix)
- static boolean **endsWith** (CharSequence iSource, CharSequence iSuffix)
- static String **toHexString** (byte[] yBytes)
- static void **toHexString** (byte[] yBytes, StringBuilder sBuf)
- static String **toHexString** (byte[] yBytes, int nOffset, int nLength)
- static void **toHexString** (byte[] yBytes, int nOffset, int nLength, StringBuilder sBuf)
- static byte[] **fromHexString** (StringBuilder sBuf)
- static void **replaceAll** (StringBuilder sBuffer, String sSearch, String sReplace)
- static int **getBytes** (byte[] yBuffer, CharSequence iCharSeq)
- static String **truncate** (String sValue, int nLength)
- static boolean **isEmpty** (CharSequence iSeq)

---

### Detailed Description

Provides methods to parse strings to extract numerical values. Also contains methods to format and compare character sequences.

---

### Member Function Documentation

**static int imrcp.system.Text.compare (CharSequence iSeqL, CharSequence iSeqR) [static]**

Lexicographically compare two character sequences. Using the character sequence interface enables the mixing of comparisons between `String`, `StringBuffer`, and `StringBuilder` objects. The character values at each index of the sequences is compared up to the minimum number of available characters. The sequence lengths determine the comparison when the contents otherwise appear to be equal.

#### Parameters

<i>iSeqL</i>	the first character sequence to be compared
<i>iSeqR</i>	the second character sequence to be compared

#### Returns

a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second

**static int imrcp.system.Text.compareIgnoreCase (CharSequence iSeqL, CharSequence iSeqR) [static]**

Lexicographically compare two character sequences. Comparison is performed without regard to differing character case.

*Parameters*

<i>iSeqL</i>	the first character sequence to be compared
<i>iSeqR</i>	the second character sequence to be compared

*Returns*

a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second

**static boolean imrcp.system.Text.endsWith (CharSequence iSource, CharSequence iSuffix) [static]**

Tests if the source character sequence ends with the suffix character sequence.

*Parameters*

<i>iSource</i>	the character sequence to check
<i>iSuffix</i>	the search character sequence

*Returns*

true if the characters at the end of the source match the characters in the suffix. false otherwise.

**static byte[] imrcp.system.Text.fromHexString (StringBuilder sBuf) [static]**

Converts a hexadecimal sequence into a byte array

*Parameters*

<i>sBuf</i>	String buffer holding hexadecimal characters.
-------------	---

*Returns*

A byte array containing the interpreted bytes.

**static int imrcp.system.Text.getBytes (byte[] yBuffer, CharSequence iCharSeq) [static]**

Copies the contents of a character sequence into the provided byte buffer. No locale translation is performed. This is mostly useful for UTF-8 and ASCII encoded strings.

*Parameters*

<i>yBuffer</i>	The byte buffer where characters are copied.
<i>iCharSeq</i>	The sequence of characters to convert to bytes.

*Returns*

The number of bytes copied into the buffer.

**static boolean imrcp.system.Text.isEmpty (CharSequence iSeq) [static]**

Determines if the CharSequence is empty.

*Parameters*

<i>iSeq</i>	CharSequence to test
-------------	----------------------

*Returns*

true if the CharSequence is null or its length is 0

*static double imrcp.system.Text.parseDouble (CharSequence iCharSeq) [static]*

Converts the character sequence into a double value.

*Parameters*

<i>iCharSeq</i>	a set of characters to be converted into a double value
-----------------	---

*Returns*

the converted double value.

*static int imrcp.system.Text.parseInt (CharSequence iCharSeq) [static]*

Wraps `Text#parseInt(java.lang.CharSequence, int, int)` to convert the sequence from beginning to end.

*Parameters*

<i>iCharSeq</i>	a set of characters to be converted into an integer value
-----------------	---

*Returns*

the decimal integer value represented by the character sequence.

*static int imrcp.system.Text.parseInt (CharSequence iCharSeq, int nPos, int nEndPos) throws NumberFormatException [static]*

Parses the character sequence argument as a signed decimal integer. The characters in the sequence must all be decimal digits, except that the first character may be an ASCII minus sign '-' ('\u002D') to indicate a negative value.

*Parameters*

<i>iCharSeq</i>	a set of characters to be converted into an integer value
<i>nPos</i>	the position in the sequence where conversion begins
<i>nEndPos</i>	the sequence position where conversion stops (exclusive)

*Returns*

the decimal integer value represented by the character sequence

*Exceptions*

<i>NumberFormatException</i>	if the character sequence does not contain characters that can be converted to a decimal integer
------------------------------	--

*static long imrcp.system.Text.parseLong (CharSequence iCharSeq) [static]*

Wraps `Text#parseInt(java.lang.CharSequence, int, int)` to convert the sequence from beginning to end.

*Parameters*

<i>iCharSeq</i>	a set of characters to be converted into a long value
-----------------	---

*Returns*

the decimal long value represented by the character sequence.

*static long imrcp.system.Text.parseLong (CharSequence iCharSeq, int nPos, int nEndPos) throws NumberFormatException [static]*

Parses the character sequence argument as a signed decimal long. The characters in the sequence must all be decimal digits, except that the first character may be an ASCII minus sign '-' ('\u002D') to indicate a negative value.

*Parameters*

<i>iCharSeq</i>	a set of characters to be converted into an integer value
<i>nPos</i>	the position in the sequence where conversion begins
<i>nEndPos</i>	the sequence position where conversion stops (exclusive)

*Returns*

the decimal long value represented by the character sequence

*Exceptions*

<i>NumberFormatException</i>	if the character sequence does not contain characters that can be converted to a decimal long
------------------------------	---

*static void imrcp.system.Text.removeWhitespace (StringBuilder sBuffer) [static]*

Removes whitespace from the provided string builder.

*Parameters*

<i>sBuffer</i>	the string to remove whitespace from.
----------------	---------------------------------------

*static void imrcp.system.Text.replaceAll (StringBuilder sBuffer, String sSearch, String sReplace) [static]*

Replaces all occurrences of the search string within the supplied buffer with the replacement string.

*Parameters*

<i>sBuffer</i>	StringBuilder buffer containing text to be searched.
<i>sSearch</i>	Search string to find in the buffer.
<i>sReplace</i>	Replacement string to substitute for the search string.

*static boolean imrcp.system.Text.startsWith (CharSequence iSource, CharSequence iPrefix) [static]*

Tests if the source character sequence begins with the prefix character sequence.

*Parameters*

<i>iSource</i>	the character sequence to check
<i>iPrefix</i>	the search character sequence

*Returns*

`true` if the initial source characters match the characters in the prefix. `false` otherwise.

*static String imrcp.system.Text.toHexString (byte[] yBytes) [static]*

Converts a byte array into a hexadecimal string

**Parameters**

<i>yByte</i>	Byte array containing data to be converted.
--------------	---

**Returns**

Hexadecimal string that represents the supplied byte data.

**static String imrcp.system.Text.toHexString (byte[] yBytes, int nOffset, int nLength) [static]**

Converts a byte array into a hexadecimal string

**Parameters**

<i>yBytes</i>	Byte array containing data to be converted.
<i>nOffset</i>	The position within the array to begin converting.
<i>nLength</i>	The number of bytes to be converted.

**Returns**

Hexadecimal string that represents the supplied byte data.

**static void imrcp.system.Text.toHexString (byte[] yBytes, int nOffset, int nLength, StringBuilder sBuf) [static]**

Converts a byte array into a hexadecimal string

**Parameters**

<i>yBytes</i>	Byte array containing data to be converted.
<i>nOffset</i>	The position within the array to begin converting.
<i>nLength</i>	The number of bytes to be converted.
<i>sBuf</i>	String buffer that holds the converted characters.

**static void imrcp.system.Text.toHexString (byte[] yBytes, StringBuilder sBuf) [static]**

Converts a byte array into a hexadecimal string

**Parameters**

<i>yBytes</i>	Byte array containing data to be converted.
<i>sBuf</i>	String buffer that holds the converted characters.

**static String imrcp.system.Text.truncate (String sValue, int nLength) [static]**

Truncates the passed string value if it is longer than the passed length, or returns the original value if it isn't.

**Parameters**

<i>sValue</i>	The value to truncate
<i>nLength</i>	The maximum length of the truncated string.

**Returns**

The truncated string

The documentation for this class was generated from the following file:

- system/Text.java
- 

## imrcp.web.tiles.Tile Class Reference

### Public Member Functions

- `Tile (int nX, int nY, int nZ)`
- `int compareTo (int[] o)`
- `void createAreas (Mercator oM, int nX, int nY)`
- `int compare (double[] o1, double[] o2)`

### Public Attributes

- `int m_nX`
- `int m_nY`
- `int m_nZoom`
- `ArrayList< TileArea > m_oAreas`

### Protected Member Functions

- `Tile ()`
- 

### Detailed Description

Represents a map tile that contains polygons. The polygons are created by the run length encoding algorithm found in `TileWrapper#createTileList(imrcp.store.EntryData, int)` so that are rectangular strips created from gridded data.

#### Author

Federal Highway Administration

---

### Constructor & Destructor Documentation

#### `imrcp.web.tiles.Tile.Tile () [protected]`

Default constructor. Does nothing.

#### `imrcp.web.tiles.Tile.Tile (int nX, int nY, int nZ)`

Constructs a `Tile` with the given map tile coordinates

#### Parameters

<code>nX</code>	map tile x index
<code>nY</code>	map tile y index
<code>nZ</code>	map tile zoom level

---

## Member Function Documentation

`int imrcp.web.tiles.Tile.compare (double[] o1, double[] o2)`

C.compares the two double array which represent rings by group value (pos 0) then max lat (pos 2) then min lon (pos 1).

`int imrcp.web.tiles.Tile.compareTo (int[] o)`

C.compares Tiles by x index, then y index

`void imrcp.web.tiles.Tile.createAreas (Mercator oM, int nX, int nY)`

C.createAndFills m\_oAreas with TileArea objects by clipping the polygons that intersect this map tile (stored as double[] in this)

### Parameters

<code>oM</code>	Mercator object
<code>nX</code>	map tile x index
<code>nY</code>	map tile y index

---

## Member Data Documentation

`int imrcp.web.tiles.Tile.m_nX`

x index of map tile

`int imrcp.web.tiles.Tile.m_nY`

y index of map tile

`int imrcp.web.tiles.Tile.m_nZoom`

Zoom level of map tile

`ArrayList<TileArea> imrcp.web.tiles.Tile.m_oAreas`

Stores **TileArea** which contain the geometric definitions of the polygons that intersect the tile, clipped by the bounds of the tile.

---

*The documentation for this class was generated from the following file:*

- web/tiles/Tile.java
- 

## imrcp.web.tiles.TileArea Class Reference

### Public Member Functions

- `TileArea ()`
- `TileArea (Path2D.Double oPath, double dVal)`
- `int compareTo (TileArea o)`

### Public Attributes

- `double m_dGroupValue`
-

## Detailed Description

Extends the Area class to include a group value for polygons. This class is used to aid in clipping polygons with the boundaries of map tiles.

### Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

### `imrcp.web.tiles.TileArea.TileArea ()`

Default constructor. Wrapper for `Area#Area ()`

### `imrcp.web.tiles.TileArea.TileArea (Path2D.Double oPath, double dVal)`

Constructs a `TileArea` with the given group value and the geometry defined by the given Path by calling `Area#Area (java.awt.Shape)`

#### Parameters

<code>oPath</code>	Path object defining the geometry of the polygon
<code>dVal</code>	group value

---

## Member Function Documentation

### `int imrcp.web.tiles.TileArea.compareTo (TileArea o)`

Compares TileAreas by group value and then `java.lang.Object#hashCode ()`

---

## Member Data Documentation

### `double imrcp.web.tiles.TileArea.m_dGroupValue`

Group value of the polygon, used for presentation on the map

---

*The documentation for this class was generated from the following file:*

- web/tiles/TileArea.java

---

## imrcp.geosrv.osm.TileBucket Class Reference

### Public Member Functions

- `TileBucket ()`
- `TileBucket (int nX, int nY)`
- `int compareTo (TileBucket o)`

### Public Attributes

- `int m_nX`
- `int m_nY`

## Detailed Description

This class stores roadway segments that intersect the map tile it represents. Storing roadway segments in this way acts as a spatial index to allow quick look up of roadway segments.

### Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

`imrcp.geosrv.osm.TileBucket.TileBucket ()`

Wrapper for `ArrayList#ArrayList ()`

`imrcp.geosrv.osm.TileBucket.TileBucket (int nX, int nY)`

Constructs a **TileBucket** with the given map tile indices

### Parameters

<code>nX</code>	map tile x index
<code>nY</code>	map tile y index

---

## Member Function Documentation

`int imrcp.geosrv.osm.TileBucket.compareTo (TileBucket o)`

Compare TileBuckets by x index then y index

---

## Member Data Documentation

`int imrcp.geosrv.osm.TileBucket.m_nX`

Map tile x index

`int imrcp.geosrv.osm.TileBucket.m_nY`

Map tile y index

---

*The documentation for this class was generated from the following file:*

- geosrv/osm/TileBucket.java

---

## imrcp.web.tiles.TileCache Class Reference

### Public Member Functions

- boolean **loadFileToMemory** (String sFullPath, int nFormatIndex)
- void **reset** ()
- boolean **start** ()
- int **doMvt** (HttpServletRequest oRequest, HttpServletResponse oResponse, Session oSess) throws ServletException, IOException
- **GriddedFileWrapper** **getNewFileWrapper** ()

## Protected Member Functions

- abstract **GriddedFileWrapper getDataWrapper** (int nFormatIndex)

## Protected Attributes

- int **m\_nHashZoom**
- int **m\_nTileObsType**
- int **m\_nCheckInterval**
- long **m\_lCheckTimeout** = 0
- int **m\_nLayerMultiplier**

## Additional Inherited Members

---

### Detailed Description

FileCache implementation that manages **TileWrapper**s. It responsible for fulfilling IMRCP Map UI requests for the geometries of Area Layer polygons.

#### Author

Federal Highway Administration

---

### Member Function Documentation

*int imrcp.web.tiles.TileCache.doMvt (HttpServletRequest oRequest,  
HttpServletResponse oResponse, Session oSess) throws ServletException, IOException*

This method handles requests for .mvt (Mapbox Vector Tile) files by generating and caching the polygons that intersect the requested map tile

#### Parameters

<i>oRequest</i>	object that contains the request the client has made of the servlet
<i>oResponse</i>	object that contains the response the servlet sends to the client
<i>oSess</i>	object that contains information about the user that made the request

#### Returns

HTTP status code to be included in the response.

#### Exceptions

<i>ServletException</i>	
<i>n</i>	

<i>IOException</i>	
--------------------	--

*abstract GriddedFileWrapper imrcp.web.tiles.TileCache.getDataWrapper (int nFormatIndex) [abstract], [protected]*

Child classes implement this to get the correctly configured FileWrapper for getting data to generate the polygons grouped by value.

#### Parameters

<i>nFormatIndex</i>	index to use for <b>m_oFormatters</b>
---------------------	---------------------------------------

*Returns*

a new GriddedFileWrapper containing the data needed to generate polygons grouped by value.

Reimplemented in **imrcp.web.tiles.DataObsTileCache** (p.130), **imrcp.web.tiles.GribTileCache** (p.250), and **imrcp.web.tiles.NcfTileCache** (p.372).

*GriddedFileWrapper imrcp.web.tiles.TileCache.getNewFileWrapper ()*

*Returns*

a new **TileWrapper**

Reimplemented from **imrcp.store.FileCache** (p.212).

*boolean imrcp.web.tiles.TileCache.loadFileToMemory (String sFullPath, int nFormatIndex)*

Loads the given data file into memory to create a **TileWrapper** based off of the data in that file for the configured observation type.

Reimplemented from **imrcp.store.FileCache** (p.213).

*void imrcp.web.tiles.TileCache.reset ()*

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.store.FileCache** (p.214).

Reimplemented in **imrcp.web.tiles.GribTileCache** (p.250), and **imrcp.web.tiles.NcfTileCache** (p.372).

*boolean imrcp.web.tiles.TileCache.start ()*

Does nothing. Implemented so **FileCache#start()** is not called.

*Returns*

true

Reimplemented from **imrcp.store.FileCache** (p.214).

---

**Member Data Documentation**

*long imrcp.web.tiles.TileCache.m\_lCheckTimeout = 0 [protected]*

Time in milliseconds since Epoch to start checking for a "better" file

*int imrcp.web.tiles.TileCache.m\_nCheckInterval [protected]*

Time in milliseconds after loading a file to check if a "better" file exists

*int imrcp.web.tiles.TileCache.m\_nHashZoom [protected]*

The map zoom level used when creating **Tile** objects

*int imrcp.web.tiles.TileCache.m\_nLayerMultiplier [protected]*

Value to multiply group values by when creating layer names. Default is 1. This is used when multiple group values would be rounded to the same integer since that would yield the same layer name for the tile.

*int imrcp.web.tiles.TileCache.m\_nTileObsType [protected]*

IMRCP observation type id of the data the polygons represent

---

*The documentation for this class was generated from the following file:*

- web/tiles/TileCache.java
- 

## imrcp.web.tiles.TileServlet Class Reference

### Public Member Functions

- void **init** (ServletConfig oSConfig) throws ServletException
- void **reset** ()
- int **doMvt** (HttpServletRequest oRequest, HttpServletResponse oResponse, Session oSession) throws ServletException, IOException

### Protected Member Functions

- void **doLinestring** (int nZ, int nX, int nY, long lTimestamp, long lRefTime, int nRequestType, HttpServletResponse oResponse, Session oSession) throws IOException
- void **doPoint** (int nZ, int nX, int nY, long lTimestamp, long lRefTime, int nRequestType, String sRangeString, HttpServletResponse oResponse, Session oSession) throws IOException
- void **doCap** (int nZ, int nX, int nY, long lTimestamp, long lRefTime, int nRequestType, HttpServletResponse oResponse) throws IOException

### Protected Attributes

- String[] **m\_sKeys** = new String[]{"roadtype", "bridge"}
- String[] **m\_sValues** = new String[]{"H", "A", "Y", "N"}
- int **m\_nMinArterialZoom**
- HashMap< Integer, Integer > **m\_oObsTypeZoom** = new HashMap()
- int **m\_nMinZoom**

### Static Protected Attributes

- static CAPStore **CAPSTORE**
- static ObsView **OBSVIEW**
- static WayNetworks **WAYS**
- static Comparator< double[]> **LINECOMP** = (double[] o1, double[] o2) -> {return Double.compare(o1[0], o2[0]);}
- static int[] **POINTREQUESTS**
- static final int **RNM** = Integer.valueOf("rnm", 36)

### Additional Inherited Members

---

### Detailed Description

This class is responsible for fulfilling IMRCP Map UI requests for the geometries and locations of Road Layer and Points Layer objects.

### Author

Federal Highway Administration

---

## Member Function Documentation

`void imrcp.web.tiles.TileServlet.doCap (int nZ, int nX, int nY, long lTimestamp, long lRefTime, int nRequestType, HttpServletResponse oResponse) throws IOException [protected]`

Add the vector tile for the given request parameters to the response

### Parameters

<code>nZ</code>	request map tile zoom level
<code>nX</code>	request map tile x index
<code>nY</code>	request map tile y index
<code>lTimestamp</code>	query time in milliseconds since Epoch
<code>lRefTime</code>	reference time in milliseconds since Epoch
<code>nRequestType</code>	requested IMRCP observation type id
<code>oResponse</code>	object that contains the response the servlet sends to the client

### Exceptions

<code>IOException</code>
--------------------------

`void imrcp.web.tiles.TileServlet.doLinestring (int nZ, int nX, int nY, long lTimestamp, long lRefTime, int nRequestType, HttpServletResponse oResponse, Session oSession) throws IOException [protected]`

Add the vector tile for the given request parameters to the response

### Parameters

<code>nZ</code>	request map tile zoom level
<code>nX</code>	request map tile x index
<code>nY</code>	request map tile y index
<code>lTimestamp</code>	query time in milliseconds since Epoch
<code>lRefTime</code>	reference time in milliseconds since Epoch
<code>nRequestType</code>	requested IMRCP observation type id
<code>oResponse</code>	object that contains the response the servlet sends to the client
<code>oSession</code>	object that contains information about the user that made the request

### Exceptions

<code>IOException</code>
--------------------------

`int imrcp.web.tiles.TileServlet.doMvt (HttpServletRequest oRequest, HttpServletResponse oResponse, Session oSession) throws ServletException, IOException`

This method handles requests for .mvt (Mapbox Vector **Tile**) files by generating points and linestrings that intersect the requested map tile

### Parameters

<code>oRequest</code>	object that contains the request the client has made of the servlet
<code>oResponse</code>	object that contains the response the servlet sends to the client

<i>oSession</i>	object that contains information about the user that made the request
-----------------	---

*Returns*

HTTP status code to be included in the response.

*Exceptions*

<i>ServletException</i>	
<i>IOException</i>	

***void imrcp.web.tiles.TileServlet.doPoint (int nZ, int nX, int nY, long lTimestamp, long lRefTime, int nRequestType, String sRangeString, HttpServletResponse oResponse, Session oSession) throws IOException [protected]***

Add the vector tile for the given request parameters to the response

*Parameters*

<i>nZ</i>	request map tile zoom level
<i>nX</i>	request map tile x index
<i>nY</i>	request map tile y index
<i>lTimestamp</i>	query time in milliseconds since Epoch
<i>lRefTime</i>	reference time in milliseconds since Epoch
<i>nRequestType</i>	requested IMRCP observation type id
<i>sRangeString</i>	used for <b>ObsType.EVT</b> . Range of values that are valid for this request
<i>oResponse</i>	object that contains the response the servlet sends to the client
<i>oSession</i>	object that contains information about the user that made the

*Exceptions*

<i>IOException</i>
--------------------

***void imrcp.web.tiles.TileServlet.init (ServletConfig oSConfig) throws ServletException***

Initializes the servlet by parsing both the servlet configuration and the IMRCP configuration.

Reimplemented from **imrcp.web.SecureBaseBlock** (p.491).

***void imrcp.web.tiles.TileServlet.reset ()***

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.web.SecureBaseBlock** (p.492).

## Member Data Documentation

***CAPStore imrcp.web.tiles.TileServlet.CAPSTORE [static], [protected]***

Reference to the store that manages CAP alerts

*Comparator<double[]> imrcp.web.tiles.TileServlet.LINECOMP = (double[] o1, double[] o2) -> {return Double.compare(o1[0], o2[0]);} [static], [protected]*

Comparator used to compare double[] that represent the linestring that need to be added to tiles by index 0 which is the group value

*int imrcp.web.tiles.TileServlet.m\_nMinArterialZoom [protected]*

The first zoom level arterial roadway segments are included in the vector tiles sent as responses.

*int imrcp.web.tiles.TileServlet.m\_nMinZoom [protected]*

Default minimum zoom level for observation types to be included in the vector tiles sent as responses.

*HashMap<Integer, Integer> imrcp.web.tiles.TileServlet.m\_oObsTypeZoom = new  
HashMap() [protected]*

Maps observation type ids to a zoom level that is the minimum zoom level that observation type will be included in the vector tiles sent as responses

*String [] imrcp.web.tiles.TileServlet.m\_sKeys = new String[]{"roadtype",  
"bridge"} [protected]*

Array of keys that get added to vector tiles

*String [] imrcp.web.tiles.TileServlet.m\_sValues = new String[]{"H", "A", "Y",  
"N"} [protected]*

Array of values that get added to vector tiles

*ObsView imrcp.web.tiles.TileServlet.OBSVIEW [static], [protected]*

Reference to ObsView which is used to make data queries from all the data stores.

*int [] imrcp.web.tiles.TileServlet.POINTREQUESTS [static], [protected]*

Contains IMRCP observation type ids that are used for Points Layer requests

*final int imrcp.web.tiles.TileServlet.RNM = Integer.valueOf("rnm", 36) [static],  
[protected]*

Observation type id that stands for Road Network Model. Used for Road Layer requests that only need the geometry of the segments, not any associated data.

*WayNetworks imrcp.web.tiles.TileServlet.WAYS [static], [protected]*

Reference to the object to look up roadway segments

---

*The documentation for this class was generated from the following file:*

- web/tiles/TileServlet.java

---

## imrcp.web.tiles.TileUtil Class Reference

### Static Public Member Functions

- static int **command** (int nId, int nCount)
- static int **parameter** (double dValue)

- static int **getPos** (double dVal, double dMin, double dMax, double dExtent, boolean bInvert)
- static void **addPolygon** (VectorTile.Tile.Feature.Builder oFeatureBuilder, int[] nCur, double[] dMercBounds, int nExtent, Area oPoly, int[] nPointBuffer)
- static void **addLinestring** (VectorTile.Tile.Feature.Builder oFeatureBuilder, int[] nCur, double[] dMercBounds, int nExtent, double[] dLine, int[] nPointBuffer, int... nTags)
- static void **addPointToFeature** (VectorTile.Tile.Feature.Builder oFeatureBuilder, int[] nCur, double[] dMercBounds, int nExtent, double dLon, double dLat)
- static void **writePointBuffer** (VectorTile.Tile.Feature.Builder oFeatureBuilder, int[] nPointBuffer, int[] nCur, boolean bClose)
- static int[] **newAddLinestring** (VectorTile.Tile.Feature.Builder oFeatureBuilder, int[] nCur, double[] dBounds, int nExtent, double[] dLine, int[] nPointBuffer)
- static int[] **newAddLinestring** (VectorTile.Tile.Feature.Builder oFeatureBuilder, int[] nCur, double[] dBounds, int nExtent, int nStartPos, double[] dLine, int[] nPointBuffer)
- static void **newWritePointBuffer** (VectorTile.Tile.Feature.Builder oFeatureBuilder, int[] nPointBuffer, int[] nCur, boolean bClose)

### Static Public Attributes

- static final int **MOVETO** = 1
- static final int **LINETO** = 2
- static final int **CLOSEPATH** = 7

### Detailed Description

Contains methods to aid in the creation of Mapbox Vector Tiles(MVT). See the Mapbox Vector Tile Specification at <https://github.com/mapbox/vector-tile-spec/tree/master/2.1>

### Author

Federal Highway Administration

### Member Function Documentation

*static void imrcp.web.tiles.TileUtil.addLinestring (VectorTile.Tile.Feature.Builder oFeatureBuilder, int[] nCur, double[] dMercBounds, int nExtent, double[] dLine, int[] nPointBuffer, int... nTags) [static]*

Adds the linestring defined by the given double array as a Feature to the FeatureBuilder.

#### Parameters

<i>oFeatureBuilder</i>	Feature Builder for the VectorTile
<i>nCur</i>	reusable array that contains the position of the cursor which is the current position in tile coordinates
<i>dMercBounds</i>	bounds in mercator meters of the MVT. [min x, min y, max x, max y]
<i>nExtent</i>	number of positions to use along each axis inside the tile
<i>dLine</i>	array that defines the linestring. [group value, id, value index for highway, value index for bridge, x0, y0, x1, y1... xn, yn]
<i>nPointBuffer</i>	reusable growable array that stores the tile coordinates that correspond to the points of the linestring
<i>nTags</i>	not implemented

```
static void imrcp.web.tiles.TileUtil.addPointToFeature (VectorTile.Tile.Feature.Builder
oFeatureBuilder, int[] nCur, double[] dMercBounds, int nExtent, double dLon,
double dLat) [static]
```

Adds the point defined by the given longitude and latitude as a Feature to the FeatureBuilder.

#### Parameters

<i>oFeatureBuilder</i>	Feature Builder for the VectorTile
<i>nCur</i>	reusable array that contains the position of the cursor which is the current position in tile coordinates
<i>dMercBounds</i>	bounds in mercator meters of the MVT. [min x, min y, max x, max y]
<i>nExtent</i>	number of positions to use along each axis inside the tile
<i>dLon</i>	longitude of the point in decimal degrees
<i>dLat</i>	latitude of the point in decimal degrees

```
static void imrcp.web.tiles.TileUtil.addPolygon (VectorTile.Tile.Feature.Builder
oFeatureBuilder, int[] nCur, double[] dMercBounds, int nExtent, Area oPoly, int[]
nPointBuffer) [static]
```

Adds the polygon defined by the given Area as a Feature to the FeatureBuilder.

#### Parameters

<i>oFeatureBuilder</i>	Feature Builder for the VectorTile
<i>nCur</i>	reusable array that contains the position of the cursor which is the current position in tile coordinates
<i>dMercBounds</i>	bounds in mercator meters of the MVT. [min x, min y, max x, max y]
<i>nExtent</i>	number of positions to use along each axis inside the tile
<i>oPoly</i>	Polygon with coordinates in mercator meters
<i>nPointBuffer</i>	reusable growable array that stores the tile coordinates that correspond to the points of the polygon

```
static int imrcp.web.tiles.TileUtil.command (int nId, int nCount) [static]
```

Calculates the CommandInteger which indicates the command to be executed in the geometry encoding of a MVT based off the given command id and command count.

#### Parameters

<i>nId</i>	the command ID. 1 = MoveTo, 2 = LineTo, 7 = ClosePath
<i>nCount</i>	The command count which indicates the number of times the command will be executed.

#### Returns

The CommandInteger

```
static int imrcp.web.tiles.TileUtil.getPos (double dVal, double dMin, double dMax,
double dExtent, boolean bInvert) [static]
```

Gets the position of the given value as a number between 0 and the extent based off of the minimum and maximum. This function is used to determine where the points of geometries are located inside of a MVT.

#### Parameters

<i>dVal</i>	coordinate to get the position of
<i>dMin</i>	minimum bound
<i>dMax</i>	maximum bound
<i>dExtent</i>	total number of positions inside of the tile to use
<i>bInvert</i>	use true for y axis, false for x axis

#### Returns

Position inside the tile of the coordinate.  $0 \leq pos < extent$

```
static int[] imrcp.web.tiles.TileUtil.newAddLinestring (VectorTile.Tile.Feature.Builder
oFeatureBuilder, int[] nCur, double[] dBounds, int nExtent, double[] dLine, int[]
nPointBuffer) [static]
```

Wrapper  
**newAddLinestring**(vector\_tile.VectorTile.Tile.Feature.Builder  
, int[], double[], int, int, double[], int[]) with a start position  
of 1.

#### Parameters

<i>oFeatureBuilder</i>	Feature Builder for the VectorTile
<i>nCur</i>	reusable array that contains the position of the cursor which is the current position in tile coordinates
<i>dBounds</i>	bounds in mercator meters of the MVT. [min x, min y, max x, max y]
<i>nExtent</i>	number of positions to use along each axis inside the tile
<i>dLine</i>	growable array that defines the linestring. [group value, id, value index for highway, value index for bridge, x0, y0, x1, y1... xn, yn]
<i>nPointBuffer</i>	reusable growable array that stores the tile coordinates that correspond to the points of the linestring

#### Returns

reference to the array that describes nPointBuffer (the reference could possibly change if more space needed to be allocated to add the points)

```
static int[] imrcp.web.tiles.TileUtil.newAddLinestring (VectorTile.Tile.Feature.Builder
oFeatureBuilder, int[] nCur, double[] dBounds, int nExtent, int nStartPos, double[]
dLine, int[] nPointBuffer) [static]
```

Adds the linestring defined by the given double array as a Feature to the FeatureBuilder.

*Parameters*

<i>oFeatureBuilder</i>	Feature Builder for the VectorTile
<i>nCur</i>	reusable array that contains the position of the cursor which is the current position in tile coordinates
<i>dBounds</i>	bounds in mercator meters of the MVT. [min x, min y, max x, max y]
<i>nExtent</i>	number of positions to use along each axis inside the tile
<i>nStartPos</i>	the position in the growable array that the coordinates of the linestring start
<i>dLine</i>	growable array that defines the linestring. [group value, id, value index for highway, value index for bridge, x0, y0, x1, y1... xn, yn]
<i>nPointBuffer</i>	reusable growable array that stores the tile coordinates that correspond to the points of the linestring

*Returns*

reference to the array that describes nPointBuffer (the reference could possibly change if more space needed to be allocated to add the points)

```
static void imrcp.web.tiles.TileUtil.newWritePointBuffer
(VectorTile.Tile.Feature.Builder oFeatureBuilder, int[] nPointBuffer, int[] nCur,
boolean bClose) [static]
```

Writes the tile coordinates in the point buffer to the FeatureBuilder. This differs from `writePointBuffer(vector_tile.VectorTile.Tile.Feature.Builder, int[], int[], boolean)` because it accumulates the deltas of the coordinate in a buffer before writing them to the Feature Builder in attempt to avoid invalid geometries

*Parameters*

<i>oFeatureBuilder</i>	Feature Builder for the VectorTile
<i>nPointBuffer</i>	reusable growable array that stores the tile coordinates that correspond to the points of the feature
<i>nCur</i>	reusable array that contains the position of the cursor which is the current position in tile coordinates
<i>bClose</i>	flag indicating if the feature needs to be closed. Should be true for polygons, otherwise false

```
static int imrcp.web.tiles.TileUtil.parameter (double dValue) [static]
```

Calculates the ParameterInteger of the given value. A ParameterInteger is a zigzag encoded so that small negative and positive values are both encoded as small integers.

*Parameters*

<i>dValue</i>	the value to encode
---------------	---------------------

*Returns*

the ParameterInteger of the value

`static void imrcp.web.tiles.TileUtil.writePointBuffer (VectorTile.Tile.Feature.Builder oFeatureBuilder, int[] nPointBuffer, int[] nCur, boolean bClose) [static]`

Writes the tile coordinates in the point buffer to the FeatureBuilder

#### Parameters

<code>oFeatureBuilder</code>	Feature Builder for the VectorTile
<code>nPointBuffer</code>	reusable growable array that stores the tile coordinates that correspond to the points of the feature
<code>nCur</code>	reusable array that contains the position of the cursor which is the current position in tile coordinates
<code>bClose</code>	flag indicating if the feature needs to be closed. Should be true for polygons, otherwise false

---

## Member Data Documentation

`final int imrcp.web.tiles.TileUtil.CLOSEPATH = 7 [static]`

Command id for ClosePath in MVT

`final int imrcp.web.tiles.TileUtil.LINETO = 2 [static]`

Command id for LineTo in MVT

`final int imrcp.web.tiles.TileUtil.MOVETO = 1 [static]`

Command id for MoveTo in MVT

---

*The documentation for this class was generated from the following file:*

- `web/tiles/TileUtil.java`
- 

## imrcp.web.tiles.TileWrapper Class Reference

### Public Member Functions

- `void load (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId)`  
throws Exception
- `ArrayList< Tile > getTileList (long lTimestamp)`
- `void cleanup (boolean bDelete)`
- `double getReading (int nObsType, long lTimestamp, int nLat, int nLon, Date oTimeRecv)`
- `void set (GriddedFileWrapper oWrapper, int nObsType, int nZoom, int nFormatIndex)`
- `void getIndices (int nLon, int nLat, int[] nIndices)`
- `double getReading (int nObsType, long lTimestamp, int[] nIndices)`

### Public Attributes

- `ArrayList< TileList > m_oTileLists = new ArrayList()`

### Protected Attributes

- `GriddedFileWrapper m_oDataWrapper`
- `int m_nZoom`
- `int m_nObsType`

## Additional Inherited Members

---

### Detailed Description

FileWrapper implementation containing the logic to create polygons for gridded data sources used for presentation of data on the IMRCP Map UI

#### Author

Federal Highway Administration

---

### Member Function Documentation

**`void imrcp.web.tiles.TileWrapper.cleanup (boolean bDelete)`**

Calls `m_oDataWrapper#cleanup (boolean)` and  
`imrcp.web.tiles.TileWrapper.TileList#clear ()` on `m_oTileLists`

#### Parameters

<code>bDelete</code>	
----------------------	--

Reimplemented from **imrcp.store.FileWrapper** (p.223).

**`void imrcp.web.tiles.TileWrapper.getIndices (int nLon, int nLat, int[] nIndices)`**

NOT IMPLEMENTED

Reimplemented from **imrcp.store.GriddedFileWrapper** (p.258).

**`double imrcp.web.tiles.TileWrapper.getReading (int nObsType, long lTimestamp, int nLat, int nLon, Date oTimeRecv)`**

NOT IMPLEMENTED

Reimplemented from **imrcp.store.GriddedFileWrapper** (p.258).

**`double imrcp.web.tiles.TileWrapper.getReading (int nObsType, long lTimestamp, int[] nIndices)`**

NOT IMPLEMENTED

Reimplemented from **imrcp.store.GriddedFileWrapper** (p.259).

**`ArrayList< Tile > imrcp.web.tiles.TileWrapper.getTileList (long lTimestamp)`**

Attempts to get the list of tiles that correspond to the given time. If that tile set does not exist yet, it is created and added to the cached tile list.

#### Parameters

<code>lTimestamp</code>	query time in milliseconds since Epoch.
-------------------------	---

#### Returns

a list of Tiles valid for the query time.

**`void imrcp.web.tiles.TileWrapper.load (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId) throws Exception`**

Sets the times and contributor id. It does not actually load all of the data into memory like other FileWrappers

Reimplemented from **imrcp.store.FileWrapper** (p.224).

`void imrcp.web.tiles.TileWrapper.set (GriddedFileWrapper oWrapper, int nObsType, int nZoom, int nFormatIndex)`

Sets the member variable to the given parameters

#### Parameters

<code>oWrapper</code>	FileWrapper containing the gridded data
<code>nObsType</code>	IMRCP observation type id the tiles created represent
<code>nZoom</code>	map zoom level used to create the polygons and tiles
<code>nFormatIndex</code>	index used to get the file name of oWrapper from <code>imrcp.web.tiles.TileCache#m_oFormatters</code> of the <b>TileCache</b> that manages this file

---

## Member Data Documentation

`int imrcp.web.tiles.TileWrapper.m_nObsType [protected]`

Observation type id this file provides

`int imrcp.web.tiles.TileWrapper.m_nZoom [protected]`

Map zoom level used to generate the tiles and polygons

`GriddedFileWrapper imrcp.web.tiles.TileWrapper.m_oDataWrapper [protected]`

File that contains the data used to generate polygons

`ArrayList<TileList> imrcp.web.tiles.TileWrapper.m_oTileLists = new ArrayList()`

List of TileLists sorted by time index

---

*The documentation for this class was generated from the following file:*

- `web/tiles/TileWrapper.java`
- 

---

## imrcp.store.SpatialFileCache.TileXCache Class Reference

### Public Member Functions

- `int compareTo (TileXCache o)`

### Protected Member Functions

- `TileYCache search (int nY)`

### Protected Attributes

- `int m_nX`
- `ReentrantReadWriteLock m_oLockX = new ReentrantReadWriteLock(true)`

---

### Detailed Description

**TileXCache** is a list of **TileYCache**. The x tile index of this **TileXCache** combined with the y tile index of the TileYCaches in this list represent the map tiles that have files cached.

## Member Function Documentation

*int imrcp.store.SpatialFileCache.TileXCache.compareTo (TileXCache o)*

Compares **TileXCache** by x index

*TileYCache imrcp.store.SpatialFileCache.TileXCache.search (int nY) [protected]*

Searches for the given y tile index in the list of caches, if it doesn't exist a new **TileYCache** is add to the cache list and returned.

*Parameters*

<i>nY</i>	y tile index to search for
-----------	----------------------------

*Returns*

The **TileYCache** with the requested y tile index.

---

## Member Data Documentation

*int imrcp.store.SpatialFileCache.TileXCache.m\_nX [protected]*

x tile index

*ReentrantReadWriteLock imrcp.store.SpatialFileCache.TileXCache.m\_oLockX = new ReentrantReadWriteLock(true) [protected]*

Read/Write lock for this list of TileYCaches

---

*The documentation for this class was generated from the following file:*

- store/SpatialFileCache.java
- 

## imrcp.store.SpatialFileCache.TileYCache Class Reference

### Public Member Functions

- int **compareTo** (TileYCache o)

### Protected Member Functions

- int **search** (long lValid, long lStart, long lEnd)
- void **removeLastTwoUsed** ()
- void **clearFiles** ()

### Protected Attributes

- int **m\_nY**
  - ReentrantReadWriteLock **m\_oLockY** = new ReentrantReadWriteLock(true)
- 

### Detailed Description

A **TileYCache** is always a part of a **TileXCache** so the files contained by this a **TileYCache** represent files in the tile that has the x index of the **TileXCache** and the y index of the **TileYCache**

---

## Member Function Documentation

`void imrcp.store.SpatialFileCache.TileYCache.clearFiles () [protected]`

Checks for and removes any file in the cache that has not been used for the configured timeout.

`int imrcp.store.SpatialFileCache.TileYCache.compareTo (TileYCache o)`

Compares **TileYCache** by y index

`void imrcp.store.SpatialFileCache.TileYCache.removeLastTwoUsed () [protected]`

Determines and removes the 2 least recently used files from the cache.

`int imrcp.store.SpatialFileCache.TileYCache.search (long lValid, long lStart, long lEnd) [protected]`

Searches for a file with the given time parameters in the list of files.

### Parameters

<code>lValid</code>	time the file starts being valid in milliseconds since Epoch
<code>lStart</code>	start time of the file in milliseconds since Epoch
<code>lEnd</code>	end time of the file in milliseconds since Epoch

### Returns

The result of the `Collections#binarySearch(java.util.List, java.lang.Object, java.util.Comparator)` call. If the result is greater or equal to 0 it is the position in the list of the matching file. If the result is negative, the 2's compliment is insertion point.

---

## Member Data Documentation

`int imrcp.store.SpatialFileCache.TileYCache.m_nY [protected]`

y tile index

`ReentrantReadWriteLock imrcp.store.SpatialFileCache.TileYCache.m_oLockY = new ReentrantReadWriteLock(true) [protected]`

Read/Write lock for this list of files

---

*The documentation for this class was generated from the following file:*

- store/SpatialFileCache.java
- 

---

## imrcp.store.NHCWrapper.TrackParser Class Reference

### Protected Member Functions

- `void parse (InputStream oIn, int nSize) throws Exception`
-

## Detailed Description

Object used to parse the track .kml files from the National Hurricane Center

---

## Member Function Documentation

***void imrcp.store.NHCWrapper.TrackParser.parse (InputStream oIn, int nSize) throws Exception [protected]***

Parses the given InputStream which should be at the start of a track .kml file from the National Hurricane Center.

### Parameters

<i>oIn</i>	cone .kml file
<i>nSize</i>	length of the file in bytes

### Exceptions

<i>Exception</i>
------------------

---

*The documentation for this class was generated from the following file:*

- store/NHCWrapper.java
- 

## imrcp.store.TrafficEventCsv Class Reference

### Public Member Functions

- **TrafficEventCsv (int[] nObsTypes)**
- **void load (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId)**  
throws Exception
- **void deleteFile (File oFile)**

### Additional Inherited Members

---

## Detailed Description

Parses and creates **EventObs** from IMRCP CSV work zone and event files

### Author

Federal Highway Administration

---

## Constructor & Destructor Documentation

***imrcp.store.TrafficEventCsv.TrafficEventCsv (int[] nObsTypes)***

Wrapper for **CsvWrapper#CsvWrapper (int[])**

### Parameters

<i>nObsTypes</i>	IMRCP observation types this file provides.
------------------	---

---

## Member Function Documentation

### `void imrcp.store.TrafficEventCsv.deleteFile (File oFile)`

Attempts to delete the given File. Right now the implementation is commented out so files are not accidentally deleted

#### Parameters

<code>oFile</code>	File to delete
--------------------	----------------

Reimplemented from **imrcp.store.FileWrapper** (*p.224*).

### `void imrcp.store.TrafficEventCsv.load (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId) throws Exception`

IMRCP CSV observation files can be appended to while the file is already in memory. If the file is not in memory `m_oCsvFile` gets initialized and skips the header. Observations are then created and added to `m_oObs` for each line of the file.

Reimplemented from **imrcp.store.CsvWrapper** (*p.125*).

---

*The documentation for this class was generated from the following file:*

- store/TrafficEventCsv.java
- 

## imrcp.store.TrafficEventStore Class Reference

### Public Member Functions

- `ResultSet getData (int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)`
- `void getData (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)`

### Protected Member Functions

- `FileWrapper getNewFileWrapper ()`

### Additional Inherited Members

---

#### Detailed Description

**FileCache** that manages IMRCP CSV work zone and event files

#### Author

Federal Highway Administration

---

## Member Function Documentation

`void imrcp.store.TrafficEventStore.getData (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)`

Determines the files that match the query and then adds the **EventObs** from those files that match the query

Reimplemented from **imrcp.store.CsvStore** (p.123).

`ResultSet imrcp.store.TrafficEventStore.getData (int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)`

### Returns

a new **ImrcpEventResultSet** filled with observations that match the query.

Reimplemented from **imrcp.system.BaseBlock** (p.97).

`FileWrapper imrcp.store.TrafficEventStore.getNewFileWrapper () [protected]`

### Returns

a new **TrafficEventCsv** with the configured observation types

Reimplemented from **imrcp.store.CsvStore** (p.124).

---

*The documentation for this class was generated from the following file:*

- store/TrafficEventStore.java
- 

## imrcp.store.TrafficSpeedStore Class Reference

### Public Member Functions

- `void getData (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)`

### Protected Member Functions

- `FileWrapper getNewFileWrapper ()`

### Additional Inherited Members

---

### Detailed Description

**FileCache** that manages IMRCP's binary traffic speed files that can be loaded using **TrafficSpeedStoreWrapper**s.

### Author

Federal Highway Administration

## Member Function Documentation

`void imrcp.store.TrafficSpeedStore.getData (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)`

Determines the files that match the query and calls `TrafficSpeedStoreWrapper#getData(imrcp.store.ImrcpResultSet, int, long, long, int, int, int, long)` for each one.

Reimplemented from `imrcp.system.BaseBlock` (p.96).

`FileWrapper imrcp.store.TrafficSpeedStore.getNewFileWrapper () [protected]`

### Returns

a new `TrafficSpeedStoreWrapper` with the configured observation types

Reimplemented from `imrcp.store.FileCache` (p.212).

---

*The documentation for this class was generated from the following file:*

- store/TrafficSpeedStore.java
- 

## imrcp.store.TrafficSpeedStoreWrapper Class Reference

### Public Member Functions

- `void deleteFile (File oFile)`
- `void load (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId)`  
throws Exception
- `void getData (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)`
- `void cleanup (boolean bDelete)`

### Additional Inherited Members

---

#### Detailed Description

Parses and creates `Obs` from IMRCP binary traffic speed files.

#### Author

Federal Highway Administration

---

## Member Function Documentation

`void imrcp.store.TrafficSpeedStoreWrapper.cleanup (boolean bDelete)`

Closes the InputStream if it is not null.

#### Parameters

<code>bDelete</code>	not used for this implementation
----------------------	----------------------------------

Reimplemented from **imrcp.store.FileWrapper** (p.223).

**void imrcp.store.TrafficSpeedStoreWrapper.deleteFile (File oFile)**

Attempts to delete the given File. Right now the implementation is commented out so files are not accidentally deleted

**Parameters**

<i>oFile</i>	File to delete
--------------	----------------

Reimplemented from **imrcp.store.FileWrapper** (p.224).

**void imrcp.store.TrafficSpeedStoreWrapper.getData (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)**

Iterates through the observations of the file and adds them to the **ImrcpResultSet** if they match the query.

**Parameters**

<i>oReturn</i>	<b>ImrcpResultSet</b> that will be filled with obs
<i>nType</i>	obstype id
<i>lStartTime</i>	start time of the query in milliseconds since Epoch
<i>lEndTime</i>	end time of the query in milliseconds since Epoch
<i>nStartLat</i>	lower bound of latitude in decimal degrees scaled to 7 decimal places
<i>nEndLat</i>	upper bound of latitude in decimal degrees scaled to 7 decimal places
<i>nStartLon</i>	lower bound of longitude in decimal degrees scaled to 7 decimal places
<i>nEndLon</i>	upper bound of longitude in decimal degrees scaled to 7 decimal places
<i>lRefTime</i>	reference time (observations received after this time will not be included)

**void imrcp.store.TrafficSpeedStoreWrapper.load (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId) throws Exception**

Parses the given IMRCP binary traffic speed file creating speed and traffic observations.

Reimplemented from **imrcp.store.FileWrapper** (p.224).

---

*The documentation for this class was generated from the following file:*

- store/TrafficSpeedStoreWrapper.java
- 

## imrcp.system.Units.UnitConv Class Reference

### Public Member Functions

- double **convert** (double dValue)
- int **compareTo** (**UnitConv** oUnitConv)

## Protected Attributes

- double **m\_dMultiply** = 1.0
  - double **m\_dDivide** = 1.0
  - double **m\_dAdd** = 0.0
  - String **m\_sFromUnit**
  - String **m\_sToUnit**
- 

## Detailed Description

Wraps forward conversions which are conversions of the form:

from-units -> to-units

It extends {

*See also*

**UnitConv**} implementing the {

**UnitConv::convert**} method.

**UnitConv**

**UnitConvR**

---

## Member Function Documentation

***int imrcp.system.Units.UnitConv.compareTo (UnitConv oUnitConv)***

Compares the units by their labels to determine if they're the same.

*Parameters*

<i>oUnitConv</i>	The units to compare to the base units.
------------------	---

*Returns*

0 - if both the convert-to and convert-from labels of the base units match those of oUnitConv.

***double imrcp.system.Units.UnitConv.convert (double dValue)***

The convert method returns the value passed in. It is meant to be the default conversion if no conversions can be found. Extension of UnitConv perform standard, more useful overridden conversion methods based off the conversion factors.

*Parameters*

<i>dValue</i>	The value to be converted.
---------------	----------------------------

*Returns*

The newly converted value.

---

## Member Data Documentation

***double imrcp.system.Units.UnitConv.m\_dAdd = 0.0 [protected]***

Addition factor.

***double imrcp.system.Units.UnitConv.m\_dDivide = 1.0 [protected]***

Division factor.

*double imrcp.system.Units.UnitConv.m\_dMultiply = 1.0 [protected]*

Multiply factor.

*String imrcp.system.Units.UnitConv.m\_sFromUnit [protected]*

Unit label corresponding to the units to be converted from.

*String imrcp.system.Units.UnitConv.m\_sToUnit [protected]*

Unit label corresponding to the units to be converted to.

---

*The documentation for this class was generated from the following file:*

- system/Units.java
- 

## imrcp.system.Units Class Reference

### Classes

- class **UnitConv**

### Public Member Functions

- double **convert** (String sFromUnit, String sToUnit, double dVal)
- String **getSourceUnits** (int nObsType, int nContribId)
- **UnitConv getConversion** (String sFromUnit, String sToUnit)

### Static Public Member Functions

- static **Units getInstance** ()

### Protected Attributes

- ArrayList< **SourceUnit** > **m\_oSourceUnits** = new ArrayList()
- 

### Detailed Description

The Units class keeps track of {

#### See also

**UnitConv**}'s stored in the database and wraps both Forward and Reverse conversions.

Implements the **ILockFactory** interface to allow {

#### See also

**UnitConv**}'s to be modified in a mutually exclusive fashion through the use of **StripeLock** containers.

This is a singleton class who's instance can be retrieved by the {

#### See also

**Units::getInstance**} method.

#### See also

**UnitConvF**

**UnitConvR**

---

## Member Function Documentation

*double imrcp.system.Units.convert (String sFromUnit, String sToUnit, double dVal)*

Attempts to convert the given value from one unit to another.

### Parameters

<i>sFromUnit</i>	The unit to convert from
<i>sToUnit</i>	The unit to convert to
<i>dVal</i>	the value to convert

### Returns

The converted value if conversion was possible. If conversion isn't possible (due to either of the unit Strings being null, the unit Strings being equal to each other, or not having a conversion defined for the given units) the original value is returned.

*UnitConv imrcp.system.Units.getConversion (String sFromUnit, String sToUnit)*

This method searches the list of conversion for the conversion between the supplied units.

### Parameters

<i>sFromUnit</i>	The unit to convert from.
<i>sToUnit</i>	The unit to convert to.

### Returns

Null when either of the supplied units are null, if the supplied units are the same, or if the conversion is not stored in the conversion list. Else it returns the queried conversion.

*static Units imrcp.system.Units.getInstance () [static]*

Retrieves the singleton instance of Units.

### Returns

The instance of Units.

*String imrcp.system.Units.getSourceUnits (int nObsType, int nContribId)*

### Parameters

<i>nObsType</i>	
<i>nContribId</i>	

### Returns

---

*The documentation for this class was generated from the following file:*

- system/Units.java
-

## imrcp.web.USCenPlaceLookup Class Reference

### Public Member Functions

- void **reset** ()
- boolean **start** () throws Exception
- int **doLookup** (HttpServletRequest oReq, HttpServletResponse oRes, Session oSession) throws IOException, ServletException
- int **doGeo** (HttpServletRequest oReq, HttpServletResponse oRes, Session oSession) throws IOException, ServletException

### Public Attributes

- ArrayList< CenGroup > **m\_oGroups** = new ArrayList()
- ArrayList< CenPlace > **m\_oPlacesById** = new ArrayList()
- char[] **m\_cSearch** = new char[3]
- String[] **m\_sAbrevs**

### Static Public Attributes

- static Comparator< CenPlace > **PLACEBYLABEL** = (CenPlace o1, CenPlace o2) -> o1.m\_sLabel.compareTo(o2.m\_sLabel)
- static Comparator< CenPlace > **PLACEBYGEOID** = (CenPlace o1, CenPlace o2) -> o1.m\_nGeoId - o2.m\_nGeoId

### Additional Inherited Members

---

#### Detailed Description

This servlet manages requests for US Census Places (cities, counties, states).

#### Author

Federal Highway Administration

---

#### Member Function Documentation

`int imrcp.web.USCenPlaceLookup.doGeo (HttpServletRequest oReq,  
HttpServletResponse oRes, Session oSession) throws IOException, ServletException`

Adds the geometry of the **CenPlace** described by the "place" request parameter to the response as a GeoJSON Polygon coordinate array.

#### Parameters

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oRes</i>	object that contains the response the servlet sends to the client
<i>oSession</i>	object that contains information about the user that made the request

#### Returns

HTTP status code to be included in the response.

#### Exceptions

<i>IOException</i>	
<i>ServletException</i>	

`int imrcp.web.USCenPlaceLookup.doLookup (HttpServletRequest oReq,  
HttpServletResponse oRes, Session oSession) throws IOException, ServletException`

Adds all of the CenPlaces that start with the same characters as the "lookup" request parameter to the response.

*Parameters*

<code>oReq</code>	object that contains the request the client has made of the servlet
<code>oRes</code>	object that contains the response the servlet sends to the client
<code>oSession</code>	object that contains information about the user that made the request

*Returns*

HTTP status code to be included in the response.

*Exceptions*

<code>IOException</code>	
<code>ServletException</code>	

`void imrcp.web.USCenPlaceLookup.reset ()`

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.web.SecureBaseBlock** (*p.492*).

`boolean imrcp.web.USCenPlaceLookup.start () throws Exception`

Reads all of the US Census shapefiles to have the Place definitions in memory.

*Returns*

true if no Exceptions are thrown

*Exceptions*

<code>Exception</code>	
------------------------	--

Reimplemented from **imrcp.system.BaseBlock** (*p.99*).

---

## Member Data Documentation

`char [] imrcp.web.USCenPlaceLookup.m_cSearch = new char[3]`

Convenience search object used for finding groups

`ArrayList<CenGroup> imrcp.web.USCenPlaceLookup.m_oGroups = new ArrayList()`

Stores all of the Census Groups. Groups contain Places and act as a way of indexing Places for quick lookup

`ArrayList<CenPlace> imrcp.web.USCenPlaceLookup.m_oPlacesByld = new ArrayList()`

Stores all of the Census Places, sorted by id

### *String [] imrcp.web.USCenPlaceLookup.m\_sAbbrevs*

Array containing abbreviations for Census states and territories. The indices represent the state/territory FIPS code. Some positions will be null since all of the numbers are not used as FIPS codes

### *Comparator<CenPlace> imrcp.web.USCenPlaceLookup.PLACEBYGEOID = (CenPlace o1, CenPlace o2) -> o1.m\_nGeoid - o2.m\_nGeoid [static]*

C.compares CenPlaces by geo id

### *Comparator<CenPlace> imrcp.web.USCenPlaceLookup.PLACEBYLABEL = (CenPlace o1, CenPlace o2) -> o1.m\_sLabel.compareTo(o2.m\_sLabel) [static]*

C.compares CenPlaces by label

---

*The documentation for this class was generated from the following file:*

- web/USCenPlaceLookup.java
- 

## imrcp.web.UserProfile Class Reference

### Public Member Functions

- **UserProfile ()**
- **UserProfile (CsvReader oIn)** throws IOException

### Public Attributes

- String[] **m\_sNetworks**
  - ArrayList< String > **m\_oGraphs**
- 

### Detailed Description

Object that contains information about system components the user is allowed to access.

#### Author

Federal Highway Administration

---

### Constructor & Destructor Documentation

#### *imrcp.web.UserProfile.UserProfile ()*

Default constructor. Gives the user access to no networks.

#### *imrcp.web.UserProfile.UserProfile (CsvReader oIn) throws IOException*

Constructs a **UserProfile** from the given user profile CSV file

#### Parameters

<i>oIn</i>	CsvReader wrapping the InputStream of the user profile CSV file
------------	---

#### Exceptions

<i>IOException</i>
--------------------

---

## Member Data Documentation

*ArrayList<String> imrcp.web.UserProfile.m\_oGraphs*

The dashboard graphs the User has permission to access

*String [] imrcp.web.UserProfile.m\_sNetworks*

The Network ids the User has permission to access

---

*The documentation for this class was generated from the following file:*

- web/UserProfile.java
- 

## imrcp.web.UserSettingsServlet Class Reference

### Public Member Functions

- void **reset** ()
- int **doMapSettings** (HttpServletRequest oReq, HttpServletResponse oResp, Session oSession) throws ServletException, IOException
- int **doSaveMapSettings** (HttpServletRequest oReq, HttpServletResponse oResp, Session oSession) throws ServletException, IOException

### Additional Inherited Members

---

#### Detailed Description

Servlet that handles request from the IMRCP Map UI dealing with user's map settings.

#### Author

Federal Highway Administration

---

### Member Function Documentation

*int imrcp.web.UserSettingsServlet.doMapSettings (HttpServletRequest oReq, HttpServletResponse oResp, Session oSession) throws ServletException, IOException*

Adds the user's map settings to the response.

#### Parameters

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oResp</i>	object that contains the response the servlet sends to the client
<i>oSession</i>	object that contains information about the user that made the request

#### Returns

HTTP status code to be included in the response.

#### Exceptions

<i>ServletException</i>	
<i>n</i>	

*int imrcp.web.UserSettingsServlet.doSaveMapSettings (HttpServletRequest oReq, HttpServletResponse oResp, Session oSession) throws ServletException, IOException*

Saves the user's map settings in the request on disk.

#### Parameters

<i>oReq</i>	object that contains the request the client has made of the servlet
<i>oResp</i>	object that contains the response the servlet sends to the client
<i>oSession</i>	object that contains information about the user that made the request

#### Returns

HTTP status code to be included in the response.

#### Exceptions

<i>ServletException</i>	
<i>n</i>	

*void imrcp.web.UserSettingsServlet.reset ()*

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.web.SecureBaseBlock** (p.492).

---

*The documentation for this class was generated from the following file:*

- web/UserSettingsServlet.java
- 

## imrcp.system.Util Class Reference

### Static Public Member Functions

- static String **getLastLineOfFile** (String sFilename) throws Exception
  - static byte[] **longToBytes** (long lLong, byte[] yBytes)
  - static long **bytesToLong** (byte[] yBytes)
  - static GZIPOutputStream **getGZIPOutputStream** (OutputStream oOs) throws IOException
  - static float **toFloat** (int hbits)
  - static float **fromHpf** (int hbits)
  - static int **toHpf** (float fval)
- 

### Detailed Description

Contains utility methods for half-precision floating point numbers, getting byte array from primitive numbers, and reading the last line of a file.

*Author*

Federal Highway Administration

---

**Member Function Documentation**

*static long imrcp.system.Util.bytesToLong (byte[] yBytes) [static]*

Converts the bytes in the array into a long.

*Parameters*

<i>yBytes</i>	the bytes defining a long
---------------	---------------------------

*Returns*

a long defined by the bytes in the array

*static float imrcp.system.Util.fromHpf (int hbits) [static]*

Calculates the half-precision floating point number for the given integer which is treated as a short as the higher 16 bits are ignored.

*Parameters*

<i>hbits</i>	short to convert to half-precision floating point
--------------	---

*Returns*

*static GZIPOutputStream imrcp.system.Util.getGZIPOutputStream (OutputStream oOs) throws IOException [static]*

Convenience function used to wrap the given OutputStream with a GZIPOutputStream using level 9 (best compression, most processing) compression

*Parameters*

<i>oOs</i>	OutputStream to wrap
------------	----------------------

*Returns*

GZIPOutputStream that is wrapping the OutputStream

*Exceptions*

<i>IOException</i>
--------------------

*static String imrcp.system.Util.getLastLineOfFile (String sFilename) throws Exception [static]*

Gets the last line of the text file with the given path.

*Parameters*

<i>sFilename</i>	path to the file to open
------------------	--------------------------

*Returns*

the last line of the file

### Exceptions

<i>Exception</i>	
------------------	--

**static byte[] imrcp.system.Util.longToBytes (long lLong, byte[] yBytes) [static]**

Fills the given byte array with the bytes of the given long.

### Parameters

<i>lLong</i>	The long to convert to bytes
<i>yBytes</i>	byte array to be filled with the bytes of the long.

### Returns

the reference of the byte array passed into the function. It is filled with the bytes that make up the long.

**static float imrcp.system.Util.toFloat (int hbits) [static]**

Looks up the half-precision floating point number associated with the given short.

### Parameters

<i>hbits</i>	half-precision floating point short value
--------------	---

### Returns

half-precision floatign point number associated with the given short

**static int imrcp.system.Util.toHpf (float fval) [static]**

Converts the given float to a short value that acts as a look up index for **SHORT\_FLOAT**

### Parameters

<i>fval</i>	the float to convert
-------------	----------------------

### Returns

The short value associated with the given float as a half-precision floating point number. All higher 16 bits as 0 for all results

---

*The documentation for this class was generated from the following file:*

- system/Util.java
- 

## imrcp.system.Utility Class Reference

### Static Public Member Functions

- static short **swap** (short rValue)
- static int **swap** (int nValue)
- static long **swap** (long lValue)
- static float **swap** (float fValue)
- static double **swap** (double dValue)
- static double **swapD** (long lValue)
- static int **unsignByte** (byte yValue)
- static int **floor** (int nValue, int nPrecision)

- static boolean **isPointInsideRegion** (int nX, int nY, int nTop, int nRight, int nBottom, int nLeft, int nTolerance)
  - static synchronized void **timing** (int nValue)
- 

## Detailed Description

Standard utility functions.

---

## Member Function Documentation

**static int imrcp.system.Utility.floor (int nValue, int nPrecision) [static]**

Determines the next smallest integer value.

### Parameters

<i>nValue</i>	integer value to floor
<i>nPrecision</i>	

### Returns

next smallest integer to the parameter integer

**static boolean imrcp.system.Utility.isPointInsideRegion (int nX, int nY, int nTop, int nRight, int nBottom, int nLeft, int nTolerance) [static]**

Determines if the specified point is within the specified rectangular region.

### Parameters

<i>nX</i>	x coordinate of point
<i>nY</i>	y coordinate of point
<i>nTop</i>	y value of the top of the region
<i>nRight</i>	x value of the right side of the region
<i>nBottom</i>	y value of the bottom of the region
<i>nLeft</i>	x value of the left side of the region
<i>nTolerance</i>	coordinate thickness of the rectangular region

### Returns

true if the points are within or on the rectangular region, false otherwise

**static double imrcp.system.Utility.swap (double dValue) [static]**

Reverses the eight bytes in a double precision number. The first byte becomes the eighth byte, the second byte becomes the seventh byte, etc...

### Parameters

<i>dValue</i>	double precision value to have its bytes swapped
---------------	--

### Returns

long integer with bytes swapped

***static float imrcp.system.Utility.swap (float fValue) [static]***

Reverses the four bytes in a floating point number. The first byte becomes the fourth byte, the second byte becomes the third byte, the third byte becomes the second byte and the fourth byte becomes the first byte.

*Parameters*

<i>fValue</i>	floating point value to have its bytes swapped
---------------	--

*Returns*

long integer with bytes swapped

***static int imrcp.system.Utility.swap (int nValue) [static]***

Reverses the four bytes in an integer. The first byte becomes the fourth byte, the second byte becomes the third byte, the third byte becomes the second byte and the fourth byte becomes the first byte.

Example: swap(134217728) returns 16909320

- <nobr>134217728 => 00001000 00000100 00000010 00000001</nobr>
- <nobr>16777216 => 00000001 00000010 00000100 00001000</nobr>

*Parameters*

<i>nValue</i>	integer value to have its bytes swapped
---------------	---

*Returns*

integer with bytes swapped

***static long imrcp.system.Utility.swap (long lValue) [static]***

Reverses the eight bytes in a long integer. The first byte becomes the eighth byte, the second byte becomes the seventh byte, etc...

Example: swap(72057594037927936) returns 1

- <nobr>134217728 => 00000001 00000000 00000000 00000000 00000000 00000000 00000000 00000000</nobr>
- <nobr>16777216 => 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000001</nobr>

*Parameters*

<i>lValue</i>	long integer value to have its bytes swapped
---------------	--

*Returns*

long integer with bytes swapped

***static short imrcp.system.Utility.swap (short rValue) [static]***

Reverses the two bytes in a short integer. The first byte of the short becomes the second byte and the second byte of the short becomes the first.

Example: swap(1) returns 256

- <nobr>1 => 00000000 00000001</nobr>
- <nobr>256 => 00000001 00000000</nobr>

and swap(257) returns 257

- <nobr>257 => 00000001 00000001</nobr>
- <nobr>257 => 00000001 00000001</nobr>

**Parameters**

<i>rValue</i>	short integer value to have its bytes swapped
---------------	---

**Returns**

short integer with bytes swapped

**static double imrcp.system.Utility.swapD (long lValue) [static]**

**Parameters**

<i>lValue</i>
---------------

**Returns**

**static synchronized void imrcp.system.Utility.timing (int nValue) [static]**

Provides information on the currently running thread. Prints the current system time in milliseconds, the parameter value, and the currently running thread identifier.

**Parameters**

<i>nValue</i>	identifying value
---------------	-------------------

**static int imrcp.system.Utility.unsignByte (byte yValue) [static]**

Returns the unsigned integer value of a byte.

**Parameters**

<i>yValue</i>	the byte
---------------	----------

**Returns**

the unsigned integer value of the byte

---

*The documentation for this class was generated from the following file:*

- system/Utility.java
- 

---

**imrcp.geosrv.osm.WayIterator Class Reference**

**Public Member Functions**

- boolean **hasNext ()**
  - int[] **next ()**
  - void **remove ()**
-

## Detailed Description

Object used to iterate through the nodes of a roadway segment, where each iteration provides a line segment define by the current point and the next point.

## Author

Federal Highway Administration

---

## Member Function Documentation

`int[] imrcp.geosrv.osm.WayIterator.next ()`

Copies the points of the next line segment into **m\_nLine** and returns it reference.

`void imrcp.geosrv.osm.WayIterator.remove ()`

NOT IMPLEMENTED

---

*The documentation for this class was generated from the following file:*

- geosrv/osm/WayIterator.java
- 

## imrcp.geosrv.WayNetworks.WayMetadata Class Reference

### Public Member Functions

- int **compareTo** (WayMetadata o)

### Public Attributes

- **Id m\_oId**
  - int **m\_nSpdLimit** = -1
  - int **m\_nLanes** = -1
  - int **m\_nHOV** = 0
  - int **m\_nPavementCondition** = 1
  - double **m\_dMinMslElev** = Double.NaN
  - double **m\_dMaxMslElev** = Double.NaN
  - double **m\_dMinGroundElev** = Double.NaN
  - double **m\_dMaxGroundElev** = Double.NaN
- 

## Detailed Description

Encapsulates multiple metadata parameters associated with roadway segments

---

## Member Function Documentation

`int imrcp.geosrv.WayNetworks.WayMetadata.compareTo (WayMetadata o)`

Compares WayMetadata by Id

---

## Member Data Documentation

`double imrcp.geosrv.WayNetworks.WayMetadata.m_dMaxGroundElev = Double.NaN`

Maximum Ground Elevation of the roadway segment

`double imrcp.geosrv.WayNetworks.WayMetadata.m_dMaxMslElev = Double.NaN`

Maximum Mean Sea Level Elevation of the roadway segment

`double imrcp.geosrv.WayNetworks.WayMetadata.m_dMinGroundElev = Double.NaN`

Minimum Ground Elevation of the roadway segment

`double imrcp.geosrv.WayNetworks.WayMetadata.m_dMinMslElev = Double.NaN`

Minimum Mean Sea Level Elevation of the roadway segment

`int imrcp.geosrv.WayNetworks.WayMetadata.m_nHOV = 0`

HOV flag. 0 = no HOV lane 1 = has an HOV lane

`int imrcp.geosrv.WayNetworks.WayMetadata.m_nLanes = -1`

Number of lanes of the roadway segment

`int imrcp.geosrv.WayNetworks.WayMetadata.m_nPavementCondition = 1`

Pavement condition enumeration. 1 = good condition 2 = average condition 3 = poor condition

`int imrcp.geosrv.WayNetworks.WayMetadata.m_nSpdLimit = -1`

Speed limit in mph of the roadway segment

`Id imrcp.geosrv.WayNetworks.WayMetadata.m_old`

Id of the associated OsmWay

---

*The documentation for this class was generated from the following file:*

- geosrv/WayNetworks.java
- 

## imrcp.geosrv.WayNetworks Class Reference

### Classes

- class **WayMetadata**

### Public Member Functions

- void **reset** ()
- boolean **start** () throws Exception
- **OsmWay getWay** (int nTol, int nLon, int nLat, ArrayList<**OsmWay**> oWays)
- **OsmWay getWay** (int nTol, int nLon, int nLat)
- int **getWays** (ArrayList<**OsmWay**> oWays, int nTol, int nLon1, int nLat1, int nLon2, int nLat2)
- synchronized void **getTiles** (ArrayList<**TileBucket**> oTileBuckets)
- ArrayList<**WaySnapInfo**> **getSnappedWays** (ArrayList<**OsmWay**> oWays, int nTol, int nLon, int nLat, double dHdg, double dHdgTol, Comparator<**WaySnapInfo**> oComp)
- ArrayList<**WaySnapInfo**> **getSnappedWays** (int nTol, int nLon, int nLat, double dHdg, double dHdgTol, Comparator<**WaySnapInfo**> oComp)
- **OsmWay getWayById** (**Id** oId)

- **Network getNetwork** (String sId)
- boolean **deleteNetwork** (String sId) throws IOException
- boolean **reprocessNetwork** (String sId, String sLabel) throws IOException
- void **finalizeNetwork** (String sId, String sStateShps, String sOsmDir)
- void **createNetwork** (String sId, String sLabel, String[] sOptions, String sCoords)
- void **writeNetworkFile** () throws IOException
- int **getSpdLimit** (**Id** oId)
- int **getLanes** (**Id** oId)
- double **getMsxElev** (**Id** oId)
- ArrayList< Network > **getNetworks** ()
- ArrayList< Network > **getAllNetworks** ()
- String **getTimeZone** (String sNetworkId)
- WayMetadata **getMetadata** (**Id** oId)

## Additional Inherited Members

---

### Detailed Description

This class contains methods for managing and retrieving data for roadway networks including geometric definitions, number of lanes, and speed limits. It maintains the **Network** JSON file which describes the different networks available in the system.

#### Author

Federal Highway Administration

---

### Member Function Documentation

**void imrcp.geosrv.WayNetworks.createNetwork (String sId, String sLabel, String[] sOptions, String sCoords)**

Creates and adds a new **Network** with the given parameters to the in memory list and rewrites the **Network** JSON file to save the changes.

#### Parameters

<i>sId</i>	Base64 encoded 16 byte NetworkId
<i>sLabel</i>	Label of the <b>Network</b>
<i>sOptions</i>	Array of options used to create the <b>Network</b>
<i>sCoords</i>	comma separated String of the coordinates of the polygon used to create the <b>Network</b> .

**boolean imrcp.geosrv.WayNetworks.deleteNetwork (String sId) throws IOException**

Deletes the **Network** with the given Id by removing it from the in memory list, rewriting the **Network** JSON without it, and deleting the directory and files associated with the **Network**.

#### Parameters

<i>sId</i>	Base64 encoded 16 byte NetworkId of the <b>Network</b> to delete.
------------	---

#### Returns

true if the **Network** is found and no Exceptions are thrown in the process of deleting it

### Exceptions

<i>IOException</i>
--------------------

*void imrcp.geosrv.WayNetworks.finalizeNetwork (String sId, String sStateShps, String sOsmDir)*

Calls **Scheduling#execute(java.lang.Runnable)** to schedule the **Network** with the given Id to be finalized which calls **OsmUtil#finalizeNetwork(java.lang.String, java.lang.String, java.lang.String[], java.lang.String[], java.lang.String[], java.lang.String)** and then rewrites the **Network** JSON file to save the changes.

### Parameters

<i>sId</i>	Base64 encoded 16 byte <b>Network</b> Id of the <b>Network</b> to finalize
<i>sStateShps</i>	Path to the shapefile that contains the geometric definitions of the states in the USA.
<i>sOsmDir</i>	Path to the directory that contains the Open Street Map files

*ArrayList< Network > imrcp.geosrv.WayNetworks.getAllNetworks ()*

Creates a new list and adds all of the Networks to it.

### Returns

The created list that is filled all of the Networks

*int imrcp.geosrv.WayNetworks.getLanes (Id oid)*

Gets the number of lanes of the roadway segment with the given Id.

### Parameters

<i>oId</i>	Id of the OsmWay to get the number of lanes for
------------	---

### Returns

The number of lanes stored in the metadata list for the given Id. If the Id is not found in the list, -1.

*WayMetadata imrcp.geosrv.WayNetworks.getMetadata (Id oid)*

Gets the **WayMetadata** object of the roadway segment with the given Id.

### Parameters

<i>oId</i>	Id of the OsmWay to get <b>WayMetadata</b> object for
------------	---

### Returns

The **WayMetadata** for the given Id. If the Id is not found in the list, **DEFAULTMETADATA** is returned

*double imrcp.geosrv.WayNetworks.getMslElev (Id oid)*

Gets the average mean sea level elevation of the roadway segment with the given Id.

### Parameters

<i>oId</i>	Id of the OsmWay to get the mea sea level elevation for
------------	---

#### Returns

The average of the mean sea level elevations stored in the metadata list for the given Id. If the Id is not found in the list, Double.NaN

### *Network imrcp.geosrv.WayNetworks.getNetwork (String sId)*

Gets the **Network** with the given NetworkId

#### Parameters

sId	Base64 encoded 16 byte NetworkId
-----	----------------------------------

#### Returns

The **Network** with the given NetworkId, if there is not a **Network** with the given Id, null is returned.

### *ArrayList< Network > imrcp.geosrv.WayNetworks.getNetworks ()*

Creates a new list and adds the finalized Networks to it.

#### Returns

The created list that is filled with finalized Networks

### *ArrayList< WaySnapInfo > imrcp.geosrv.WayNetworks.getSnappedWays (ArrayList< OsmWay > oWays, int nTol, int nLon, int nLat, double dHdg, double dHdgTol, Comparator< WaySnapInfo > oComp)*

Creates a new list of WaySnapInfos and returns it. The list gets filled with objects that encapsulate the roadway segments that are close to the given point and geometric parameters calculated during the snap algorithm to make better decisions on which roadway segment the point should be snapped to if there are multiple candidates.

#### Parameters

<i>oWays</i>	list of roadway segments to attempt to snap the point to
<i>nTol</i>	tolerance to add to/subtract from longitudes
<i>nLon</i>	longitude of point in decimal degrees scaled to 7 decimal places
<i>nLat</i>	latitude of point in decimal degrees scaled to 7 decimal places
<i>dHdg</i>	the angle associated with the given point. Usually the direction to the next point in a line string representing a roadway segment. Use Double.NaN if the point is not directed
<i>dHdgTol</i>	the angle used as a tolerance to determine if the roadway segment the point gets snapped to is going in the same direction.
<i>oComp</i>	Comparator used to sort the <b>WaySnapInfo</b> objects created from the snapping algorithm

#### Returns

A list of WaySnapInfos of candidate roadway segments the point could be snapped in sorted order defined by the Comparator.

### *ArrayList< WaySnapInfo > imrcp.geosrv.WayNetworks.getSnappedWays (int nTol, int nLon, int nLat, double dHdg, double dHdgTol, Comparator< WaySnapInfo > oComp)*

Wrapper for calling `getWays(java.util.ArrayList, int, int, int, int, int)` and then `getSnappedWays(java.util.ArrayList, int, int, int, double, double, java.util.Comparator)`

#### Parameters

<i>nTol</i>	tolerance to add to/subtract from longitudes
<i>nLon</i>	longitude of point in decimal degrees scaled to 7 decimal places
<i>nLat</i>	latitude of point in decimal degrees scaled to 7 decimal places
<i>dHdg</i>	the angle associated with the given point. Usually the direction to the next point in a line string representing a roadway segment. Use Double.NaN if the point is not directed
<i>dHdgTol</i>	the angle used as a tolerance to determine if the roadway segment the point gets snapped to is going in the same direction.
<i>oComp</i>	Comparator used to sort the <b>WaySnapInfo</b> objects created from the snapping algorithm

#### Returns

A list of WaySnapInfos of candidate roadway segments the point could be snapped in sorted order defined by the Comparator.

*int imrcp.geosrv.WayNetworks.getSpdLimit (Id old)*

Gets the speed limit of the roadway segment with the given Id.

#### Parameters

<i>oId</i>	Id of the OsmWay to get the speed limit for
------------	---

#### Returns

The speed limit value stored in the metadata list for the given Id. If the Id is not found in the list, -1.

*synchronized void imrcp.geosrv.WayNetworks.getTiles (ArrayList< TileBucket > oTileBuckets)*

Fills the given list with all of the TileBuckets in **m\_oBucketCache**

#### Parameters

<i>oTileBuckets</i>	List to be filled with TileBuckets
---------------------	------------------------------------

*String imrcp.geosrv.WayNetworks.getTimeZone (String sNetworkId)*

Gets the TimeZone String associated with the given NetworkId. If there is not a configured mapping for the given Id, "UTC" is returned.

#### Parameters

<i>sNetworkId</i>	Base64 encoded 16 byte Network Id
-------------------	-----------------------------------

#### Returns

The associated TimeZone string of the given NetworkId, or "UTC" if there is not an associated TimeZone.

*OsmWay imrcp.geosrv.WayNetworks.getWay (int nTol, int nLon, int nLat)*

Determines that roadway segments that are close to the given point and calls **getWay(int, int, int, java.util.ArrayList)**.

#### Parameters

<i>nTol</i>	tolerance to add to/subtract from longitudes
<i>nLon</i>	longitude of point in decimal degrees scaled to 7 decimal places
<i>nLat</i>	latitude of point in decimal degrees scaled to 7 decimal places

#### Returns

The closest roadway segment to the point. If the point fails to snap to any roadway segment null is returned

*OsmWay imrcp.geosrv.WayNetworks.getWay (int nTol, int nLon, int nLat, ArrayList<OsmWay> oWays)*

Attempts to get roadway segment from the list that is the closest to the given point that is within the tolerance using a snap point to line algorithm.

#### Parameters

<i>nTol</i>	tolerance to add to/subtract from longitudes
<i>nLon</i>	longitude of point in decimal degrees scaled to 7 decimal places
<i>nLat</i>	latitude of point in decimal degrees scaled to 7 decimal places
<i>oWays</i>	list of roadway segments to try and snap the point to

#### Returns

The closest roadway segment to the point. If the point fails to snap to any roadway segment null is returned

*OsmWay imrcp.geosrv.WayNetworks.getWayById (Id old)*

Gets the roadway segment with the given Id.

#### Parameters

<i>old</i>	Id of the OsmWay to search for
------------	--------------------------------

#### Returns

The OsmWay with the given Id, if there is not an OsmWay with the given Id, null is returned.

*int imrcp.geosrv.WayNetworks.getWays (ArrayList<OsmWay> oWays, int nTol, int nLon1, int nLat1, int nLon2, int nLat2)*

Fills the given list with roadway segments are in the hash grids that intersect the bounding box defined by the two longitude and latitude points and the tolerance.

#### Parameters

<i>oWays</i>	list to fill with roadway segments
<i>nTol</i>	tolerance to add to/subtract from longitudes
<i>nLon1</i>	longitude of point 1 in decimal degrees scaled to 7 decimal places
<i>nLat1</i>	latitude of point 1 in decimal degrees scaled to 7 decimal places
<i>nLon2</i>	longitude of point 2 in decimal degrees scaled to 7 decimal places
<i>nLat2</i>	latitude of point 2 in decimal degrees scaled to 7 decimal places

#### Returns

tolerance adjusted for latitude

`boolean imrcp.geosrv.WayNetworks.reprocessNetwork (String sId, String sLabel)`  
`throws IOException`

Resets the **Network** with the given Id to have its roadway segments be reprocessed.

*Parameters*

<code>sId</code>	Base64 encoded 16 byte NetworkId of the <b>Network</b> to reprocess.
<code>sLabel</code>	Label to assign to the <b>Network</b>

*Returns*

true if the **Network** is found and no Exceptions are thrown in resetting it

*Exceptions*

<code>IOException</code>
--------------------------

`void imrcp.geosrv.WayNetworks.reset ()`

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.system.BaseBlock** (p.98).

`boolean imrcp.geosrv.WayNetworks.start () throws Exception`

Parses the **Network** JSON file to load the defined Networks into memory to be available to the system.

*Returns*

true if no Exceptions are thrown

*Exceptions*

<code>Exception</code>
------------------------

Reimplemented from **imrcp.system.BaseBlock** (p.99).

`void imrcp.geosrv.WayNetworks.writeNetworkFile () throws IOException`

Creates and writes a GeoJson representation of each network as the **Network** JSON file. It contains of a JSONArray that contains GeoJson Feature objects.

*Exceptions*

<code>IOException</code>
--------------------------

---

*The documentation for this class was generated from the following file:*

- geosrv/WayNetworks.java
- 

## imrcp.geosrv.WaySnapInfo Class Reference

### Public Member Functions

- **WaySnapInfo ()**
- **WaySnapInfo (OsmWay oWay)**
- **void setValues (WaySnapInfo oInfo)**

- void **reset** ()
- int **compareTo** (**WaySnapInfo** o)

#### Public Attributes

- int **m\_nLonIntersect**
- int **m\_nLatIntersect**
- int **m\_nSqDist**
- int **m\_nRightHandRule**
- double **m\_dProjSide**
- **OsmWay** **m\_oWay** = null
- int **m\_nIndex**

---

#### Detailed Description

Encapsulate information used when snapping a point to a roadway segment (**OsmWay** )

#### Author

Federal Highway Administration

---

#### Constructor & Destructor Documentation

*imrcp.geosrv.WaySnapInfo.WaySnapInfo ()*

Default constructor. Initializes values to the defaults

*imrcp.geosrv.WaySnapInfo.WaySnapInfo (OsmWay oWay)*

Calls **WaySnapInfo ()** and set **m\_oWay** to the given roadway segment

#### Parameters

<i>oWay</i>	roadway segment a point is being snapped to
-------------	---

---

#### Member Function Documentation

*int imrcp.geosrv.WaySnapInfo.compareTo (WaySnapInfo o)*

Compares **WaySnapInfo** by squared distance then by right hand rule

*void imrcp.geosrv.WaySnapInfo.reset ()*

Resets the values to the defaults.

*void imrcp.geosrv.WaySnapInfo.setValues (WaySnapInfo oInfo)*

Sets the values of this to the values of the given **WaySnapInfo**

#### Parameters

<i>oInfo</i>	object to copy values from
--------------	----------------------------

## Member Data Documentation

### *double imrcp.geosrv.WaySnapInfo.m\_dProjSide*

0 if the point snaps (projects) onto the line segment. Positive if the point snaps to the line segment past its terminal point, negative if the point snaps to the line segment before its initial point.

### *int imrcp.geosrv.WaySnapInfo.m\_nIndex*

The node index of the line segment the point snapped to

### *int imrcp.geosrv.WaySnapInfo.m\_nLatIntersect*

Latitude of the snap point in decimal degrees scaled to 7 decimal places

### *int imrcp.geosrv.WaySnapInfo.m\_nLonIntersect*

Longitude of the snap point in decimal degrees scaled to 7 decimal places

### *int imrcp.geosrv.WaySnapInfo.m\_nRightHandRule*

Right hand rule value

### *int imrcp.geosrv.WaySnapInfo.m\_nSqDist*

Squared perpendicular distance from the original point to snap point

### *OsmWay imrcp.geosrv.WaySnapInfo.m\_oWay = null*

Reference to the roadway segment the point snapped to

---

*The documentation for this class was generated from the following file:*

- geosrv/WaySnapInfo.java
- 

## imrcp.store.WeatherStore Class Reference

### Public Member Functions

- void **getData** (**ImrcpResultSet** oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)
- void **reset** ()

### Protected Member Functions

- **GriddedFileWrapper** **getNewFileWrapper** ()

### Protected Attributes

- int[] **m\_nObsTypes**
- String[] **m\_sObsTypes**
- String **m\_sHz**
- String **m\_sVrt**
- String **m\_sTime**
- int **m\_nFilesPerPeriod**

### Additional Inherited Members

---

## Detailed Description

**FileCache** that manages weather files that can be loaded by **NcfWrapper**s, which comes from different National Weather Service products like RTMA, RAP, NDFD, and GFS.

### Author

Federal Highway Administration

---

## Member Function Documentation

**void imrcp.store.WeatherStore.getData (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)**

Determines the files that match the query and then calls **NcfWrapper#getData (int, long, int, int, int, int)** on each of those files

Reimplemented from **imrcp.system.BaseBlock** (p.96).

**GriddedFileWrapper imrcp.store.WeatherStore.getNewFileWrapper () [protected]**

### Returns

a new **NcfWrapper** with the configured parameters.

Reimplemented from **imrcp.store.FileCache** (p.212).

Reimplemented in **imrcp.store.NDFDQpfStore** (p.377), and **imrcp.store.RAPStore** (p.470).

**void imrcp.store.WeatherStore.reset ()**

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.store.FileCache** (p.214).

---

## Member Data Documentation

**int imrcp.store.WeatherStore.m\_nFilesPerPeriod [protected]**

Number of files that are expected to be downloaded each collection cycle

**int [] imrcp.store.WeatherStore.m\_nObsTypes [protected]**

IMRCP observation types the file provides

**String imrcp.store.WeatherStore.m\_sHz [protected]**

Label of the horizontal axis in the file

**String [] imrcp.store.WeatherStore.m\_sObsTypes [protected]**

Label of the corresponding observation types found in the file

**String imrcp.store.WeatherStore.m\_sTime [protected]**

Label of the time axis in the file

**String imrcp.store.WeatherStore.m\_sVrt [protected]**

Label of the vertical axis in the file

*The documentation for this class was generated from the following file:*

- store/WeatherStore.java
- 

## [imrcp.forecast.mlp.Work Class Reference](#)

### Public Member Functions

- **Work ()**

### Public Attributes

- long **m\_lTimestamp**
  - int **m\_nThread**
  - **ByteArrayOutputStream m\_oBoas**
  - **OutputStreamWriter m\_oCompressor**
- 

### Detailed Description

Container used to aid with multi-threaded processing of the MLP model.

#### Author

Federal Highway Administration

---

### Constructor & Destructor Documentation

#### [\*imrcp.forecast.mlp.Work.Work \(\)\*](#)

Default constructor. Does nothing.

---

### Member Data Documentation

#### *long imrcp.forecast.mlp.Work.m\_lTimestamp*

Run time of the MLP model

#### *int imrcp.forecast.mlp.Work.m\_nThread*

Thread number

#### *ByteArrayOutputStream imrcp.forecast.mlp.Work.m\_oBoas*

Stores the bytes for the input files

#### *OutputStreamWriter imrcp.forecast.mlp.Work.m\_oCompressor*

Writer that wraps **m\_oBoas**, intended to be used with a GZIP stream

---

*The documentation for this class was generated from the following file:*

- forecast/mlp/Work.java
-

## [imrcp.forecast.mlp.MLPBlock.WorkDelegate Class Reference](#)

### Public Member Functions

- `void run ()`
- 

### Detailed Description

Delegate object that helps with multi-threading processing

---

### Member Function Documentation

#### `void imrcp.forecast.mlp.MLPBlock.WorkDelegate.run ()`

Removes and processes a `Work`. If this is the last thread to finish executing, calls `finishWork(long)`

---

*The documentation for this class was generated from the following file:*

- `forecast/mlp/MLPBlock.java`
- 

## [imrcp.forecast.mlp.WorkObject Class Reference](#)

### Public Member Functions

- `WorkObject ()`
- `WorkObject (OsmWay oSegment, ArrayList< OsmWay > oDownstream, MLPMetadata oMetadata)`
- `int compareTo (WorkObject o)`

### Public Attributes

- `OsmWay m_oWay`
  - `ArrayList< OsmWay > m_oDownstream`
  - `MLPMetadata m_oMetadata`
- 

### Detailed Description

Contains the information needed to create records for the input files used by the online prediction and long time series update models of MLP.

### Author

Federal Highway Administration

---

### Constructor & Destructor Documentation

#### `imrcp.forecast.mlp.WorkObject.WorkObject ()`

Default constructor. Does nothing.

---

*imrcp.forecast.mlp.WorkObject.WorkObject (OsmWay oSegment, ArrayList<OsmWay> oDownstream, MLPMetadata oMetadata)*

Constructs a **WorkObject** with the given parameters.

**Parameters**

<i>oSegment</i>	segment to make predictions for
<i>oDownstream</i>	list of segments downstream oSegment
<i>oMetadata</i>	metadata associated with oSegment

---

## Member Function Documentation

*int imrcp.forecast.mlp.WorkObject.compareTo (WorkObject o)*

C.compares WorkObjects by their OsmWay Id.

**See also**

`java.lang.Comparable::compareTo(java.lang.Object)`

---

## Member Data Documentation

*ArrayList<OsmWay> imrcp.forecast.mlp.WorkObject.m\_oDownstream*

List of roadway segments that are downstream of **m\_oWay**

*MLPMetadata imrcp.forecast.mlp.WorkObject.m\_oMetadata*

Contains metadata for the **m\_oWay** used in the MLP model

*OsmWay imrcp.forecast.mlp.WorkObject.m\_oWay*

Roadway segment used to make speed predictions

---

*The documentation for this class was generated from the following file:*

- forecast/mlp/WorkObject.java
- 

## imrcp.store.WxDECsv Class Reference

### Public Member Functions

- **WxDECsv** (int[] nObsTypes)
- void **deleteFile** (File oFile)
- void **load** (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId) throws Exception

### Additional Inherited Members

---

#### Detailed Description

Parses and creates **Obs** from WxDE CSV subscription files.

*Author*

Federal Highway Administration

---

**Constructor & Destructor Documentation**

*imrcp.store.WxDECsv.WxDECsv (int[] nObsTypes)*

Wrapper for **CsvWrapper#CsvWrapper (int [])**

*Parameters*

<i>nObsTypes</i>	observation types the file provides
------------------	-------------------------------------

---

**Member Function Documentation**

*void imrcp.store.WxDECsv.deleteFile (File oFile)*

Attempts to delete the given File. Right now the implementation is commented out so files are not accidentally deleted

*Parameters*

<i>oFile</i>	File to delete
--------------	----------------

Reimplemented from **imrcp.store.FileWrapper** (p.224).

*void imrcp.store.WxDECsv.load (long lStartTime, long lEndTime, long lValidTime, String sFilename, int nContribId) throws Exception*

WxD E subscription files are concatenated together for a configured amount of time. The file can be appended to while the file is already in memory. If the file is not in memory **m\_oCsvFile** gets initialized and skips the header. Observations are then created and added to **m\_oObs** for each line of the file.

Reimplemented from **imrcp.store.CsvWrapper** (p.125).

---

*The documentation for this class was generated from the following file:*

- store/WxDECsv.java
- 

---

**imrcp.collect.WxDESub Class Reference**

**Public Member Functions**

- void **reset ()**
- boolean **start ()**
- void **execute ()**

**Additional Inherited Members**

---

**Detailed Description**

**Collector** for Weather Data Environment subscription files

*Author*

Federal Highway Administration

---

**Member Function Documentation**

***void imrcp.collect.WxDESSub.execute ()***

Downloads the current subscription file from WXDE and appends it to the rolling observation file.

Reimplemented from **imrcp.system.BaseBlock** (*p.95*).

***void imrcp.collect.WxDESSub.reset ()***

Sets configurable member variables from the BlockConfig object, should be overridden by child classes.

Reimplemented from **imrcp.collect.Collector** (*p.114*).

***boolean imrcp.collect.WxDESSub.start ()***

Calls **WxDESSub#execute ()** then sets a schedule to execute on a fixed interval.

*Returns*

true if no exceptions are thrown

Reimplemented from **imrcp.system.BaseBlock** (*p.99*).

---

*The documentation for this class was generated from the following file:*

- collect/WxDESSub.java

---

## imrcp.store.WxDESubStore Class Reference

### Public Member Functions

- void **getData** (**ImrcpResultSet** oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)
- void **getDataFromFile** (**ImrcpResultSet** oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime, **CsvWrapper** oFile)

### Protected Member Functions

- **FileWrapper** **getNewFileWrapper** ()

### Additional Inherited Members

---

#### Detailed Description

**FileCache** that manages CSV subscription files downloaded from the Weather Data Environment (WxDE)

#### Author

Federal Highway Administration

---

#### Member Function Documentation

**void imrcp.store.WxDESubStore.getData (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime)**

Determines the files that match the query and then calls **getDataFromFile(imrcp.store.ImrcpResultSet, int, long, long, int, int, int, long, imrcp.store.CsvWrapper)** on each of those files

Reimplemented from **imrcp.store.CsvStore** (p.123).

**void imrcp.store.WxDESubStore.getDataFromFile (ImrcpResultSet oReturn, int nType, long lStartTime, long lEndTime, int nStartLat, int nEndLat, int nStartLon, int nEndLon, long lRefTime, CsvWrapper oFile)**

Iterates through the observations of the file and adds them to the **ImrcpResultSet** if they match the query. If there are multiple observations for the same sensor then the most recent observation is the only one added to the **ImrcpResultSet**.

Reimplemented from **imrcp.store.CsvStore** (p.124).

**FileWrapper imrcp.store.WxDESubStore.getNewFileWrapper () [protected]**

#### Returns

a new **WxDECsv** with the configured observation types.

Reimplemented from **imrcp.store.CsvStore** (p.124).

---

*The documentation for this class was generated from the following file:*

- store/WxDESubStore.java
-

U.S. Department of Transportation  
Federal Highway Administration  
Office of Operations  
1200 New Jersey Avenue, SE  
Washington, DC 20590

Office of Operations website  
<https://ops.fhwa.dot.gov>