IBM Systems - iSeries
Lightweight Directory Access Protocol (LDAP) APIs

Version 5 Release 4
Note

Before using this information and the product it supports, be sure to read the information in "Notices," on page 303.
<table>
<thead>
<tr>
<th>LDAP OPT_ERROR_NUMBER</th>
<th>LDAP OPT_ERROR_STRING</th>
<th>LDAP OPT_EXT_ERROR</th>
<th>LDAP OPT_EXT_GSS_ERR</th>
<th>Return Value</th>
<th>Error Conditions</th>
<th>Error Messages</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>101</td>
<td>101</td>
<td>102</td>
<td>102</td>
<td>102</td>
<td>102</td>
<td>102</td>
</tr>
</tbody>
</table>

ldap_get_values()—Retrieve a Set of Attribute Values from an Entry

<table>
<thead>
<tr>
<th>Authorities and Locks</th>
<th>Parameters</th>
<th>Return Value</th>
<th>Error Conditions</th>
<th>Error Messages</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>103</td>
<td>103</td>
<td>103</td>
<td>103</td>
<td>103</td>
<td>103</td>
</tr>
</tbody>
</table>

ldap_get_values_len()—Retrieve a Set of Binary Attribute Values

<table>
<thead>
<tr>
<th>Authorities and Locks</th>
<th>Parameters</th>
<th>Return Value</th>
<th>Error Conditions</th>
<th>Error Messages</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>104</td>
<td>104</td>
<td>104</td>
<td>104</td>
<td>104</td>
<td>104</td>
</tr>
</tbody>
</table>

ldap_init()—Perform an LDAP Initialization Operation

<table>
<thead>
<tr>
<th>Authorities and Locks</th>
<th>Parameters</th>
<th>Return Value</th>
<th>Error Conditions</th>
<th>Error Messages</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>105</td>
<td>106</td>
<td>107</td>
<td>107</td>
<td>107</td>
<td>107</td>
</tr>
</tbody>
</table>

ldap_insert_control()—Insert a Control in the List of LDAP Server Controls

<table>
<thead>
<tr>
<th>Authorities and Locks</th>
<th>Parameters</th>
<th>Return Value</th>
<th>Error Conditions</th>
<th>Error Messages</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>108</td>
<td>108</td>
<td>108</td>
<td>108</td>
<td>108</td>
<td>108</td>
</tr>
</tbody>
</table>

ldap_is_ldap_url()—Verify LDAP URL

<table>
<thead>
<tr>
<th>Authorities and Locks</th>
<th>Parameters</th>
<th>Return Value</th>
<th>Error Conditions</th>
<th>Error Messages</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>109</td>
<td>109</td>
<td>109</td>
<td>109</td>
<td>109</td>
<td>109</td>
</tr>
</tbody>
</table>

ldap_is_ldap_url_utf8()—Verify UTF-8 codepage LDAP URL

<table>
<thead>
<tr>
<th>Authorities and Locks</th>
<th>Parameters</th>
<th>Return Value</th>
<th>Error Conditions</th>
<th>Error Messages</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>109</td>
<td>110</td>
<td>110</td>
<td>110</td>
<td>110</td>
<td>110</td>
</tr>
</tbody>
</table>

ldap_memfree()—Free Memory Allocated by LDAP API

<table>
<thead>
<tr>
<th>Authorities and Locks</th>
<th>Parameters</th>
<th>Return Value</th>
<th>Error Conditions</th>
<th>Error Messages</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td>110</td>
<td>111</td>
<td>111</td>
<td>111</td>
<td>111</td>
</tr>
</tbody>
</table>

ldap_modify()—Perform an LDAP Modify Entry Request

<table>
<thead>
<tr>
<th>Authorities and Locks</th>
<th>Parameters</th>
<th>Return Value</th>
<th>Error Conditions</th>
<th>Error Messages</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>112</td>
<td>113</td>
<td>113</td>
<td>113</td>
<td>113</td>
</tr>
</tbody>
</table>

Return Value

<table>
<thead>
<tr>
<th>Error Conditions</th>
<th>Error Messages</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>113</td>
<td>113</td>
<td>113</td>
</tr>
</tbody>
</table>

ldap_modify_ext()—Perform an LDAP Modify Entry Request with Controls

<table>
<thead>
<tr>
<th>Authorities and Locks</th>
<th>Parameters</th>
<th>Return Value</th>
<th>Error Conditions</th>
<th>Error Messages</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>115</td>
<td>115</td>
<td>115</td>
<td>115</td>
<td>115</td>
<td>115</td>
</tr>
</tbody>
</table>

ldap_modify_ext_s()—Perform an LDAP Modify Entry Request (Synchronous)

<table>
<thead>
<tr>
<th>Authorities and Locks</th>
<th>Parameters</th>
<th>Return Value</th>
<th>Error Conditions</th>
<th>Error Messages</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>118</td>
<td>118</td>
<td>118</td>
<td>118</td>
<td>118</td>
<td>118</td>
</tr>
</tbody>
</table>

ldap_modrdn()—Perform an LDAP Modify RDN Request

<table>
<thead>
<tr>
<th>Authorities and Locks</th>
<th>Parameters</th>
<th>Return Value</th>
<th>Error Conditions</th>
<th>Error Messages</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>120</td>
<td>120</td>
<td>120</td>
<td>120</td>
<td>120</td>
<td>120</td>
</tr>
</tbody>
</table>

ldap_modrdn_s()—Perform an LDAP Modify RDN Request (Synchronous)

<table>
<thead>
<tr>
<th>Authorities and Locks</th>
<th>Parameters</th>
<th>Return Value</th>
<th>Error Conditions</th>
<th>Error Messages</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>122</td>
<td>122</td>
<td>122</td>
<td>122</td>
<td>122</td>
<td>122</td>
</tr>
</tbody>
</table>

ldap_mods_free()—Free LDAP Modify Storage

<table>
<thead>
<tr>
<th>Authorities and Locks</th>
<th>Parameters</th>
<th>Return Value</th>
<th>Error Conditions</th>
<th>Error Messages</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
</tr>
</tbody>
</table>

ldap_msgfree()—Free LDAP Result Message

<table>
<thead>
<tr>
<th>Authorities and Locks</th>
<th>Parameters</th>
<th>Return Values</th>
<th>Error Conditions</th>
<th>Error Messages</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>125</td>
<td>125</td>
<td>125</td>
<td>125</td>
<td>125</td>
<td>125</td>
</tr>
</tbody>
</table>

ldap_msgid()—Retrieve the Message ID Associated with an LDAP Message

<table>
<thead>
<tr>
<th>Authorities and Locks</th>
<th>Parameters</th>
<th>Return Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>126</td>
<td>126</td>
<td></td>
</tr>
</tbody>
</table>

Contents vii
Parameters .................................................. 127
Return Value ............................................... 127
Error Conditions ........................................... 127
Error Messages ............................................. 127
Related Information ........................................ 127
ldap_msgtype()—Retrieve the Type of an LDAP Message 128
Authorities and Locks ................................. 128
Parameters .................................................. 128
Return Value ............................................... 129
Error Conditions ........................................... 129
Error Messages ............................................. 129
Related Information ........................................ 129
ldap_next_attribute()—Retrieve Next Attribute in an Entry 129
Authorities and Locks ................................. 130
Parameters .................................................. 130
Return Value ............................................... 130
Error Conditions ........................................... 130
Error Messages ............................................. 130
Related Information ........................................ 131
ldap_next_entry()—Retrieve Next LDAP Entry 131
Authorities and Locks ................................. 131
Parameters .................................................. 131
Return Value ............................................... 132
Error Conditions ........................................... 132
Error Messages ............................................. 132
Related Information ........................................ 132
ldap_next_message()—Retrieve Next LDAP Message 132
Authorities and Locks ................................. 133
Parameters .................................................. 133
Return Value ............................................... 133
Error Conditions ........................................... 133
Error Messages ............................................. 133
Related Information ........................................ 133
ldap_next_reference()—Retrieve the next Continuation Reference in a Chain of Search Results 134
Authorities and Locks ................................. 134
Parameters .................................................. 134
Return Value ............................................... 134
Error Conditions ........................................... 134
Error Messages ............................................. 134
Related Information ........................................ 135
ldap_open()—Perform an LDAP Open Operation 135
Authorities and Locks ................................. 136
Parameters .................................................. 136
Return Value ............................................... 136
Error Conditions ........................................... 137
Error Messages ............................................. 137
Related Information ........................................ 137
ldap_parse_extended_result()—Parse extended result 138
Authorities and Locks ................................. 138
Parameters .................................................. 138
Return Value ............................................... 139
Error Conditions ........................................... 139
Error Messages ............................................. 139
Related Information ........................................ 139
ldap_parse_page_control()—Retrieve Values in a Paged Results Control 139
Authorities and Locks ................................. 140
Parameters .................................................. 140
Example ..................................................... 140
Return Value ............................................... 140
Related Information ........................................ 140
ldap_parse_pwdpolicy_response()—Obtain the LDAP Password Policy Error or Warning Codes 141
Authorities and Locks ................................. 142
Parameters .................................................. 142
Error Conditions ........................................... 142
Error Messages ............................................. 142
Related Information ........................................ 142
ldap_parse_reference_np()—Extract Information from a Continuation Reference 143
Authorities and Locks ................................. 143
Parameters .................................................. 143
Return Value ............................................... 144
Error Conditions ........................................... 144
Error Messages ............................................. 144
Related Information ........................................ 144
ldap_parse_result()—Extract Information from Results 144
Authorities and Locks ................................. 145
Parameters .................................................. 145
Return Value ............................................... 146
Error Conditions ........................................... 146
Error Messages ............................................. 146
Related Information ........................................ 146
ldap_parse_sasl_bind_result()—Extract Server Credentials from SASL Bind Results 146
Authorities and Locks ................................. 147
Parameters .................................................. 147
Return Value ............................................... 147
Error Conditions ........................................... 147
Error Messages ............................................. 147
Related Information ........................................ 147
ldap_parse_sort_control()—Retrieve Values in a Sorted Results Control 148
Authorities and Locks ................................. 148
Parameters .................................................. 148
Return Value ............................................... 148
Example ..................................................... 148
Related Information ........................................ 149
ldap_putstr()—Print LDAP Error Information 149
Authorities and Locks ................................. 149
Parameters .................................................. 149
Return Value ............................................... 150
Error Conditions ........................................... 150
Error Messages ............................................. 150
Related Information ........................................ 150
ldap_pwdpolicy_err2string()—Convert the Numeric LDAP Password Policy Error or Warning Code into a String 150
Authorities and Locks ................................. 150
Parameters .................................................. 151
Return Value ............................................... 151
Error Conditions ........................................... 151
Error Messages ............................................. 151
Related Information ........................................ 151
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ldap_set_iconv_local_codepage()</strong> — Set the Active LDAP Code Page</td>
<td></td>
<td>185</td>
</tr>
<tr>
<td>Authorities and Locks</td>
<td></td>
<td>185</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
<td>185</td>
</tr>
<tr>
<td>Return Value</td>
<td></td>
<td>186</td>
</tr>
<tr>
<td>Error Conditions</td>
<td></td>
<td>186</td>
</tr>
<tr>
<td>Error Messages</td>
<td></td>
<td>186</td>
</tr>
<tr>
<td>Related Information</td>
<td></td>
<td>186</td>
</tr>
<tr>
<td><strong>ldap_set_idermo()</strong> — Set Error Information</td>
<td></td>
<td>186</td>
</tr>
<tr>
<td>Authorities and Locks</td>
<td></td>
<td>187</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
<td>187</td>
</tr>
<tr>
<td>Return Value</td>
<td></td>
<td>187</td>
</tr>
<tr>
<td>Error Conditions</td>
<td></td>
<td>187</td>
</tr>
<tr>
<td>Error Messages</td>
<td></td>
<td>187</td>
</tr>
<tr>
<td>Related Information</td>
<td></td>
<td>187</td>
</tr>
<tr>
<td><strong>ldap_setlocale()</strong> — Change the Locale Used by LDAP</td>
<td></td>
<td>188</td>
</tr>
<tr>
<td>Authorities and Locks</td>
<td></td>
<td>188</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
<td>188</td>
</tr>
<tr>
<td>Return Value</td>
<td></td>
<td>188</td>
</tr>
<tr>
<td>Error Conditions</td>
<td></td>
<td>188</td>
</tr>
<tr>
<td>Error Messages</td>
<td></td>
<td>188</td>
</tr>
<tr>
<td>Related Information</td>
<td></td>
<td>189</td>
</tr>
<tr>
<td><strong>ldap_setoption()</strong> — Set LDAP Options.</td>
<td></td>
<td>189</td>
</tr>
<tr>
<td>Authorities and Locks</td>
<td></td>
<td>189</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
<td>189</td>
</tr>
<tr>
<td>LDAP_OPT_SIZELIMIT</td>
<td></td>
<td>190</td>
</tr>
<tr>
<td>LDAP_OPT_TIMELIMIT</td>
<td></td>
<td>191</td>
</tr>
<tr>
<td>LDAP_OPT_REFHOPLIMIT</td>
<td></td>
<td>191</td>
</tr>
<tr>
<td>LDAP_OPT_DEREF</td>
<td></td>
<td>191</td>
</tr>
<tr>
<td>LDAP_OPT_REFERRALS</td>
<td></td>
<td>191</td>
</tr>
<tr>
<td>LDAP_OPT_DEBUG</td>
<td></td>
<td>191</td>
</tr>
<tr>
<td>LDAP_OPT_SSL_CIPHER</td>
<td></td>
<td>192</td>
</tr>
<tr>
<td>LDAP_OPT_SSL_TIMEOUT</td>
<td></td>
<td>193</td>
</tr>
<tr>
<td>LDAP_OPT_REBIND_FNL</td>
<td></td>
<td>193</td>
</tr>
<tr>
<td>LDAP_OPT_PROTOCOL_VERSION</td>
<td></td>
<td>193</td>
</tr>
<tr>
<td>LDAP_OPT_SERVER_CONTROLS</td>
<td></td>
<td>193</td>
</tr>
<tr>
<td>LDAP_OPT_CLIENT_CONTROLS</td>
<td></td>
<td>193</td>
</tr>
<tr>
<td>LDAP_OPT_UTF8_1O</td>
<td></td>
<td>194</td>
</tr>
<tr>
<td>Return Value</td>
<td></td>
<td>194</td>
</tr>
<tr>
<td>Error Conditions</td>
<td></td>
<td>194</td>
</tr>
<tr>
<td>Error Messages</td>
<td></td>
<td>194</td>
</tr>
<tr>
<td>Related Information</td>
<td></td>
<td>194</td>
</tr>
<tr>
<td><strong>ldap_setrebindproc()</strong> — Set Rebind Procedure</td>
<td></td>
<td>195</td>
</tr>
<tr>
<td>Authorities and Locks</td>
<td></td>
<td>195</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
<td>195</td>
</tr>
<tr>
<td>Return Value</td>
<td></td>
<td>196</td>
</tr>
<tr>
<td>Error Conditions</td>
<td></td>
<td>196</td>
</tr>
<tr>
<td>Error Messages</td>
<td></td>
<td>196</td>
</tr>
<tr>
<td>Related Information</td>
<td></td>
<td>196</td>
</tr>
<tr>
<td><strong>ldap_simplebind()</strong> — Perform a Simple LDAP Bind Request</td>
<td></td>
<td>196</td>
</tr>
<tr>
<td>Authorities and Locks</td>
<td></td>
<td>197</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
<td>197</td>
</tr>
<tr>
<td>Return Value</td>
<td></td>
<td>197</td>
</tr>
<tr>
<td>Error Conditions</td>
<td></td>
<td>197</td>
</tr>
<tr>
<td>Error Messages</td>
<td></td>
<td>197</td>
</tr>
<tr>
<td>Related Information</td>
<td></td>
<td>197</td>
</tr>
<tr>
<td><strong>ldap_simplebind_s()</strong> — Perform a Simple LDAP Bind Request (Synchronous)</td>
<td></td>
<td>198</td>
</tr>
<tr>
<td>Authorities and Locks</td>
<td></td>
<td>198</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
<td>198</td>
</tr>
<tr>
<td>Return Value</td>
<td></td>
<td>198</td>
</tr>
<tr>
<td>Error Conditions</td>
<td></td>
<td>199</td>
</tr>
<tr>
<td>Error Messages</td>
<td></td>
<td>199</td>
</tr>
<tr>
<td>Related Information</td>
<td></td>
<td>199</td>
</tr>
<tr>
<td><strong>ldap_sslclientinit()</strong> — Initialize the SSL Library.</td>
<td></td>
<td>199</td>
</tr>
<tr>
<td>Authorities and Locks</td>
<td></td>
<td>200</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
<td>200</td>
</tr>
<tr>
<td>Example</td>
<td></td>
<td>200</td>
</tr>
<tr>
<td>Return Value</td>
<td></td>
<td>201</td>
</tr>
<tr>
<td>Error Conditions</td>
<td></td>
<td>201</td>
</tr>
<tr>
<td>Error Messages</td>
<td></td>
<td>202</td>
</tr>
<tr>
<td>Related Information</td>
<td></td>
<td>202</td>
</tr>
<tr>
<td><strong>ldap_sslenvironmentinit()</strong> — Initialize SSL for a secure connection</td>
<td></td>
<td>202</td>
</tr>
<tr>
<td>Authorities and Locks</td>
<td></td>
<td>203</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
<td>203</td>
</tr>
<tr>
<td>Return Value</td>
<td></td>
<td>203</td>
</tr>
<tr>
<td>Error Conditions</td>
<td></td>
<td>203</td>
</tr>
<tr>
<td>Error Messages</td>
<td></td>
<td>204</td>
</tr>
<tr>
<td>Related Information</td>
<td></td>
<td>204</td>
</tr>
<tr>
<td><strong>ldap_sslinit</strong> — Initialize an SSL Connection.</td>
<td></td>
<td>204</td>
</tr>
<tr>
<td>Authorities and Locks</td>
<td></td>
<td>204</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
<td>204</td>
</tr>
<tr>
<td>Example</td>
<td></td>
<td>205</td>
</tr>
<tr>
<td>Return Value</td>
<td></td>
<td>206</td>
</tr>
<tr>
<td>Error Conditions</td>
<td></td>
<td>206</td>
</tr>
<tr>
<td>Error Messages</td>
<td></td>
<td>206</td>
</tr>
<tr>
<td>Related Information</td>
<td></td>
<td>206</td>
</tr>
<tr>
<td><strong>ldap_sslstart()</strong> — Start a Secure LDAP Connection</td>
<td></td>
<td>207</td>
</tr>
<tr>
<td>Authorities and Locks</td>
<td></td>
<td>207</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
<td>207</td>
</tr>
<tr>
<td>Return Value</td>
<td></td>
<td>208</td>
</tr>
<tr>
<td>Error Conditions</td>
<td></td>
<td>208</td>
</tr>
<tr>
<td>Error Messages</td>
<td></td>
<td>208</td>
</tr>
<tr>
<td>Related Information</td>
<td></td>
<td>208</td>
</tr>
<tr>
<td><strong>ldap_stoptls_snp()</strong> — End a TLS Session</td>
<td></td>
<td>209</td>
</tr>
<tr>
<td>Authorities and Locks</td>
<td></td>
<td>209</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
<td>209</td>
</tr>
<tr>
<td>Return Value</td>
<td></td>
<td>209</td>
</tr>
<tr>
<td>Error Conditions</td>
<td></td>
<td>209</td>
</tr>
<tr>
<td>Error Messages</td>
<td></td>
<td>209</td>
</tr>
<tr>
<td>Related Information</td>
<td></td>
<td>209</td>
</tr>
<tr>
<td><strong>ldap_unbind()</strong> — Perform an LDAP Unbind Request</td>
<td></td>
<td>209</td>
</tr>
<tr>
<td>Authorities and Locks</td>
<td></td>
<td>210</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
<td>210</td>
</tr>
<tr>
<td>Return Value</td>
<td></td>
<td>210</td>
</tr>
<tr>
<td>Error Conditions</td>
<td></td>
<td>210</td>
</tr>
<tr>
<td>Error Messages</td>
<td></td>
<td>210</td>
</tr>
<tr>
<td>Related Information</td>
<td></td>
<td>210</td>
</tr>
<tr>
<td><strong>ldap_unbind_ext()</strong> — Perform an LDAP Unbind Request</td>
<td></td>
<td>210</td>
</tr>
<tr>
<td>Authorities and Locks</td>
<td></td>
<td>211</td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
<td>211</td>
</tr>
<tr>
<td>Return Value</td>
<td></td>
<td>212</td>
</tr>
<tr>
<td>Error Conditions</td>
<td></td>
<td>212</td>
</tr>
<tr>
<td>Error Messages</td>
<td></td>
<td>212</td>
</tr>
<tr>
<td>Related Information</td>
<td></td>
<td>212</td>
</tr>
<tr>
<td><strong>ldap_unbinds()</strong> — Perform an LDAP Unbind Request (Synchronous)</td>
<td></td>
<td>212</td>
</tr>
<tr>
<td>ldap_url_parse() — Parse an LDAP URL</td>
<td>214</td>
<td></td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>-----</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>215</td>
<td></td>
</tr>
<tr>
<td>Return Value</td>
<td>215</td>
<td></td>
</tr>
<tr>
<td>Error Conditions</td>
<td>215</td>
<td></td>
</tr>
<tr>
<td>Error Messages</td>
<td>215</td>
<td></td>
</tr>
<tr>
<td>Related Information</td>
<td>215</td>
<td></td>
</tr>
<tr>
<td>ldap_url_search() — Perform an LDAP URL Search Operation</td>
<td>217</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>217</td>
<td></td>
</tr>
<tr>
<td>Return Value</td>
<td>217</td>
<td></td>
</tr>
<tr>
<td>Error Conditions</td>
<td>217</td>
<td></td>
</tr>
<tr>
<td>Error Messages</td>
<td>217</td>
<td></td>
</tr>
<tr>
<td>Related Information</td>
<td>217</td>
<td></td>
</tr>
<tr>
<td>ldap_url_search_s() — Perform an LDAP URL Search Operation (Synchronous)</td>
<td>219</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>219</td>
<td></td>
</tr>
<tr>
<td>Return Value</td>
<td>219</td>
<td></td>
</tr>
<tr>
<td>Error Conditions</td>
<td>219</td>
<td></td>
</tr>
<tr>
<td>Error Messages</td>
<td>219</td>
<td></td>
</tr>
<tr>
<td>Related Information</td>
<td>219</td>
<td></td>
</tr>
<tr>
<td>ldap_url_search_st() — Perform an LDAP URL Search Operation (Timed Synchronous)</td>
<td>221</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>221</td>
<td></td>
</tr>
<tr>
<td>Return Value</td>
<td>221</td>
<td></td>
</tr>
<tr>
<td>Error Conditions</td>
<td>221</td>
<td></td>
</tr>
<tr>
<td>Error Messages</td>
<td>221</td>
<td></td>
</tr>
<tr>
<td>Related Information</td>
<td>221</td>
<td></td>
</tr>
<tr>
<td>ldap_value_free() — Free Memory Allocated by ldap_get_values()</td>
<td>223</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>223</td>
<td></td>
</tr>
<tr>
<td>Return Value</td>
<td>223</td>
<td></td>
</tr>
<tr>
<td>Error Conditions</td>
<td>223</td>
<td></td>
</tr>
<tr>
<td>Error Messages</td>
<td>223</td>
<td></td>
</tr>
<tr>
<td>Related Information</td>
<td>223</td>
<td></td>
</tr>
<tr>
<td>ldap_value_free_len() — Free Memory Allocated by ldap_get_values_len()</td>
<td>224</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>224</td>
<td></td>
</tr>
<tr>
<td>Return Value</td>
<td>224</td>
<td></td>
</tr>
<tr>
<td>Error Conditions</td>
<td>224</td>
<td></td>
</tr>
<tr>
<td>Error Messages</td>
<td>224</td>
<td></td>
</tr>
<tr>
<td>Related Information</td>
<td>224</td>
<td></td>
</tr>
<tr>
<td>ldap_version — Obtain LDAP Version and SSL Cipher Information</td>
<td>225</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>225</td>
<td></td>
</tr>
<tr>
<td>Return Value</td>
<td>225</td>
<td></td>
</tr>
<tr>
<td>Error Conditions</td>
<td>225</td>
<td></td>
</tr>
<tr>
<td>Error Messages</td>
<td>225</td>
<td></td>
</tr>
<tr>
<td>Related Information</td>
<td>225</td>
<td></td>
</tr>
<tr>
<td>ldap_xlate_local_to_unicode() — Convert String from the Local Code Page to UCS-2 (or UNICODE) Encoding</td>
<td>226</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>226</td>
<td></td>
</tr>
<tr>
<td>Return Value</td>
<td>226</td>
<td></td>
</tr>
<tr>
<td>Error Conditions</td>
<td>226</td>
<td></td>
</tr>
<tr>
<td>Error Messages</td>
<td>226</td>
<td></td>
</tr>
<tr>
<td>ldap_xlate_unicode_to_local() — Convert String from the UCS-2 (or UNICODE) Encoding to Local Code Page</td>
<td>229</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>230</td>
<td></td>
</tr>
<tr>
<td>Return Value</td>
<td>230</td>
<td></td>
</tr>
<tr>
<td>Error Conditions</td>
<td>230</td>
<td></td>
</tr>
<tr>
<td>Error Messages</td>
<td>230</td>
<td></td>
</tr>
<tr>
<td>Related Information</td>
<td>230</td>
<td></td>
</tr>
<tr>
<td>ldap_xlate_utf8_to_local() — Convert String from the UTF-8 Encoding to Local Code Page</td>
<td>231</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>231</td>
<td></td>
</tr>
<tr>
<td>Return Value</td>
<td>231</td>
<td></td>
</tr>
<tr>
<td>Error Conditions</td>
<td>231</td>
<td></td>
</tr>
<tr>
<td>Error Messages</td>
<td>231</td>
<td></td>
</tr>
<tr>
<td>Related Information</td>
<td>231</td>
<td></td>
</tr>
<tr>
<td>Configure Directory Server (QdldCfgDirSvr)</td>
<td>232</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>233</td>
<td></td>
</tr>
<tr>
<td>Return Value</td>
<td>233</td>
<td></td>
</tr>
<tr>
<td>Error Conditions</td>
<td>233</td>
<td></td>
</tr>
<tr>
<td>Error Messages</td>
<td>233</td>
<td></td>
</tr>
<tr>
<td>Related Information</td>
<td>233</td>
<td></td>
</tr>
<tr>
<td>CFGD0100 Format</td>
<td>233</td>
<td></td>
</tr>
<tr>
<td>Field Descriptions</td>
<td>234</td>
<td></td>
</tr>
<tr>
<td>Error Messages</td>
<td>235</td>
<td></td>
</tr>
<tr>
<td>List Authorized Users (QSULAUTU) API</td>
<td>235</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>235</td>
<td></td>
</tr>
<tr>
<td>Return Value</td>
<td>235</td>
<td></td>
</tr>
<tr>
<td>Error Conditions</td>
<td>235</td>
<td></td>
</tr>
<tr>
<td>Error Messages</td>
<td>235</td>
<td></td>
</tr>
<tr>
<td>User Space Variables</td>
<td>236</td>
<td></td>
</tr>
<tr>
<td>Error Messages</td>
<td>236</td>
<td></td>
</tr>
<tr>
<td>Change Directory Server Attributes (QdldChgDirSvrA)</td>
<td>238</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>239</td>
<td></td>
</tr>
<tr>
<td>Required Parameter Group</td>
<td>239</td>
<td></td>
</tr>
<tr>
<td>Format of Input Data</td>
<td>239</td>
<td></td>
</tr>
<tr>
<td>CSVR0100 Format</td>
<td>241</td>
<td></td>
</tr>
<tr>
<td>CSVR0200 Format</td>
<td>242</td>
<td></td>
</tr>
<tr>
<td>CSVR0300 Format</td>
<td>243</td>
<td></td>
</tr>
<tr>
<td>CSVR0400 Format</td>
<td>243</td>
<td></td>
</tr>
<tr>
<td>CSVR0500 Format</td>
<td>244</td>
<td></td>
</tr>
<tr>
<td>CSVR0600 Format</td>
<td>244</td>
<td></td>
</tr>
</tbody>
</table>
Lightweight Directory Access Protocol (LDAP) APIs

The Lightweight Directory Access Protocol (LDAP) client APIs can be used to access LDAP-enabled directories in a network. Administrative and configuration APIs for IBM(R) Directory Server for iSeries(TM) are included.

Select one of the following for more information:

- "LDAP API Overview” on page 296
- “LDAP Version Support” on page 298
- “Accessing Schema Information” on page 299
- “API Prototype Changes” on page 300
- “Deprecated APIs” on page 300
- “LDAP Client API Error Conditions” on page 300

The Lightweight Directory Access Protocol (LDAP) APIs are:

- "ldap_abandon()—Abandon an LDAP Operation in Progress” on page 6 (Abandon an LDAP Operation in Progress)
- "ldap_abandon_ext()—Abandon (abort) an Asynchronous Operation with Controls” on page 8 (Abandon an LDAP Operation with Controls)
- "ldap_add()—Perform an LDAP Add Operation” on page 9 (Perform an LDAP Add Operation)
- "ldap_add_control()—Create a Control and Insert it into the List of LDAP Server Controls” on page 10 (Add New Server Control)
- "ldap_add_ext()—Perform an LDAP Add Operation with Controls” on page 11 (Perform an LDAP Add Operation with Controls)
- "ldap_add_ext_s()—Perform an LDAP Add Operation with Controls (Synchronous)” on page 13 (Perform an LDAP Add Operation with Controls (Synchronous))
- "ldap_add_s()—Perform an LDAP Add Operation (Synchronous)” on page 14 (Perform an LDAP Add Operation (Synchronous))
- "ldap_app_ssl_client_init_np()—Initialize the LDAP Client for a Secure Connection using DCM” on page 15 (Initialize the LDAP Client for a Secure Connection using DCM)
- "ldap_app_ssl_init_np —Initialize an SSL Connection” on page 18 (Initialize an SSL Connection)
- "ldap_app_ssl_start_np()—Start a Secure LDAP Connection using DCM” on page 20 (Start a Secure LDAP Connection using DCM)
- "ldap_ber_free()—Free storage allocated by the LDAP library” on page 22 (Free storage allocated for BerElement)
- "ldap_bind()—Perform an LDAP Bind Request” on page 23 (Perform an LDAP Bind Request)
- "ldap_bind_s()—Perform an LDAP Bind Request (Synchronous)” on page 25 (Perform an LDAP Bind Request (Synchronous))
- "ldap_compare()—Perform an LDAP Compare Operation” on page 26 (Perform an LDAP Compare Operation)
- "ldap_compare_ext()—Perform an LDAP Compare Operation with Controls” on page 28 (Perform an LDAP Compare Operation with Controls)
- "ldap_compare_ext_s()—Perform an LDAP Compare Operation with Controls (Synchronous)” on page 30 (Perform an LDAP Compare Operation with Controls (Synchronous))
- "ldap_compare_s()—Perform an LDAP Compare Operation (Synchronous)” on page 31 (Perform an LDAP Compare Operation (Synchronous))
v “ldap_control_free()—Free storage allocated by the LDAP library” on page 33 (Free Storage Allocated
by the LDAP Library)
v “ldap_controls_free()—Free storage allocated by the LDAP library” on page 32 (Free an array of
LDAPControl structures)
v “ldap_copy_controls()—Make a Copy of the List of LDAP Server Controls” on page 34 (Copy Control
List)
v “ldap_count_attributes()—Retrieve Count of Attributes for an LDAP Entry” on page 35 (Retrieve Count
of Attributes for an LDAP Entry)
v “ldap_count_entries()—Retrieve Count of LDAP Entries” on page 36 (Retrieve Count of LDAP Entries)
v “ldap_count_messages()—Count messages in a result chain” on page 38 (Count messages in a result
chain, as returned by ldap_result)
v “ldap_count_references()—Count continuation references in a chain of search results” on page 39
(Count continuation references in a result chain of search results)
v “ldap_count_values()—Retrieve Count of Attribute Values” on page 40 (Retrieve Count of Attribute
Values)
v “ldap_count_values_len()—Retrieve Count of Binary Attribute Values” on page 41 (Retrieve Count of
Binary Attribute Values)
v “ldap_create_page_control()—Create a Paged Results Control used when paging search results” on
page 42 (Create a Paged Results Control)
v
“ldap_create_proxyauth_control()—Create Control for Proxy Identity” on page 47 (Create Control for
Proxy Identity)
v “ldap_create_sort_control()—Create a Sorted Results Control used when Sorting Search Results” on
page 49 (Create a Sorted Results Control)
v “ldap_create_sort_keylist()—Create a Structure with Sort Key Values used when Sorting Search
Results” on page 51 (Create a Structure with Sort Key Values)
v “ldap_default_dn_get()— Retrieve the User’s Default DN” on page 56 (Retrieve the User’s Default DN)
v “ldap_default_dn_set()— Store the User’s Default DN” on page 57 (Store the User’s Default DN)
v “ldap_delete()—Perform an LDAP Delete Operation” on page 59 (Perform an LDAP Delete Operation)
v “ldap_delete_ext()—Perform an LDAP Delete Operation with Controls” on page 60 (Perform an LDAP
Delete Operation with Controls)
v “ldap_delete_ext_s()—Perform an LDAP Delete Operation with Controls” on page 61 (Perform an
LDAP Delete Operation with Controls (Synchronous))
v “ldap_delete_s()—Perform an LDAP Delete Operation (Synchronous)” on page 63 (Perform an LDAP
Delete Operation (Synchronous))
v “ldap_dn2ufn()—Convert a Distinguished Name into a User Friendly Name” on page 64 (Convert a
Distinguished Name into a User Friendly Name)
v “ldap_enetwork_domain_get()— Retrieve the User’s Default eNetwork Domain Name” on page 65
(Retrieve the User’s Default eNetwork Domain Name)
v “ldap_enetwork_domain_set()— Store the User’s Default eNetwork Domain Name” on page 66 (Store
the User’s Default eNetwork Domain Name)
v “ldap_err2string()—Retrieve LDAP Error Message String” on page 68 (Retrieve LDAP Error Message
String)
v “ldap_explode_dn()—Break a Distinguished Name into Its Components” on page 69 (Break a
Distinguished Name into Its Components)
v “ldap_explode_dn_utf8()—Break a UTF8 codepage Distinguished Name into Its Components” on page
71 (Break a UTF8 codepage Distinguished Name into Its Components)
v “ldap_explode_dns()—Break a DNS-style Distinguished Name into Its Components” on page 70 (Break
a DNS-style Distinguished Name into Its Components)

2

IBM Systems - iSeries: Lightweight Directory Access Protocol (LDAP) APIs


• "ldap_explode_rdn()—Break a Relative Distinguished Name into Its Components” on page 72 (Break a Relative Distinguished Name into Its Components)

• "ldap_explode_rdn_utf8()—Break a UTF8 codepage Relative Distinguished Name into Its Components” on page 74 (Break a UTF8 codepage Relative Distinguished Name into Its Components)

• "ldap_extended_operation()—Perform extended operations.” on page 75 (Perform extended operations)

• "ldap_extended_operation_s()—Perform extended operations synchronously” on page 77 (Perform extended operations synchronously)

• "ldap_first_attribute()—Retrieve First Attribute in an Entry” on page 79 (Retrieve First Attribute in an Entry)

• "ldap_first_entry()—Retrieve First LDAP Entry” on page 81 (Retrieve First LDAP Entry)

• "ldap_first_message()—Retrieve First LDAP Message” on page 82 (Retrieve First LDAP Message)

• "ldap_first_reference()—Retrieve First Continuation Reference in a Chain of Search Results” on page 84 (Return first continuation reference in a chain of search results)

• "ldap_free_sort_keylist()—Free all Memory used by the Sort Key List” on page 85 (Free the Sort Key List Structure)

• "ldap_free_urldesc()—Free an LDAP URL Description” on page 86 (Retrieve the Distinguished Name of an Entry)

• "ldap_get_bind_controls()—Get LDAP Bind Controls” on page 88 (Get LDAP Bind Controls ) get controls sent by a LDAP server during a bind operation

• "ldap_get_dn()—Retrieve the Distinguished Name of an Entry” on page 89 (Extract the DN from an entry)

• "ldap_get_entry_controls_np()—Extract Server Controls from an Entry” on page 90 (Extract server controls from an entry)

• "ldap_get_errno()—Retrieve Error Information” on page 91 (Retrieve Error Information)

• "ldap_get_iconv_local_codepage()— Get the Active LDAP Code Page” on page 92 (Get the Active LDAP Code Page)

• "ldap_get_lderrno()—Retrieve Error Information” on page 93 (Retrieve Error Information)

• "ldap_get_locale()— Get Active LDAP Locale” on page 95 (Get Active LDAP Locale)

• "ldap_get_option()—Retrieve LDAP Options” on page 96 (Retrieve LDAP Options)

• "ldap_get_values()—Retrieve a Set of Attribute Values from an Entry” on page 102 (Retrieve a Set of Attribute Values from an Entry)

• "ldap_get_values_len()—Retrieve a Set of Binary Attribute Values” on page 104 (Retrieve a Set of Binary Attribute Values)

• "ldap_init()—Perform an LDAP Initialization Operation” on page 105 (Perform an LDAP Initialization Operation)

• "ldap_insert_control()—Insert a Control in the Llist of LDAP Server Controls” on page 108 (Insert Control into Control List)

• "ldap_is_ldap_url()—Verify LDAP URL” on page 108 (Verify LDAP URL)

• "ldap_is_ldap_url_utf8()—Verify UTF-8 codepage LDAP URL” on page 109 (Verify UTF-8 Codepage LDAP URL) verifies UTF-8 codepage string is an LDAP URL. 

• "ldap_memfree()—Free Memory Allocated by LDAP API” on page 110 (Free Memory Allocated by LDAP API)

• "ldap_modify()—Perform an LDAP Modify Entry Request” on page 111 (Perform an LDAP Modify Entry Request)

• "ldap_modify_ext()—Perform an LDAP Modify Entry Request with Controls” on page 113 (Perform an LDAP Modify Entry Request with Controls)

• "ldap_modify_ext_s()—Perform an LDAP Modify Entry Request with Controls” on page 116 (Perform an LDAP Modify Entry Request with Controls (Synchronous))
• "ldap_modify_s()—Perform an LDAP Modify Entry Request (Synchronous)" on page 118 (Perform an LDAP Modify Entry Request (Synchronous))
• "ldap_modrdn()—Perform an LDAP Modify RDN Request” on page 120 (Perform an LDAP Modify RDN Request)
• "ldap_modrdn_s()—Perform an LDAP Modify RDN Request (Synchronous)” on page 122 (Perform an LDAP Modify RDN Request (Synchronous))
• "ldap_mods_free()—Free LDAP Modify Storage” on page 123 (Free LDAP Modify Storage)
• "ldap_msfree()—Free LDAP Result Message” on page 125 (Free LDAP Result Message)
• "ldapmsgid()—Retrieve the Message ID Associated with an LDAP Message” on page 126 (Retrieve Message ID Associated with an LDAP Message)
• "ldap_msgtype()—Retrieve the Type of an LDAP Message” on page 128 (Retrieve Type of an LDAP Message)
• "ldap_next_attribute()—Retrieve Next Attribute in an Entry” on page 129 (Retrieve Next Attribute in an Entry)
• "ldap_next_entry()—Retrieve Next LDAP Entry” on page 131 (Retrieve Next LDAP Entry)
• "ldap_next_message()—Retrieve Next LDAP Message” on page 132 (Retrieve Next LDAP Message)
• "ldap_next_reference()—Retrieve the next Continuation Reference in a Chain of Search Results” on page 134 (Retrieve Next Continuation Reference in a Chain of Search Results)
• "ldap_open()—Perform an LDAP Open Operation” on page 135 (Perform an LDAP Open Operation)
• "ldap_parse_extended_result()—Parse extended result” on page 138 (Parse extended result)
• "ldap_parse_page_control()—Retrieve Values in a Paged Results Control” on page 139 (Retrieve Values in a Paged Results Control)
• "ldap_parse_pwdpolicy_response()—Obtain the LDAP Password Policy Error or Warning Codes” on page 141 (Obtain Error and Warning Codes from the Password Policy Response Control)
• "ldap_parse_reference_np()—Extract Information from a Continuation Reference” on page 143 (Extract information from a continuation reference)
• "ldap_parse_result()—Extract Information from Results” on page 144 (Extract information from results)
• "ldap_parse_sasl_bind_result()—Extract Server Credentials from SASL Bind Results” on page 146 (Extract server credentials from SASL bind results)
• "ldap_parse_sort_control()—Retrieve Values in a Sorted Results Control” on page 148 (Retrieve Values in a Sorted Results Control)
• "ldap_error()—Print LDAP Error Information” on page 149 (Print LDAP Error Information)
• "ldap_pwdpolicy_err2string()—Convert the Numeric LDAP Password Policy Error or Warning Code into a String” on page 150 (Convert Numeric Password Policy Error or Warning Code into a Message String)
• "ldap_remove_control()—Remove a Control from the List of LDAP Server Controls” on page 151 (Remove Server Control)
• "ldap_rename()—Asynchronously Rename an Entry” on page 152 (Asynchronously rename an entry)
• "ldap_rename_s()—Synchronously Rename an Entry” on page 154 (Synchronously rename an entry)
• "ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156 (Retrieve Result of an Asynchronous LDAP Operation)
• "ldap_result2error()—Retrieve LDAP Error Information” on page 158 (Retrieve LDAP Error Information)
• "ldap_sasl_bind()—Perform an LDAP SASL Bind Request” on page 159 (Perform an LDAP SASL Bind Request)
• "ldap_sasl_bind_s()—Perform an LDAP SASL Bind Request (Synchronous)” on page 161 (Perform an LDAP SASL Bind Request (Synchronous))
• "ldap_search()—Perform an LDAP Search Operation” on page 164 (Perform an LDAP Search Operation)
• “ldap_search_ext — Asynchronously Search the Directory Using Controls” on page 166
  (Asynchronously search the directory using controls)
• “ldap_search_ext_s — Synchronously Search the Directory Using Controls” on page 168
  (Synchronously search the directory using controls)
• “ldap_search_s()—Perform an LDAP Search Operation (Synchronous)” on page 171
  (Perform an LDAP Search Operation (Synchronous))
• “ldap_search_st()—Perform an LDAP Search Operation (Timed Synchronous)” on page 173
  (Perform an LDAP Search Operation (Timed Synchronous))
• “ldap_server_conf_save()—Store Server Information into Local Configuration” on page 175
  (Store Server Information into Local Configuration)
• “ldap_server_free_list()—Free the List of LDAP Servers” on page 177
  (Free the List of LDAP Servers)
• “ldap_server_locate()—Locate Suitable LDAP Servers” on page 178
  (Locate Suitable LDAP Servers)
• “ldap_set_iconv_local_charset()—Set the Active LDAP Character Set” on page 183
  (Set the Active LDAP Character Set)
• “ldap_set_iconv_local_codepage() — Set the Active LDAP Code Page” on page 185
  (Set the Active LDAP Code Page)
• “ldap_set_idErrno() — Set Error Information” on page 186 (Set Error Information)
• “ldap_set_locale() — Change the Locale Used by LDAP” on page 188
  (Change the Locale Used by LDAP)
• “ldap_set_option() — Set LDAP Options” on page 189
  (Set LDAP Options)
• “ldap_set_rebind_proc()—Set Rebind Procedure” on page 195
  (Set Rebind Procedure)
• “ldap_simple_bind()—Perform a Simple LDAP Bind Request” on page 196
  (Perform a Simple LDAP Bind Request)
• “ldap_simple_bind_s()—Perform a Simple LDAP Bind Request (Synchronous)” on page 198
  (Perform a Simple LDAP Bind Request (Synchronous))
• “ldap_ssl_client_init” on page 199 (Initialize the SSL library)
  ➯ “ldap_ssl_environment_init()—Initialize SSL for a secure connection” on page 202
  (Initialize SSL for a secure connection)
• “ldap_ssl_init — Initialize an SSL Connection.” on page 204
  (Initialize an SSL connection)
• “ldap_ssl_start()—Start a Secure LDAP Connection” on page 207
  (Start a Secure LDAP Connection)
  ➯ “ldap_start_tls_app_np()” (Start a TLS Session with an Application ID)
  ➯ “ldap_start_tls_s_np()” (Start a TLS Session with a Certificate)
  ➯ “ldap_stop_tls_s_np()” (End a TLS Session)
• “ldap_unbind()—Perform an LDAP Unbind Request” on page 209
  (Perform an LDAP Unbind Request)
• “ldap_unbind_ext()—Perform an LDAP Unbind Request” on page 211
  (Perform an LDAP Unbind Request)
• “ldap_unbind_s()—Perform an LDAP Unbind Request (Synchronous)” on page 212
  (Perform an LDAP Unbind Request (Synchronous))
• “ldap_url_parse()—Parse an LDAP URL” on page 214
  (Parse an LDAP URL)
• “ldap_url_parse_utf8()—Parse a UTF8 codepage LDAP URL string” on page 215
  (Parse a UTF8 codepage LDAP URL string)
• “ldap_url_search()—Perform an LDAP URL Search Operation” on page 217
  (Perform an LDAP URL Search Operation)
• “ldap_url_search_s() — Perform an LDAP URL Search Operation (Synchronous)” on page 219
  (Perform an LDAP URL Search Operation (Synchronous))
• “ldap_url_search_st()—Perform an LDAP URL Search Operation (Timed Synchronous)” on page 221
  (Perform an LDAP URL Search Operation (Timed Synchronous))
ldap_abandon()—Abandon an LDAP Operation in Progress

Syntax
#include <ldap.h>

int ldap_abandon(
    LDAP  *ld,
    int    msgid)

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes
The `ldap_abandon()` function is used to abandon or cancel an LDAP operation in progress. The `msgid` passed should be the message ID of an outstanding LDAP operation, as returned by a call to an asynchronous LDAP operation such as `ldap_search()`, `ldap_modify()`, and so on.

The `ldap_abandon()` APIs check to see if the result of the operation has already been returned by the server. If it has, it deletes it from the queue of pending messages. If not, it sends an LDAP abandon operation to the the LDAP server.

The caller can expect that the result of an abandoned operation will not be returned from a future call to `ldap_result()`.

** Authorities and Locks

No i5/OS authority is required.

** Parameters

- `ld` (Input) The LDAP pointer returned by a previous call to `ldap_init()`—Perform an LDAP Initialization Operation” on page 105, “ldap_ssl_init—Initialize an SSL Connection.” on page 204 or “ldap_open()—Perform an LDAP Open Operation” on page 135.

- `msgid` (Input) The message ID of an outstanding LDAP operation, as returned by a call to an asynchronous LDAP operation such as “ldap_search()—Perform an LDAP Search Operation” on page 164 or “ldap_modify()—Perform an LDAP Modify Entry Request” on page 111.

** Return Value

- `LDAP_SUCCESS` if the request was successful.
- `-1` if the request was not successful.

** Error Conditions

If `ldap_abandon()` is not successful, `ld_errno` will be set to indicate the error. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error codes values and “ldap_get_errno()—Retrieve Error Information” on page 91 function to retrieve the error information.

** Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_abandon API.</td>
</tr>
</tbody>
</table>

** Related Information

- “`ldap_abandon_ext()`—Abandon (abort) an Asynchronous Operation with Controls” on page 8 — Abandon (abort) an asynchronous operation with controls.

API introduced: V4R3
ldap_abandon_ext()—Abandon (abort) an Asynchronous Operation with Controls

Syntax
#include <ldap.h>

int ldap_abandon_ext(
   LDAP *ld,
   intmsgid,
   LDAPControl **serverctrls,
   LDAPControl **clientctrls)

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

The ldap_abandon_ext() function is used to abandon or cancel an LDAP operation in progress. The msgid passed should be the message ID of an outstanding LDAP operation, as returned by a call to an asynchronous LDAP operation such as ldap_search(), ldap_modify(), and so on.

This API checks to see if the result of the operation has already been returned by the server. If it has, the result is removed from the queue of pending messages. If not, it sends an LDAP abandon operation to the the LDAP server.

The caller can expect that the result of an abandoned operation will not be returned from a future call to ldap_result().

Authorities and Locks
No i5/OS authority is required.

Parameters

Id (Input) The LDAP pointer returned by a previous call to ["ldap_init”—Perform an LDAP Initialization Operation” on page 105]; ["ldap_ssl_init—Initialize an SSL Connection.” on page 204]; or ["ldap_open”—Perform an LDAP Open Operation” on page 135].

msgid (Input) The message ID of an outstanding LDAP operation, as returned by a call to an asynchronous LDAP operation such as ldap_search or ldap_modify.

serverctrls (Input) A list of LDAP server controls. This parameter may be set to null. See LDAP Controls for more information about server controls.

clientctrls (Input) A list of LDAP client controls. This parameter may be set to null. See LDAP Controls for more information about client controls.

Return Value

LDAP_SUCCESS
   if the ldap_abandon() was successful.

Other LDAP error code
   if the request was not successful.

Error Conditions
If ldap_abandon_ext() is not successful, LDAP error code will be returned. See ["LDAP Client API Error Conditions” on page 300] for possible LDAP error codes values.
Error Messages
The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_abandon_ext API.</td>
</tr>
</tbody>
</table>

Related Information

- "ldap_abandon()—Abandon an LDAP Operation in Progress” on page 6 — Abandon (abort) an asynchronous operation.

API introduced: V4R5

Syntax

```
#include <ldap.h>

int ldap_add(  
  LDAP *ld,  
  const char *dn,  
  LDAPMod **attrs)
```

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

The `ldap_add()` function is used to perform an LDAP add operation.

Authorities and Locks
No i5/OS authority is required. All authority checking is done by the LDAP server.

Parameters

- **ld** (Input) The LDAP pointer returned by a previous call to "ldap_init()—Perform an LDAP Initialization Operation” on page 105, "ldap_ssl_init — Initialize an SSL Connection.” on page 204, or "ldap_open()—Perform an LDAP Open Operation” on page 135.
- **dn** (Input) The DN of the entry to add.
- **attrs** (Input) The entry’s attributes, specified using the LDAPMod structure, as defined for "ldap_modify()—Perform an LDAP Modify Entry Request” on page 111. The `mod_type` and `mod_vals` fields should be filled in. The `mod_op` field is ignored unless ORed with the constant LDAP_MOD_BVALUES. In this case, the `mod_op` field is used to select the `mod_bvalues` case of the `mod_vals` union.

Return Value

Message ID of the operation initiated if the request was successfully sent. A subsequent call to "ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156, can be used to obtain the result of the operation.

-1 if the request was not successful.
Error Conditions

If `ldap_add()` is not successful, `ld_errno` will be set to indicate the error. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values and “ldap_get_errno()—Retrieve Error Information” on page 91 function to retrieve the error information.

Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_add API.</td>
</tr>
</tbody>
</table>

Related Information

- “ldap_add_s()—Perform an LDAP Add Operation (Synchronous)” on page 14 — Synchronously add an entry.
- “ldap_add_ext()—Perform an LDAP Add Operation with Controls” on page 11 — Asynchronously add an entry with controls.
- “ldap_add_ext_s()—Perform an LDAP Add Operation with Controls (Synchronous)” on page 13 — Synchronously add an entry with controls.
- “ldap_modify()—Perform an LDAP Modify Entry Request” on page 111 — Asynchronously modify an entry.

API introduced: V4R3

`ldap_add_control()`—Create a Control and Insert it into the List of LDAP Server Controls

Syntax

```
#include <ldap.h>

int ldap_add_control(
    char *oid,
    ber_len_t len,
    char *value,
    int isCritical,
    LDAPControl ***ctrlList)
```

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

The `ldap_add_control()` function is used to create a control and insert it into the list of LDAP server controls.

Note: The function will allocate space in the list for the control.

Authorities and Locks

No i5/OS authority is required. All authority checking is done by the LDAP server.
Parameters

oid  (Input) Specifies the control type, represented as a string.

len  (Input) Specifies the length of the value string.

value  (Input) Specifies the data associated with the control.

isCritical  (Input) Specifies whether the control is critical or not.

ctrlList  (Input) Specifies a list of LDAP server controls. See [LDAP Controls] for more information about server controls.

Return Value

LDAP_SUCCESS  if the request was successful.

LDAP_NO_MEMORY  if the control could not be added.

Related Information

- “ldap_insert_control()—Insert a Control in the List of LDAP Server Controls” on page 108 — Insert a control in the list of LDAP server controls.
- “ldap_remove_control()—Remove a Control from the List of LDAP Server Controls” on page 151 — Remove a control from the list of LDAP server controls.
- “ldap_copy_controls()—Make a Copy of the List of LDAP Server Controls” on page 34 — Synchronously add an entry with controls.

API introduced: V5R3

ldap_add_ext()—Perform an LDAP Add Operation with Controls

Syntax

#include <ldap.h>

int ldap_add_ext(
   LDAP *ld,
    const char *dn,
    LDAPMod **attrs,
    LDAPControl **serverctrls,
    LDAPControl **clientctrls,
    int *msgidp)

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

The ldap_add_ext() function is used to perform an LDAP add operation with controls.

Authorities and Locks

No i5/OS authority is required. All authority checking is done by the LDAP server.
Parameters

\( ld \) (Input) The LDAP pointer returned by a previous call to \( \text{ldap_init()} \)—Perform an LDAP Initialization Operation” on page 105, \( \text{ldap_ssl_init} \)—Initialize an SSL Connection.” on page 204, or \( \text{ldap_open()} \)—Perform an LDAP Open Operation” on page 135.

\( dn \) (Input) The DN of the entry to add.

\( \text{attrs} \) (Input) The entry’s attributes, specified using the LDAPMod structure, as defined for \( \text{ldap_modify()} \)—Perform an LDAP Modify Entry Request” on page 111. The mod_type and mod_vals fields should be filled in. The mod_op field is ignored unless ORed with the constant LDAP_MOD_BVALUES. In this case, the mod_op field is used to select the mod_bvalues case of the mod_vals union.

\( \text{serverctrls} \) (Input) A list of LDAP server controls. This parameter may be set to null. See LDAP Controls for more information about server controls.

\( \text{clientctrls} \) (Input) A list of LDAP client controls. This parameter may be set to null. See LDAP Controls for more information about client controls.

\( \text{msgidp} \) (Output) This result parameter is set to the message ID of the request if the \( \text{ldap_add_ext()} \) call succeeds.

Return Value

\text{LDAP_SUCCESS}

if the request was successful. If successful, \( \text{ldap_add_ext()} \) places the message ID of the request in *msgid. A subsequent call to \( \text{ldap_result()} \)—Retrieve Result of an Asynchronous LDAP Operation” on page 156 can be used to obtain the result of the operation.

another LDAP error code

if the request was not successful.

Error Conditions

If \( \text{ldap_add_ext()} \) is not successful, an LDAP error code will be returned. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values. The error code indicates if the operation completed successfully. The \( \text{ldap_parse_result()} \)—Extract Information from Results” on page 144 API is used to check the error code in the result.

Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_add_ext API.</td>
</tr>
</tbody>
</table>

Related Information

- “\text{ldap_add()} \)—Perform an LDAP Add Operation” on page 9—Asynchronously add an entry.
- “\text{ldap_add_s()} \)—Perform an LDAP Add Operation (Synchronous)” on page 14—Synchronously add an entry.
- “\text{ldap_add_ext_s()} \)—Perform an LDAP Add Operation with Controls (Synchronous)” on page 13—Synchronously add an entry with controls.
ldap_modify_ext()—Perform an LDAP Modify Entry Request with Controls

Asynchronously modify an entry with controls.

The ldap_add_ext() API supports LDAP V3 server controls and client controls.

API introduced: V4R5

ldap_add_ext_s()—Perform an LDAP Add Operation with Controls (Synchronous)

Syntax

```
#include <ldap.h>

int ldap_add_ext_s(LDAP *ld, const char *dn, LDAPMod **attr, LDAPControl **serverctrls, LDAPControl **clientctrls)
```

Library Name/Service Program: QSYS/QGLDCLNT

Default Public Authority: *USE

Threadsafe: Yes

The ldap_add_ext_s() function is used to perform synchronous LDAP add operation with controls.

Authorities and Locks

No i5/OS authority is required. All authority checking is done by the LDAP server.

Parameters

- **ld** (Input) The LDAP pointer returned by a previous call to ldap_init()—Perform an LDAP Initialization Operation or ldap_open()—Perform an LDAP Open Operation.
- **dn** (Input) The DN of the entry to add.
- **attr** (Input) The entry’s attributes, specified using the LDAPMod structure, as defined for ldap_modify()—Perform an LDAP Modify Entry Request. The mod_type and mod_vals fields should be filled in. The mod_op field is ignored unless ORed with the constant LDAP_MOD_BVALUES. In this case, the mod_op field is used to select the mod_bvalues case of the mod_vals union.
- **serverctrls** (Input) A list of LDAP server controls. This parameter may be set to null. See LDAP Controls for more information about server controls.
- **clientctrls** (Input) A list of LDAP client controls. This parameter may be set to null. See LDAP Controls for more information about client controls.

Return Value

- LDAP_SUCCESS if the request was successful.
another LDAP error code  
if the request was not successful.

Error Conditions
The `ldap_add_ext_s()` will return an LDAP error code if not successful. See [“LDAP Client API Error Conditions” on page 300](#) for possible LDAP error code values.

Error Messages
The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of <code>ldap_add_ext_s</code> API.</td>
</tr>
</tbody>
</table>

Related Information
- [“ldap_add()—Perform an LDAP Add Operation” on page 9](#) — Asynchronously add an entry.
- [“ldap_add_s()—Perform an LDAP Add Operation (Synchronous)”](#) — Synchronously add an entry.
- [“ldap_add_ext()—Perform an LDAP Add Operation with Controls” on page 11](#) — Asynchronously add an entry with controls.
- [“ldap_modify_ext_s()—Perform an LDAP Modify Entry Request with Controls” on page 116](#) — Synchronously modify an entry with controls.

The `ldap_add_ext_s()` API supports LDAP V3 server controls and client controls.

API introduced: V4R5

**Idap_add_s()—Perform an LDAP Add Operation (Synchronous)**

Syntax

```c
#include <ldap.h>

int ldap_add_s(  
    LDAP *ld,  
    const char *dn,  
    LDAPMod *attrs)
```

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

The `ldap_add_s()` function is used to perform synchronous LDAP add operation.

Authorities and Locks
No i5/OS authority is required. All authority checking is done by the LDAP server.

Parameters

- `Id` (Input) The LDAP pointer returned by a previous call to [“ldap_init()—Perform an LDAP Initialization Operation” on page 105](#) [“ldap_ssl_init—Initialize an SSL Connection.” on page 204](#) or [“ldap_open()—Perform an LDAP Open Operation” on page 135](#)
dn  (Input) The DN of the entry to add.

attrs (Input) The entry’s attributes, specified using the LDAPMod structure, as defined for
"ldap_modify()—Perform an LDAP Modify Entry Request” on page 111. The mod_type and
mod_vals fields should be filled in. The mod_op field is ignored unless ORed with the constant
LDAP_MOD_BVALUES. In this case, the mod_op field is used to select the mod_bvalues case of
the mod_vals union.

Return Value

LDAP_SUCCESS
if the request was successfully sent.

another LDAP error code
if the request was not successfully sent.

Error Conditions
If ldap_add_s() is not successful, ld_errno will be set to indicate the error. See “LDAP Client API Error
Conditions” on page 300 for possible LDAP error code values.

Error Messages
The following message may be sent from this function.

Message ID   Error Message Text
CPF3CF2E   Error(s) occurred during running of ldap_add_s API.

Related Information

• "ldap_add()—Perform an LDAP Add Operation” on page 9 — Asynchronously add an entry.
• "ldap_add_ext_s()—Perform an LDAP Add Operation with Controls (Synchronous)” on page 13 —
Synchronously add an entry with controls.
• "ldap_add_ext()—Perform an LDAP Add Operation with Controls” on page 11 — Asynchronously add
an entry with controls.
• "ldap_modify_s()—Perform an LDAP Modify Entry Request (Synchronous)” on page 118 —
Synchronously modify an entry.

API introduced: V4R3

Idap_app_ssl_client_init_np()—Initialize the LDAP Client for a Secure
Connection using DCM

Syntax
#include <ldap.h>
#include <ldapssl.h>

int ldap_app_ssl_client_init_np(
   char *dcm_identifier, 
   int *pSSLReasonCode)

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes
The `ldap_app_ssl_client_init_np()` is an LDAP V3 function used to initialize the LDAP client using the Digital Certificate Manager (DCM) to control the digital certificate in preparation for making a secure connection (using Secure Sockets Layer (SSL)) to a LDAP server.

`ldap_app_ssl_client_init_np()` must be called prior to `ldap_app_ssl_init_np — Initialize an SSL Connection` on page 18 to establish a connection, and prior to any kind of `ldap_bind()`, whether it be an "ldap_sasl_bind_s)—Perform an LDAP SASL Bind Request (Synchronous)" on page 161 or an "ldap_simple_bind_s)—Perform a Simple LDAP Bind Request (Synchronous)" on page 198. `ldap_app_ssl_client_init_np()` must be called only once per job, while multiple `ldap_app_ssl_client_init_np()` or secure connections can be done, allowing one (DCM) initialization to be done for many connections. Once the secure connection is established all subsequent LDAP messages that flow over the secure connection are encrypted, including the `ldap_bind()` parameters, until `ldap_unbind()—Perform an LDAP Unbind Request` on page 209 is called.

Either "ldap_ssl_client_init — Initialize the SSL Library.” on page 199 or `ldap_app_ssl_client_init_np()` (but not both) can be called in an application process. If you are not going to use SSL client authentication (LDAP SASL bind with the EXTERNAL mechanism), use "ldap_ssl_client_init — Initialize the SSL Library.” on page 199.

**Authorities and Locks**

*R authority is needed to the selected Certificate Store and *X to the associated directories.

**Parameters**

- **dcm_identifier**
  
  (Input) An identifier string that corresponds to a secure application registered with DCM. If NULL is used, then the default Directory Services client application ID will be used (QIBM_GLD_DIRSRV_CLIENT).

- **pSSLReasonCode**
  
  (Output) A pointer to the SSL Reason Code, which provides additional information in the event that an error occurs during initialization of the SSL stack (when `ldap_app_ssl_client_init_np()` is called). See QSYSINC/H.LDAPSSL for reason codes that can be returned.

**Examples**

See [Code disclaimer information](#) for information pertaining to code examples.

The following scenario depicts the recommended calling sequence where the entire set of LDAP transactions are “protected” by using a secure SSL connection:

```c
rc = ldap_app_ssl_client_init_np (dcm_identifier, &reasoncode);
ld = ldap_app_ssl_init_np(ldaphost, ldapport);
rc = ldap_set_option( ld, LDAP_OPT_SSL_CIPHER, &ciphers);
   rc = ldap_sasl_bind_s( ld, NULL, LDAP_MECHANISM_EXTERNAL, NULL, NULL, NULL );

...additional LDAP API calls
rc = ldap_unbind( ld );
```

The following scenario depicts the calling sequence for multiple connections using one DCM identifier:

```c
rc = ldap_app_ssl_client_init_np (dcm_identifier, &reasoncode);
ld = ldap_app_ssl_init_np(ldaphost, ldapport);
rc = ldap_set_option( ld, LDAP_OPT_SSL_CIPHER, &ciphers);
   rc = ldap_sasl_bind_s( ld, NULL, LDAP_MECHANISM_EXTERNAL, NULL, NULL, NULL );

/* For multiple secure connections using the same dcm_identifier. */
ld1 = ldap_app_ssl_init_np(ldaphost, ldapport);
   rc = ldap_sasl_bind_s( ld, NULL, LDAP_MECHANISM_EXTERNAL, NULL, NULL, NULL );
```
ld2 = ldap_app_ssl_init_np(ldaphost, ldapport);
   rc = ldap_sasl_bind_s(ld, NULL, LDAP_MECHANISM_EXTERNAL, NULL, NULL, NULL);

...additional LDAP API calls
rc = ldap_unbind(ld);
rc = ldap_unbind(ld1);
rc = ldap_unbind(ld2);

Return Value
LDAP_SUCCESS
   if the request was successful.
another LDAP error code
   if the request was not successful.

Error Conditions
If ldap_app_ssl_client_init_np() is not successful it will return an LDAP error code. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values.

Error Messages
The following message may be sent from this function.

   Message ID   Error Message Text
CPF3CF2 E     Error(s) occurred during running of ldap_ssl_client_init_np API.

Related Information
• “ldap_app_ssl_init_np —Initialize an SSL Connection” on page 18 — Initializes an SSL Connection.
• “ldap_app_ssl_start_np()—Start a Secure LDAP Connection using DCM” on page 20 — Start a Secure LDAP Connection using DCM.
• “ldap_ssl_client_init —Initialize the SSL Library.” on page 199 — Initializes the SSL Library.
• “ldap_ssl_init —Initialize an SSL Connection.” on page 204 — Initializes an SSL connection.
• “ldap_ssl_start()—Start a Secure LDAP Connection” on page 207 — Creates a secure SSL connection (deprecated).
• “ldap_bind()—Perform an LDAP Bind Request” on page 23 — Bind to the directory server.
• “ldap_sasl_bind_s()—Perform an LDAP SASL Bind Request (Synchronous)” on page 161 — Synchronously bind to the directory using Simple Authentication Security Layer (SASL).
• “ldap_unbind()—Perform an LDAP Unbind Request” on page 209 — unbind from the LDAP server and close the connection.
• “ldap_ssl_environment_init()—Initialize SSL for a secure connection” on page 202 — Initializes SSL for a secure connection between client and server.
• “ldap_start_tls_s_np() — Starts a TLS session with a Certificate”
• “ldap_start_tls_app_np() — Starts a TLS session with an Application ID”
• “ldap_stop_tls_s_np()—End a TLS Session” on page 209 — Ends a TLS session.

API introduced: V5R1
**ldap_app_ssl_init_np — Initialize an SSL Connection**

Syntax
```c
#include <ldap.h>
#include <ldapssl.h>
LDAP *ldap_app_ssl_init_np(
    char *host,
    int port)
```

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

The `ldap_app_ssl_init_np()` routine is used to initialize a secure SSL session with a server. Note that the server is not actually contacted until an operation is performed that requires it, allowing various options to be set after initialization. Once the secure connection is established, all subsequent LDAP messages that flow over the secure connection are encrypted, including the "ldap_simple_bind()—Perform a Simple LDAP Bind Request" on page 196 parameters, until "ldap_unbind()—Perform an LDAP Unbind Request" on page 209 is called.

Note that when connecting to an LDAP V2 server, one of the "ldap_simple_bind()—Perform a Simple LDAP Bind Request" on page 196 or "ldap_bind()—Perform an LDAP Bind Request" on page 23 calls must be completed before other operations can be performed on the session (with the exception of "ldap_set_option() — Set LDAP Options" on page 189/"ldap_get_option()—Retrieve LDAP Options" on page 96). The LDAP V3 protocol does not require a bind operation before performing other operations.

The ciphers for the encryption of the connection are based on the current Crypto Access Provider licensed program loaded. See "ldap_get_option()—Retrieve LDAP Options" on page 96 or "ldap_set_option() — Set LDAP Options" on page 189 for more information on setting the ciphers to be used.

**Authorities and Locks**

*R authority is needed to the selected Certificate Store and *X to the associated directories.

**Parameters**

*host (Input) Several methods are supported for specifying one or more target LDAP servers, including the following:

<table>
<thead>
<tr>
<th>Explicit Host List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies the name of the host on which the LDAP server is running. The host parameter may contain a blank-separated list of hosts to try to connect to, and each host may optionally be of the form host:port. If present, the :port overrides the port parameter.</td>
</tr>
</tbody>
</table>

The following are typical examples: ld=ldap_app_ssl_init_np ("server1", ldaps_port);
ld=ldap_app_ssl_init_np ("server2:1200", ldaps_port);
ld=ldap_app_ssl_init_np ("server1:800 server2:2000 server3", ldaps_port);

<table>
<thead>
<tr>
<th>Localhost</th>
</tr>
</thead>
<tbody>
<tr>
<td>If the host parameter is null, it is assumed that the LDAP server is running on the local host.</td>
</tr>
</tbody>
</table>
Default Hosts

If the host parameter is set to “ldaps://” the LDAP library will attempt to locate one or more default LDAP servers, with SSL ports, using the SecureWay “ldap_server_locate()”—Locate Suitable LDAP Servers” on page 175 function. The port specified on the call is ignored, since ldap_server_locate() returns the port.

For example, the following two are equivalent: ld=ldap_app_ssl_init_np (“ldaps://”, ldaps_port); ld=ldap_app_ssl_init_np (LDAPS_URL_PREFIX, LDAPS_PORT);

If more than one default server is located, the list is processed in sequence, until an active server is found.

The LDAP URL can include a Distinguished Name, used as a filter for selecting candidate LDAP servers based on the server’s suffix (or suffixes). If the most significant portion of the DN is an exact match with a server’s suffix (after normalizing for case), the server is added to the list of candidate servers. For example, the following will only return default LDAP servers that have a suffix that supports the specified DN: ld=ldap_app_ssl_init_np (“ldaps://cn=fred, dc=austin, dc=ibm, dc=com”, LDAPS_PORT)

In this case, a server that has a suffix of “dc=austin, dc=ibm, dc=com” would match. If more than one default server is located, the list is processed in sequence, until an active server is found.

If the LDAP URL contains a host name and optional port, the host is used to create the connection. No attempt is made to locate the default server(s), and the DN, if present, is ignored.

For example, the following two are equivalent: ld=ldap_app_ssl_init_np (“ldaps://myserver”, LDAPS_PORT);
ld=ldap_app_ssl_init_np (“myserver”, LDAPS_PORT);

Local Socket

If the host parameter is prefixed with “/”, the host parameter is assumed to be the name of a UNIX socket (that is, socket family is AF_UNIX) and port is ignored. This will fail for ldap_app_ssl_init_np() because UNIX sockets do not support SSL, nor is it necessary since data will not be flowing over the network.

Host with Privileged Port

If a specified host is prefixed with “privport://”, then the LDAP library will use the resvport() function to attempt to obtain one of the reserved ports (512 through 1023), instead of an “ephemeral” port. The search for a reserved port starts at 1023 and stops at 512. If a reserved port cannot be obtained, ldap_app_ssl_init_np() will fail.

For example: ld=ldap_app_ssl_init_np (“privport://server1, ldaps_port”);
ld=ldap_app_ssl_init_np (“privport://server2:1200”, ldaps_port);
ld=ldap_app_ssl_init_np (“privport://server1:800 server2:2000 privport://server3”, ldaps_port);

<table>
<thead>
<tr>
<th>port</th>
<th>(Input) The port number to which to connect. If the default IANA-assigned SSL port of 636 is desired, LDAPS_PORT should be specified. The value specified for this parameter is ignored in some situations; see the description for the host parameter.</th>
</tr>
</thead>
</table>

**Return Value**

**Session Handle**

if the request was successful. The Session Handle returned by ldap_app_ssl_init_np() is a pointer to an opaque data type representing an LDAP session. The “ldap_get_option()”—Retrieve LDAP Options” on page 96 and “ldap_set_option() — Set LDAP Options” on page 189 APIs are used to access and set a variety of session-wide parameters; see these APIs for more information.

NULL if the request was not successful.

**Error Conditions**

ldap_app_ssl_init_np0 will return NULL if not successful.
Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_app_ssl_init_np API.</td>
</tr>
</tbody>
</table>

Related Information

- “ldap_app_ssl_client_init_np()—Initialize the LDAP Client for a Secure Connection using DCM” on page 15 — Initializes the Client for a Secure LDAP Connection using DCM.
- “ldap_ssl_client_init —Initialize the SSL Library.” on page 199 — Initializes the SSL library.
- “ldap_ssl_environment_init()—Initialize SSL for a secure connection” on page 202 — Initializes SSL for a secure connection between client and server.
- “ldap_start_tls_s_np()—Starts a TLS session with a Certificate.”
- “ldap_start_tls_app_np()—Starts a TLS session with an Application ID.”
- “ldap_stop_tls_s_np()—Ends a TLS session” on page 209 — Ends a TLS session.
- “ldap_app_ssl_start_np()—Start a Secure LDAP Connection using DCM” — Creates a secure SSL connection (deprecated).
- “ldap_ssl_start()—Start a Secure LDAP Connection” on page 207 — Creates a secure SSL connection (deprecated).

Example

See Code disclaimer information for information pertaining to code examples.

The following scenario depicts the recommended calling sequence where the entire set of LDAP transactions are protected by using a secure SSL connection:

```c
rc = ldap_app_ssl_client_init_np (dcm_identifier, &reasoncode);
ld = ldap_app_ssl_init_np(ldaphost, ldapport);
rc = ldap_set_option (ld, LDAP_OPT_SSL_CIPHER, &ciphers);
    rc = ldap_sasl_bind_s (ld, NULL, LDAP_MECHANISM_EXTERNAL, NULL, NULL, NULL);
...additional LDAP API calls
rc = ldap_unbind (ld);
```

API introduced: V5R1

Top | “Lightweight Directory Access Protocol (LDAP) APIs,” on page 1 | APIs by category

**ldap_app_ssl_start_np()—Start a Secure LDAP Connection using DCM**

Syntax

```c
#include <ldap.h>
#include <ldapssl.h>

int ldap_app_ssl_start_np(LDAP *ld,
                         char *dcm_identifier)
```

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

This is a deprecated API.
The `ldap_app_ssl_start_np()` function is used to start a secure connection (using Secure Sockets Layer (SSL)) to an LDAP server using the Digital Certificate Manager (DCM) to control the digital certificate.

`ldap_app_ssl_start_np()` must be called after `ldap_open()` and prior to `ldap_bind()`. Once the secure connection is established for the `ld`, all subsequent LDAP messages that flow over the secure connection are encrypted, including the `ldap_bind()` parameters, until `ldap_unbind()` is called.

**Authorities and Locks**

*R authority is needed to the selected Certificate Store and *X to the associated directories.

**Parameters**

- `ld` *(Input)* The LDAP pointer returned by a previous call to `ldap_init()—Perform an LDAP Initialization Operation” on page 105, “ldap_ssl_init —Initialize an SSL Connection,” on page 204, or “ldap_open()—Perform an LDAP Open Operation” on page 135.*

- `dcm_identifier` *(Input)* An identifier string that corresponds to a secure application registered with DCM. The use of NULL assumes that in a prior use of the this API a valid DCM identifier for an application has been used and that it is to be used again for this connection. This allows multiple connections without going through the initialization of SSL with a DCM identifier more than once.

**Return Value**

- LDAP_SUCCESS if the request was successful.
- another LDAP error code if the request was not successful.

**Error Conditions**

`ldap_app_ssl_start_np()` will return an LDAP error code if not successful. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values. Depending on the error code, `errno` information also may be available.

**Error Messages**

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_app_ssl_start_np API.</td>
</tr>
</tbody>
</table>

**Related Information**

- “ldap_app_ssl_client_init_np()—Initialize the LDAP Client for a Secure Connection using DCM” on page 15 — Initialize the Client for a Secure LDAP Connection using DCM
- “ldap_ssl_client_init —Initialize the SSL Library.” on page 199 — Initializes the SSL Library
- “ldap_ssl_init —Initialize an SSL Connection.” on page 204 — Initializes an SSL connection
- “ldap_ssl_start()—Start a Secure LDAP Connection” on page 207 — Creates a secure SSL connection
- “ldap_bind()—Perform an LDAP Bind Request” on page 23 — Bind to the directory server
- “ldap_unbind()—Perform an LDAP Unbind Request” on page 209 — Unbind from the LDAP server and close the connection
- “ldap_open()—Perform an LDAP Open Operation” on page 135 — Open a connection to an LDAP server
ldap_ssl_environment_init()—Initialize SSL for a secure connection

ldap_start_tls_s_np()—Starts a TLS session with a Certificate

ldap_start_tls_app_np()—Starts a TLS session with an Application ID

ldap_stop_tls_s_np()—End a TLS Session

Example

See Code disclaimer information for information pertaining to code examples.

The following scenario depicts the recommended calling sequence where the entire set of LDAP transactions are “protected” by using a secure SSL connection, including the dn and password that flow on the ldap_simple_bind():

```c
ld = ldap_open(ldaphost, ldapport);
rc = ldap_app_ssl_start_np(ld, dcm_identifier);
rc = ldap_simple_bind_s(ld, binddn, passwd);
...
rc = ldap_unbind(ld);
```

The following scenario depicts the calling sequence for multiple connections using one DCM identifier:

```c
ld = ldap_open(ldaphost, ldapport);
rc = ldap_app_ssl_start_np(ld, dcm_identifier);
rc = ldap_simple_bind_s(ld, binddn, passwd);

/* For multiple secure connections using the same dcm_identifier. */

ld1 = ldap_open(ldaphost, ldapport);
    rc = ldap_app_ssl_start_np(ld1, NULL);
    rc = ldap_simple_bind_s(ld1, binddn, passwd);

ld2 = ldap_open(ldaphost, ldapport);
    rc = ldap_app_ssl_start_np(ld2, NULL);
    rc = ldap_simple_bind_s(ld2, binddn, passwd);

...
rc = ldap_unbind(ld);
rc = ldap_unbind(ld1);
rc = ldap_unbind(ld2);
```

API introduced: V4R4
Authorities and Locks
No i5/OS authority is required.

Parameters
berptr  (Input) The address of the BerElement to be freed, as returned from ldap_first_attribute() and ldap_next_attribute().

Return Value
None.

Error Conditions
The ldap_ber_free() API does not return an error code.

Error Messages
The following message may be sent from this function.

```
Message ID   Error Message Text
CPF3CF2 E    Error(s) occurred during running of ldap_ber_free API.
```

Related Information
- "ldap_first_attribute()—Retrieve First Attribute in an Entry” on page 79 — Retrieve First Attribute in an Entry
- "ldap_next_attribute()—Retrieve Next Attribute in an Entry” on page 129 — Retrieve Next Attribute in an Entry

API introduced: V4R5

ldap_bind()—Perform an LDAP Bind Request

```
#include <ldap.h>

int ldap_bind( 
    LDAP *ld, 
    const char *dn, 
    const char *cred, 
    int method) 
```

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

The ldap_bind() function provides general authentication routines, where in principle an authentication method can be chosen. In this toolkit, method must be set to LDAP_AUTH_SIMPLE.

The ldap_bind() function is used to authenticate a distinguished name (DN) to a directory server. When connecting to an LDAP V2 server, after a connection is made by using the ldap_open() API, an LDAP bind API must be called before any other LDAP APIs can be called for that connection. Binding the connection is not required for LDAP V3.
**ldap_bind()** is an asynchronous request. The result of the operation can be obtained by a subsequent call to **ldap_result()**.

Since this API is deprecated, **ldap_simple_bind()**—Perform a Simple LDAP Bind Request should be used instead.

**Authorities and Locks**
No i5/OS authority is required. All authority checking is done by the LDAP server.

**Parameters**
- **ld** *(Input)* The LDAP pointer returned by a previous call to **ldap_init()**—Perform an LDAP Initialization Operation, **ldap_ssl_init**—Initialize an SSL Connection, or **ldap_open()**—Perform an LDAP Open Operation.
- **dn** *(Input)* The distinguished name of the entry to bind as.
- **cred** *(Input)* The credentials with which to authenticate. Arbitrary credentials can be passed using this parameter. In most cases, this is the user’s password.
- **method** *(Input)* Selects the authentication method to use. Specify LDAP_AUTH_SIMPLE for simple authentication. Simple authentication is the only supported method.

Note that use of the **ldap_bind()** API is deprecated.

**Return Value**
- **Message ID of the Initiated Request**
  - if the **ldap_bind()** was successful.
  - **-1** if the request was not successful.

**Error Conditions**
If **ldap_bind()** is not successful, **ld_errno** will be set to indicate the error. See **LDAP Client API Error Conditions** on page 300 for possible LDAP error code values. Use **ldap_get_errno()**—Retrieve Error Information function to retrieve the error information.

**Error Messages**
The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_bind API.</td>
</tr>
</tbody>
</table>

**Related Information**
- **ldap_bind_s()**—Perform an LDAP Bind Request (Synchronous)—Synchronously bind to the directory (deprecated).
- **ldap_sasl_bind()**—Perform an LDAP SASL Bind Request—Asynchronously bind to the directory using SASL.
- **ldap_sasl_bind_s()**—Perform an LDAP SASL Bind Request (Synchronous)—Synchronously bind to the directory using SASL.
- **ldap_simple_bind()**—Perform a Simple LDAP Bind Request—Asynchronously bind to the directory using simple authentication.
• “ldap_simple_bind_s()—Perform a Simple LDAP Bind Request (Synchronous)” on page 198—Synchronously bind to the directory using simple authentication.

• “ldap_unbind()—Perform an LDAP Unbind Request” on page 209—Asynchronously unbind from the LDAP server and close the connection.

• “ldap_unbind_s()—Perform an LDAP Unbind Request (Synchronous)” on page 212—Synchronously unbind from the LDAP server and close the connection.

• “ldap_set_rebind_proc()—Set Rebind Procedure” on page 195—Sets the entry-point of a routine during the chasing of referrals.

API introduced: V4R3

Top | “Lightweight Directory Access Protocol (LDAP) APIs,” on page 1 | APIs by category

### ldap_bind_s()—Perform an LDAP Bind Request (Synchronous)

Syntax
```
#include <ldap.h>

int ldap_bind_s(
    LDAP* ld,
    const char* dn,
    const char* cred,
    int method)
```

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

The `ldap_bind_s()` function provide synchronous general authentication routines, where in principle an authentication method can be chosen. In this toolkit, method must be set to LDAP_AUTH_SIMPLE.

The `ldap_bind_s()` function is used to authenticate a distinguished name (DN) to a directory server. When connecting to an LDAP V2 server, after a connection is made by using the `ldap_open()` API, an LDAP bind API must be called before any other LDAP APIs can be called for that connection. Binding the connection is not required for LDAP V3.

`ldap_bind_s()` is synchronous request.

Since this APIs is deprecated, “ldap_simple_bind_s()—Perform a Simple LDAP Bind Request (Synchronous)” on page 198 should be used instead.

### Authorities and Locks

No i5/OS authority is required. All authority checking is done by the LDAP server.

### Parameters

**ld** (Input) The LDAP pointer returned by a previous call to “`ldap_init()—Perform an LDAP Initialization Operation” on page 105,”`ldap_ssl_init—Initialize an SSL Connection.” on page 204, or “`ldap_open()—Perform an LDAP Open Operation” on page 135”

**dn** (Input) The distinguished name of the entry to bind as.

**cred** (Input) The credentials with which to authenticate. Arbitrary credentials can be passed using this parameter. In most cases, this is the user’s password.
method
(Input) Selects the authentication method to use. Specify LDAP_AUTH_SIMPLE for simple authentication. Simple authentication is the only supported method.

Note that use of the ldap_bind_s() APIs is deprecated.

Return Value

LDAP_SUCCESS
if the request was successful.

another LDAP error code
if the request was not successful.

Error Conditions

The ldap_bind_s() API will return an LDAP error code if not successful. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values.

Error Messages

The following message may be sent from this function.

Message ID      Error Message Text
CPF3CF2 E        Error(s) occurred during running of ldap_bind_s API.

Related Information

- “ldap_bind()—Perform an LDAP Bind Request” on page 23 — Asynchronously bind to the directory (deprecated).
- “ldap_sasl_bind()—Perform an LDAP SASL Bind Request” on page 159 — Asynchronously bind to the directory using SASL.
- “ldap_sasl_bind_s()—Perform an LDAP SASL Bind Request (Synchronous)” on page 161 — Synchronously bind to the directory using SASL.
- “ldap_simple_bind()—Perform a Simple LDAP Bind Request” on page 196 — Asynchronously bind to the directory using simple authentication.
- “ldap_simple_bind_s()—Perform a Simple LDAP Bind Request (Synchronous)” on page 198 — Synchronously bind to the directory using simple authentication.
- “ldap_unbind()—Perform an LDAP Unbind Request” on page 209 — Asynchronously unbind from the LDAP server and close the connection.
- “ldap_unbind_s()—Perform an LDAP Unbind Request (Synchronous)” on page 212 — Synchronously unbind from the LDAP server and close the connection.
- “ldap_set_rebind_proc()—Set Rebind Procedure” on page 195 — Sets the entry-point of a routine during the chasing of referrals.

API introduced: V4R3

Top | “Lightweight Directory Access Protocol (LDAP) APIs,” on page 1 | APIs by category

ldap_compare()—Perform an LDAP Compare Operation

Syntax
#include <ldap.h>

int ldap_compare(
    LDAP *ld,
    const char *dn,
    const char *attr,
    const char *value)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The `ldap_compare()` function is used to perform an LDAP compare operation. The API uses as input the distinguished name (DN) of the entry on which to perform the compare, and uses an `attr` and `value` (the attribute type and value to compare to those found in the entry).

Binary values are not supported by this API. Use "ldap_compare_ext()—Perform an LDAP Compare Operation with Controls" on page 28 if binary values must be compared.

`ldap_compare()` is an asynchronous request. The result of the operation can be obtained by a subsequent call to "ldap_result()—Retrieve Result of an Asynchronous LDAP Operation" on page 156.

**Authorities and Locks**
No i5/OS authority is required. All authority checking is done by the LDAP server.

**Parameters**

- `ld`  (Input) The LDAP pointer returned by a previous call to "ldap_init()—Perform an LDAP Initialization Operation" on page 105, "ldap_ssl_init — Initialize an SSL Connection." on page 204, or "ldap_open()—Perform an LDAP Open Operation" on page 135.

- `dn`  (Input) The DN of the entry upon which to perform the compare.

- `attr` (Input) The attribute type to use in the comparison.

- `value` (Input) The string attribute value to compare against the value in the entry.

**Return Value**

- Message ID of the Operation Initiated
  - if the request was successful.
  - -1 if the request was not successful.

**Error Conditions**

If `ldap_compare()` is not successful, `ld_errno` will be set to indicate the error. See "LDAP Client API Error Conditions" on page 300 for possible LDAP error code values. Use "ldap_get_errno()—Retrieve Error Information" on page 91 function to retrieve the error information.

**Error Messages**

The following message may be sent from this function.

- Message ID  Error Message Text
  - CPF3CF2 E  Error(s) occurred during running of ldap_compare API.
Related Information

- "ldap_compare_s()—Perform an LDAP Compare Operation (Synchronous)" on page 31 — Synchronous compare to a directory entry.
- "ldap_compare_ext()—Perform an LDAP Compare Operation with Controls" — Asynchronous compare to a directory entry with controls.
- "ldap_compare_ext_s()—Perform an LDAP Compare Operation with Controls (Synchronous)" on page 30 — Synchronous compare to a directory entry with controls.

API introduced: V4R3

**ldap_compare_ext()—Perform an LDAP Compare Operation with Controls**

Syntax

```c
#include <ldap.h>

struct berval {
    unsigned long bv_len;
    char *bv_val;
};

int ldap_compare_ext(
    LDAP *ld,
    const char *dn,
    const char *attr,
    const berval *bvalue,
    LDAPControl **serverctrls,
    LDAPControl **clientctrls,
    int *msgidp)
```

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threading: Yes

The `ldap_compare_ext()` function is used to perform an LDAP compare operation with controls. The `ldap_compare_ext()` API initiates an asynchronous compare operation and returns the constant `LDAP_SUCCESS` if the request was successfully sent, or another LDAP error code if not.

**Authorities and Locks**

No i5/OS authority is required.

**Parameters**

- `ld` (Input) The LDAP pointer returned by a previous call to `ldap_init()—Perform an LDAP Initialization Operation” on page 105, `ldap_ssl_init —Initialize an SSL Connection.” on page 204, or `ldap_open()—Perform an LDAP Open Operation” on page 135.
- `dn` (Input) The distinguished name (DN) of the entry upon which to perform the compare.
- `attr` (Input) The attribute type to use in the comparison.
- `bvalue` (Input) The attribute value to compare against the value in the entry. This is a pointer to a struct `berval`, making it possible to compare binary values.
serverctrls
(Input) Specifies a list of LDAP server controls. This parameter may be set to null. See [LDAP Controls](#) for more information about server controls.

clientctrls
(Input) A list of LDAP client controls. This parameter may be set to null. See [LDAP Controls](#) for more information about client controls.

msgidp
(Output) This result parameter is set to the message ID of the request if the `ldap_compare_ext()` call succeeds.

**Return Value**

**LDAP_SUCCESS**
if the request was successfully sent. If successful, `ldap_compare_ext()` places the message ID of the request in `msgidp`. A subsequent call to [“ldap_result()—Retrieve Result of an Asynchronous LDAP Operation”](#) can be used to obtain the result of the operation. Once the operation has completed, [“ldap_result()—Retrieve Result of an Asynchronous LDAP Operation”](#) returns a result that contains the status of the operation in the form of an error code. The error code indicates if the operation completed successful ([LDAP_COMPARE_TRUE](#) or [LDAP_COMPARE_FALSE](#)) or another LDAP error code if the request was not successful.

**Error Conditions**
The `ldap_compare_ext()` API will return an LDAP error code if not successful. See [“LDAP Client API Error Conditions”](#) for possible LDAP error code values.

**Error Messages**
The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_compare_ext API</td>
</tr>
</tbody>
</table>

**Related Information**

- [“ldap_compare()—Perform an LDAP Compare Operation”](#) — Asynchronous compare to a directory entry.
- [“ldap_compare_s()—Perform an LDAP Compare Operation (Synchronous)”](#) — Synchronous compare to a directory entry.
- [“ldap_compare_ext_s()—Perform an LDAP Compare Operation with Controls (Synchronous)”](#) — Synchronous compare to a directory entry with controls.

The `ldap_compare_ext()` API supports LDAP V3 server controls and client controls.

API introduced: V4R5
ldap_compare_ext_s()—Perform an LDAP Compare Operation with Controls (Synchronous)

Syntax
#include <ldap.h>

struct berval {
    unsigned long bv_len;
    char *bv_val;
};

int ldap_compare_ext_s(
    LDAP *ld,
    const char *dn,
    const char *attr,
    const berval *bvalue,
    LDAPControl **serverctrls,
    LDAPControl **clientctrls
)

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

The ldap_compare_ext_s() function is used to perform a synchronous LDAP compare operation with controls.

Authorities and Locks
No i5/OS authority is required. All authority checking is done by the LDAP server.

Parameters

Id  (Input) The LDAP pointer returned by a previous call to “ldap_init()—Perform an LDAP Initialization Operation” on page 105, “ldap_ssl_init—Initialize an SSL Connection.” on page 204, or “ldap_open()—Perform an LDAP Open Operation” on page 135.

dn  (Input) The distinguished name (DN) of the entry upon which to perform the compare.

attr (Input) The attribute type to use in the comparison.

bvalue  (Input) The attribute value to compare against the value in the entry. This is a pointer to a struct berval, making it possible to compare binary values.

serverctrls  (Input) A list of LDAP server controls. This parameter may be set to null. See LDAP Controls for more information about server controls.

clientctrls  (Input) A list of LDAP client controls. This parameter may be set to null. See LDAP Controls for more information about client controls.

Return Value

LDAP_COMPARE_TRUE
    if the entry contains the attribute value.

LDAP_COMPARE_FALSE
    if the entry does not contain the attribute value.

another LDAP error code
    if the request was not successful.
Error Conditions
The `ldap_compare_ext_s()` API will return an LDAP error code if not successful. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values.

Error Messages
The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_compare_ext_s API.</td>
</tr>
</tbody>
</table>

Related Information
- **`ldap_compare()`—Perform an LDAP Compare Operation** on page 26 — Asynchronous compare to a directory entry.
- **`ldap_compare_s()`—Perform an LDAP Compare Operation (Synchronous)** — Synchronous compare to a directory entry.
- **`ldap_compare_ext()`—Perform an LDAP Compare Operation with Controls** on page 28 — Asynchronous compare to a directory entry with controls.

The `ldap_compare_ext_s()` API supports LDAP V3 server controls and client controls.

API introduced: V4R5

```
Top | "Lightweight Directory Access Protocol (LDAP) APIs," on page 1 | APIs by category
```

### `ldap_compare_s()`—Perform an LDAP Compare Operation (Synchronous)

**Syntax**

```c
#include <ldap.h>

int ldap_compare_s(
    LDAP *ld,
    const char *dn,
    const char *attr,
    const char *value)
```

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

The `ldap_compare_s()` function is used to perform an LDAP compare operation. The API uses as input the distinguished name (DN) of the entry on which to perform the compare, and uses an `attr` and `value` (the attribute type and value to compare to those found in the entry).

Binary values are not supported by this API. Use “`ldap_compare_ext_s()`—Perform an LDAP Compare Operation with Controls (Synchronous)” on page 30 if binary values must be compared.

`ldap_compare_s()` is a synchronous request.

### Authorities and Locks

No i5/OS authority is required. All authority checking is done by the LDAP server.
Parameters

\(ld\) (Input) The LDAP pointer returned by a previous call to \("\text{ldap_init()—Perform an LDAP Initialization Operation}\) on page 105, \("\text{ldap_ssl_init }\) \text{-Initialize an SSL Connection.}\) on page 204, or \("\text{ldap_open()—Perform an LDAP Open Operation}\) on page 135.

\(dn\) (Input) The distinguished name (DN) of the entry upon which to perform the compare.

\(attr\) (Input) The attribute type to use in the comparison.

\(value\) (Input) The string attribute value to compare against the value in the entry.

Return Value

\text{LDAP\_COMPARE\_TRUE}\n if the entry contains the attribute value.

\text{LDAP\_COMPARE\_FALSE}\n if the entry does not contain the attribute value.

another LDAP error code
 if the request was not successful.

Error Conditions

The \text{ldap\_compare\_s()} API will return an LDAP error code if not successful. See \("\text{LDAP Client API Error Conditions}\) on page 300 for possible LDAP error code values.

Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_compare_s API.</td>
</tr>
</tbody>
</table>

Related Information

- \("\text{ldap\_compare()}—Perform an LDAP Compare Operation}\) on page 26 — Asynchronous compare to a directory entry.
- \("\text{ldap\_compare\_ext()}—Perform an LDAP Compare Operation with Controls}\) on page 28 — Asynchronous compare to a directory entry with controls.
- \("\text{ldap\_compare\_ext\_s()}—Perform an LDAP Compare Operation with Controls (Synchronous)}\) on page 30 — Synchronous compare to a directory entry with controls.

API introduced: V4R3

\text{ldap\_controls\_free()}—Free storage allocated by the LDAP library

\text{Syntax}

\#include <ldap.h>

\text{void ldap\_controls\_free(LDAPControl **ctrls)\n}

\text{Library Name/Service Program: QSYS/QGLDCLNT}

\text{Default Public Authority: *USE}

\text{Threadsafe: Yes}
The **ldap_controls_free()** routine is used to free storage allocated by the LDAP APIs that uses an array of LDAPControl structure.

### Authorities and Locks
No i5/OS authority is required.

### Parameters

**ctrls**  (Input) The address of an LDAPControl list, represented as a NULL-terminated array of pointers to LDAPControl structures.

### Return Value
None.

### Error Conditions
The **ldap_controls_free()** API does not return an error code.

### Error Messages
The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_controls_free API.</td>
</tr>
</tbody>
</table>

### Related Information
- **“ldap_ber_free()—Free storage allocated by the LDAP library” on page 22** — Free storage allocated for BerElement structure.
- **“ldap_memfree()—Free Memory Allocated by LDAP API” on page 110** — Free storage that has been allocated by the LDAP client library.
- **“ldap_control_free()—Free storage allocated by the LDAP library”** — Free a single LDAPControl structure.
- **“ldap_msgfree()—Free LDAP Result Message” on page 125** — Free LDAP Result Message.
- **“ldap_mods_free()—Free LDAP Modify Storage” on page 123** — Free an array of pointers to mod structures.
- **“ldap_parse_result()—Extract Information from Results” on page 144** — Extract Information from Results

API introduced: V4R5

---

### ldap_control_free()—Free storage allocated by the LDAP library

**Syntax**

```c
#include <ldap.h>

void ldap_control_free(LDAPControl *ctrl)
```

**Library Name/Service Program:** QSYS/QGLDCLNT

**Default Public Authority:** *USE

**Threadsafe:** Yes
The `ldap_control_free()` routine is used to free storage allocated by the LDAP APIs that uses an LDAPControl structure.

**Authorities and Locks**
No i5/OS authority is required.

**Parameters**

`ctrl` (Input) The address of an LDAPControl structure.

**Return Value**
None.

**Error Conditions**
The `ldap_control_free()` API does not return an error code.

**Error Messages**
The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_control_free API.</td>
</tr>
</tbody>
</table>

**Related Information**
- "ldap_controls_free()—Free storage allocated by the LDAP library” on page 32 — Free an array of LDAPControl structures.
- "ldap_parse_result()—Extract Information from Results” on page 144 — Extract Information from Results
- "ldap_create_proxyauth_control()—Create Control for Proxy Identity” on page 47 — Create Control for Proxy Identity

API introduced: V4R5

The `ldap_copy_controls()` function is used to make a copy of the list of LDAP server controls.

**Syntax**

```c
#include <ldap.h>

int ldap_copy_controls(
    LDAPControl ***to_here,
    LDAPControl **from)
```

**Library Name/Service Program:** QSYS/QGLDCLNT
**Default Public Authority:** *USE
**Threadsafe:** Yes

The `ldap_copy_controls()` function is used to make a copy of the list of LDAP server controls.
Authorities and Locks
No i5/OS authority is required. All authority checking is done by the LDAP server.

Parameters

to_here (Input) Specifies the location to which to copy the control list. See LDAP Controls for more information about server controls.

from (Input) Specifies the location of the control list to be copied. See LDAP Controls for more information about server controls.

Return Value
LDAP_SUCCESS
if the request was successful.

LDAP_NO_MEMORY
if the control list could not be copied.

Related Information
• “ldap_insert_control()—Insert a Control in the List of LDAP Server Controls” on page 108 — Insert a control in the list of LDAP server controls.
• “ldap_add_control()—Create a Control and Insert it into the List of LDAP Server Controls” on page 10 — Add a new LDAP server control.
• “ldap_remove_control()—Remove a Control from the List of LDAP Server Controls” on page 151 — Remove a control from the list of LDAP server controls.

API introduced: V5R3

ldap_count_attributes()—Retrieve Count of Attributes for an LDAP Entry

Syntax
#include <ldap.h>

int ldap_count_attributes(
    LDAP *ld,
    LDAPMessage *entry)

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

The ldap_count_attributes() function returns a count of the number of attributes in an LDAP entry.

Authorities and Locks
No i5/OS authority is required.

Parameters

ld (Input) The LDAP pointer returned by a previous call to “ldap_init()—Perform an LDAP
Initialization Operation” on page 105 “ldap_ssl_init — Initialize an SSL Connection.” on page 204, or “ldap_open()—Perform an LDAP Open Operation” on page 135.

entry (Input) The attribute information as returned by “ldap_first_entry()—Retrieve First LDAP Entry” on page 81 or “ldap_next_entry()—Retrieve Next LDAP Entry” on page 131.

Return Value
Number of Attributes
if the request was successful.
-1 if the request was not successful.

Error Conditions
The ldap_count_attributes() API returns -1 if a null entry is passed as input to ldap_count_attributes().

Error Messages
The following message may be sent from this function.

Message ID Error Message Text
CPF3CF2 E Error(s) occurred during running of ldap_count_attributes API.

Related Information
• “ldap_first_attribute()—Retrieve First Attribute in an Entry” on page 79 — Return first attribute name in an entry.
• “ldap_next_attribute()—Retrieve Next Attribute in an Entry” on page 129 — Return next attribute name in an entry.
• “ldap_first_entry()—Retrieve First LDAP Entry” on page 81 — Retrieve First LDAP Entry
• “ldap_next_entry()—Retrieve Next LDAP Entry” on page 131 — Retrieve Next LDAP Entry

API introduced: V4R3

ldap_count_entries()—Retrieve Count of LDAP Entries

Syntax
#include <ldap.h>

int ldap_count_entries(
        LDAP *ld,
        LDAPMessage *result)

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

The ldap_count_entries() API returns the number of entries contained in a search result chain. It can also be used to count the number of entries that remain in a chain if called with a message, entry or continuation reference returned by ldap_first_message(), ldap_next_message(), ldap_first_entry(), ldap_next_entry(), ldap_first_reference() or ldap_next_reference(), respectively.

Authorities and Locks
No i5/OS authority is required.
Parameters

ld (Input) The LDAP pointer returned by a previous call to “ldap_init()—Perform an LDAP Initialization Operation” on page 105, “ldap_ssl_init —Initialize an SSL Connection.” on page 204, or “ldap_open()—Perform an LDAP Open Operation” on page 135.

result (Input) The result returned by a call to “ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156 or by one of synchronous search routines (“ldap_search_s()—Perform an LDAP Search Operation (Synchronous)” on page 171 or “ldap_search_st()—Perform an LDAP Search Operation (Timed Synchronous)” on page 173).

Return Value

Number of Entries

If the request is successful, ldap_count_entries() returns the number of entries contained in a search result chain. It can also be used to count the number of entries that remain in a chain if called with a message, entry or continuation reference.

-1 if the request was not successful.

Error Conditions

If ldap_count_entries() is not successful, ld_errno will be set to indicate the error. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values. Use “ldap_get_errno()—Retrieve Error Information” on page 91 function to retrieve the error information.

Error Messages

The following message may be sent from this function.

Message ID   Error Message Text
CPF3CF2 E     Error(s) occurred during running of ldap_count_entries API.

Related Information

• “ldap_first_entry()—Retrieve First LDAP Entry” on page 81 — Return first entry in a chain of search results.
• “ldap_next_entry()—Retrieve Next LDAP Entry” on page 131 — Return next entry in a chain of search results.
• “ldap_get_entry_controls_np()—Extract Server Controls from an Entry” on page 90 — Extract server controls from an entry.
• “ldap_first_reference()—Retrieve First Continuation Reference in a Chain of Search Results” on page 84 — Return first continuation reference in a chain of search results.
• “ldap_next_reference()—Retrieve the next Continuation Reference in a Chain of Search Results” on page 134 — Return next continuation reference in a chain of search results.
• “ldap_count_references()—Count continuation references in a chain of search results” on page 39 — Return number of continuation reference in a chain of search results.
• “ldap_parse_reference_np()—Extract Information from a Continuation Reference” on page 143 — Extract information from a continuation reference.

API introduced: V4R3
ldap_count_messages()—Count messages in a result chain

Syntax
#include <ldap.h>

int ldap_count_messages(LDAP *ld,
LDAPMessage *result)

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

The ldap_count_messages() routine is used to step through the list of messages in a result chain, as returned by “ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156. It is used to count the number of messages returned. The “ldap_msgtype()—Retrieve the Type of an LDAP Message” API can be used to distinguish between the different message types.

In addition to returning the number of messages contained in a chain of results, the ldap_count_messages() API can be used to count the number of messages that remain in a chain if called with a message, entry, or reference returned by ldap_first_message(), ldap_next_message(), ldap_first_entry(), ldap_next_entry(), ldap_first_reference() and ldap_next_reference().

Authorities and Locks
No i5/OS authority is required.

Parameters
ld (Input) The LDAP pointer returned by a previous call to “ldap_init()—Perform an LDAP initialization Operation” on page 105, “ldap_ssl_init —Initialize an SSL Connection.” on page 204, or “ldap_open()—Perform an LDAP Open Operation” on page 135.

result (Input) The result returned by a call to ldap_result() or one of the synchronous search routines (“ldap_search_s()—Perform an LDAP Search Operation (Synchronous)” on page 171, “ldap_search_st()—Perform an LDAP Search Operation (Timed Synchronous)” on page 173, or “ldap_search_ext_s — Synchronously Search the Directory Using Controls” on page 168).

Return Value
Number of Messages
If the request was successful, ldap_count_messages() API returns the number of messages in a result chain or number of messages that remain in a chain, as returned by ldap_result().

-1 if the request was not successful.

Error Conditions
If ldap_count_messages() is not successful, ld_errno will be set to indicate the error. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values. Use “ldap_get_errno()—Retrieve Error Information” on page 91 function to retrieve the error information.

Error Messages
The following message may be sent from this function.

Message ID   Error Message Text
CPF3CF2 E Error(s) occurred during running of ldap_count_messages API.
Related Information

- "ldap_first_message()—Retrieve First LDAP Message" on page 82 — Return first message in a result chain.
- "ldap_next_message()—Retrieve Next LDAP Message" on page 132 — Return next message in a result chain.

API introduced: V4R5

### ldap_count_references()—Count continuation references in a chain of search results

**Syntax**
```
#include <ldap.h>

int ldap_count_references(LDAP *ld, LDAPMessage *result)
```

### Library Name/Service Program
QSYS/QGLDCLNT

### Default Public Authority
*USE

### Threadsafe
Yes

The `ldap_count_references()` API is used to count the number of continuation references returned. It can also be used to count the number of continuation references that remain in a chain.

### Authorities and Locks
No i5/OS authority is required.

### Parameters
- **ld** (Input) The LDAP pointer returned by a previous call to "ldap_init()—Perform an LDAP Initialization Operation" on page 105, "ldap_ssl_init — Initialize an SSL Connection." on page 204, or "ldap_open()—Perform an LDAP Open Operation" on page 135.
- **result** (Input) The result returned by a call to "ldap_result()—Retrieve Result of an Asynchronous LDAP Operation" on page 156 or one of the synchronous search routines "ldap_search_st()—Perform an LDAP Search Operation (Synchronous)" on page 171, "ldap_search_st()—Perform an LDAP Search Operation (Timed Synchronous)" on page 173, or "ldap_search_ext_s — Synchronously Search the Directory Using Controls" on page 168.

### Return Value
Number of continuation reference

- If the request was successful, `ldap_count_references()` API returns the number of continuation references in a result chain or number of continuation references that remain in a chain, as returned by `ldap_result()`.
- -1 if the request was not successful.

### Error Conditions
If `ldap_count_references()` is not successful, `ld_errno` will be set to indicate the error. See "LDAP Client API Error Conditions" on page 300 for possible LDAP error code values. Use "ldap_get_errno()—Retrieve Error Information" on page 91 function to retrieve the error information.
Error Messages
The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_count_references API.</td>
</tr>
</tbody>
</table>

Related Information

- **“ldap_first_reference()—Retrieve First Continuation Reference in a Chain of Search Results” on page 84** — Return first continuation reference in a result chain, as returned by ldap_result.
- **“ldap_next_reference()—Retrieve the next Continuation Reference in a Chain of Search Results” on page 134** — Return next continuation reference in a result chain, as returned by ldap_result.

API introduced: V4R5

---

**ldap_count_values()—Retrieve Count of Attribute Values**

Syntax

```c
#include <ldap.h>

int ldap_count_values(
    char **vals)
```

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

The `ldap_count_values()` function returns the number of values in the array returned by the **“ldap_get_values()—Retrieve a Set of Attribute Values from an Entry” on page 102** function.

Authorities and Locks

No i5/OS authority is required.

Parameters

- `vals` (Input) A pointer to a null-terminated array of attribute values, as returned by **“ldap_get_values()—Retrieve a Set of Attribute Values from an Entry” on page 102**

Return Value

Number of Values
- if the request is successful, `ldap_count_values()` returns the number of values in the array returned by the **“ldap_get_values()—Retrieve a Set of Attribute Values from an Entry” on page 102** function.
- `-1` if the request was not successful.

Error Conditions

If `ldap_count_values()` is not successful, `ld_errno` will be set to indicate the error. See **“LDAP Client API Error Conditions” on page 300** for possible LDAP error code values. Use **“ldap_get_errno()—Retrieve Error Information” on page 91** function to retrieve the error information.
Error Messages
The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_count_values API.</td>
</tr>
</tbody>
</table>

Related Information

- "ldap_get_values()—Retrieve a Set of Attribute Values from an Entry” on page 102 — Return an attribute’s values.
- "ldap_get_values_len()—Retrieve a Set of Binary Attribute Values” on page 104 — Return an attribute’s binary values.
- "ldap_count_values_len()—Retrieve Count of Binary Attribute Values” — Return number of binary values.
- "ldap_value_free()—Free Memory Allocated by ldap_get_values()” on page 223 — Free memory allocated by ldap_get_values.
- "ldap_value_free_len()—Free Memory Allocated by ldap_get_values_len()” on page 224 — Free memory allocated by ldap_get_values_len.

API introduced: V4R3

Syntax

```c
#include <ldap.h>

struct berval {
    unsigned long bv_len;
    char *bv_val;
};

int ldap_count_values_len(
    struct berval **bvals)
```

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

The `ldap_count_values_len()` function returns the number of values in the array returned by the "ldap_get_values_len()—Retrieve a Set of Binary Attribute Values” on page 104 function. The array of values returned can be freed by calling "ldap_value_free_len()—Free Memory Allocated by ldap_get_values_len()” on page 224.

Authorities and Locks
No i5/OS authority is required.

Parameters

*bvals* (Input) A pointer to a null-terminated array of pointers to berval structures, as returned by "ldap_get_values_len()—Retrieve a Set of Binary Attribute Values” on page 104.
Return Value

Number of Values
if the request is successful, \texttt{ldap_count_values_len()} returns the number of values in the array returned by the \texttt{ldap_get_values().}

-1 if the request was not successful.

Error Conditions
if \texttt{ldap_count_values_len()} is not successful, \texttt{ld_errno} will be set to indicate the error. See \texttt{LDAP Client API Error Conditions} on page 300 for possible LDAP error code values. Use \texttt{ldap_get_errno()} function to retrieve the error information.

Error Messages
The following message may be sent from this function.

Message ID | Error Message Text
---|---
CPF3CF2 E | Error(s) occurred during running of \texttt{ldap_count_values_len} API.

Related Information
- \texttt{ldap_get_values()—Retrieve a Set of Attribute Values from an Entry} on page 102 — Return an attribute’s values.
- \texttt{ldap_get_values_len()—Retrieve a Set of Binary Attribute Values} on page 104 — Return an attribute’s binary values.
- \texttt{ldap_count_values()—Retrieve Count of Attribute Values} on page 40 — Return number of values.
- \texttt{ldap_value_free()—Free Memory Allocated by ldap_get_values()} on page 223 — Free memory allocated by \texttt{ldap_get_values().}
- \texttt{ldap_value_free_len()—Free Memory Allocated by ldap_get_values_len()} on page 224 — Free memory allocated by \texttt{ldap_get_values_len().}

API introduced: V4R3

\begin{verbatim}
#include <ldap.h>

int ldap_create_page_control(
    LDAP *ld,
    unsigned long pageSize,
    struct berval *cookie,
    const char *isCritical,
    LDAPControl **control)
\end{verbatim}

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

\texttt{ldap_create_page_control()}—Create a Paged Results Control used when paging search results

Syntax
#include <ldap.h>

\begin{verbatim}
int ldap_create_page_control(
    LDAP *ld,
    unsigned long pageSize,
    struct berval *cookie,
    const char *isCritical,
    LDAPControl **control)
\end{verbatim}

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes
The `ldap_create_page_control()` function is used to create a paged results control used when paging search results.

See [LDAP Paged Results](#) for usage information about the functions used to perform paging of entries returned from the server following an LDAP search operation.

Note: cookie and control must be freed by the caller.

**Authorities and Locks**

No i5/OS authority is required. All authority checking is done by the LDAP server.

**Parameters**

`ld`  
(Input) Specifies the LDAP pointer returned by previous call to `ldap_init()`, `ldap_ssl_init()` or `ldap_open()`. Must not be NULL.

`pageSize`  
(Input) Specifies the number of entries to be returned for this paged results search result.

`cookie`  
(Input) For the first call to `ldap_create_page_control()` for a particular paged search, cookie should be set to NULL. On subsequent calls to `ldap_create_page_control()` for a particular paged search, cookie should be set by the preceeding call to `ldap_parse_page_control()`. See the example at [LDAP Paged Results](#).

`isCritical`  
(Input) Specifies the criticality of paged results on the search. This determines what happens when the server does not support paged results. If the criticality is set to TRUE (‘T’) and the server does not support paged results, then the search does not continue. However, if the criticality is set to FALSE (‘F’) the search will continue without paged results.

`control`  
(Output) Specifies the result parameter that is filled in with an allocated array of one control for the paging function. The control must be freed by calling `ldap_control_free()`.

**Return Value**

- `LDAP_SUCCESS` if successful.
- `LDAP_NO_MEMORY` if memory cannot be acquired.
- `LDAP_PARAM_ERROR` if ld or control is NULL.

**Example**

See [Code disclaimer information](#) for information pertaining to code examples.

The following example uses `ldap_create_page_control()` and `ldap_parse_page_control()` to perform an LDAP sorted search.

```c
passport/ldap/ldapPageS.c
*/
/* Description: An example on how to perform an LDAP Paged Search */
/* using the following APIs */
/*  - ldap_create_page_control() */
/*  - ldap_parse_page_control() */
```
/* How to compile: */
/* Assuming the C source for LDAPPageS.c has been downloaded into */
/* a source physical file MyLDAPLib/QCSRC (with record length of at */
/* least 146 characters) as member LDAPPageS on an iSeries, */
/* program LDAPPageS (*PGM) can be built via the following commands: */
/* */
/* CRTCMOD MODULE(MyLDAPLib/LDAPPageS) */
/* SRCFILE(MyLDAPLib/QCSRC) */
/* */
/* CRTPGM PGM(MyLDAPLib/LDAPPageS) */
/* MODULE(MyLDAPLib/LDAPPageS) */
/* BNDSRVPGM(QSYS/QGLDCLNT QSYS/QGLDBRDR) */
/* */
/* Usage: */
/* The input parameters are as follows: */
/* */
/* argv[1] = Search base */
/* argv[2] = filter */
/* argv[3] = search scope (0=base, 1=onelevel, OR 2=subtree) */
/* argv[4] = Page size */
/* */
/* ***********************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <time.h>
#include <ldap.h>
#include <lber.h>

static char ibmid[] = "Copyright IBM Corporation 2003 LICENSED MATERIAL - PROGRAM PROPERTY OF IBM";
#define BIND_DN "cn=administrator"
#define BIND_PW "adminpwd"

int scope;
LDAP *ld;

int main (int argc, char *argv[])
{
    int l_rc, l_entries, l_port, l_entry_count=0, morePages, l_errcode=0, page_nbr;
    unsigned long pageSize;
    struct berval *cookie=NULL;
    char *l_dn;
    unsigned long totalCount;
    LDAPControl *pageControl=NULL, *M_controls[2] = { NULL, NULL }, **returnedControls = NULL;
    LDAPMessage *l_result, *l_entry;

    /***********************************************************************************/
    /* Check input parameters */
    /* */
    if (argc < 5)
    {
        printf("The input parameters are as follows:\n");
        printf("1. Search base\n");
        printf("2. Filter\n");
        printf("3. Search Scope (0=base, 1=onelevel, OR 2=subtree)\n");
        printf("4. Page size\n");
        return 0;
    }
    /* */
    /* ***********************************************************************************/
    /* Set default values: Server and Port. And then parse */
    /* input parameters into program variables */

44 IBM Systems - iSeries: Lightweight Directory Access Protocol (LDAP) APIs
/*
server = NULL;
l_port = LDAP_PORT;
*/
base = argv[1];
filter = argv[2];
scope = atoi(argv[3]);
pageSize = atoi(argv[4]);
/*
*******************************************************************************/
/* Initialize an LDAP session
*******************************************************************************/
ld = ldap_init(server, l_port);
/* Check if connection is OK
*******************************************************************************/
if (ld == NULL)
{
printf("==Error==");
printf(" Init of server %s at port %d failed.\n", server, l_port);
return 0;
}
/*
*******************************************************************************/
/* Bind as the ldap administrator
*******************************************************************************/
l_rc = ldap_simple_bind_s(ld, BIND_DN, BIND_PW);
if (l_rc != LDAP_SUCCESS)
{
printf("==Error== %s");
printf(" Unable to Bind to the LDAP server. Return code is %d.\n", l_rc);
return 0;
}
/*
*******************************************************************************/
printf(" The search parms were:\n");
printf(" base: %s\n", base);
printf(" scope: %s\n", scopes[scope]);
printf(" filter: %s\n", filter);
printf(" page size: %d\n", pageSize);
printf(" The entries returned were:\n");
page_nbr = 1;
/*
*******************************************************************************/
/* Get one page of the returned results each time */
/* through the loop */
do {
    l_rc = ldap_create_page_control(ld, pageSize, cookie, pagingCriticality, &pageControl);
    /* Insert the control into a list to be passed to the search. */
    M_controls[0] = pageControl;
    /* Search for entries in the directory using the parameters. */
    l_rc = ldap_search_ext_s(ld, base, scope, filter, NULL, 0, M_controls, NULL, NULL, 0, &l_result);
    if ((l_rc != LDAP_SUCCESS) & (l_rc != LDAP_PARTIAL_RESULTS))
    {
        printf("==Error==");
        printf(" Failure during a search. Return code is %d.\n", l_rc);
        ldap_unbind(ld);
        break;
    }
    /* Parse the results to retrieve the controls being returned. */
l_rc = ldap_parse_result(ld, l_result, &l_errcode, NULL, NULL, NULL, &returnedControls, LDAP_FALSE);

if (cookie != NULL)
{
    ber_bvfree(cookie);
    cookie = NULL;
}

/* Parse the page control returned to get the cookie and determine whether there are more pages. */
l_rc = ldap_parse_page_control(ld, returnedControls, &totalCount, &cookie);

/* Determine if the cookie is not empty, indicating there are more pages for these search parameters. */
if (cookie && cookie->bv_val != NULL && (strlen(cookie->bv_val) > 0))
{
    morePages = LDAP_TRUE;
}
else
{
    morePages = LDAP_FALSE;
}

/* Cleanup the controls used. */
if (returnedControls != NULL)
{
    ldap_controls_free(returnedControls);
    returnedControls = NULL;
}
M_controls[0] = NULL;
ldap_control_free(pageControl);
pageControl = NULL;

/***********************************************************/
/* Display the returned result */
/***********************************************************/
/* Determine how many entries have been found. */
if (morePages == LDAP_TRUE)
    printf("== Page: %d ==\n", page_nbr);
l_entries = ldap_count_entries(ld, l_result);
if (l_entries > 0)
{
    l_entry_count = l_entry_count + l_entries;
}

for (l_entry = ldap_first_entry(ld, l_result);
    l_entry != NULL;
    l_entry = ldap_next_entry(ld, l_entry) )
{
    l_dn = ldap_get_dn(ld, l_entry);
    printf(" %s\n", l_dn);
}

/* Free the search results. */
ldap_msgfree(l_result);
page_nbr = page_nbr + 1;

} while (morePages == LDAP_TRUE);

printf("\n %d entries found during the search", l_entry_count);
/* Free the cookie since all the pages for these search parameters */
/* have been retrieved. */
ber_bvfree(cookie);
cookie = NULL;
/* Close the LDAP session. */
ldap_unbind(ld);

return 0;

Related Information

- "ldap_parse_page_control()—Retrieve Values in a Paged Results Control” on page 139 — Retrieve values in a paged results control.
- "ldap_create_sort_keylist()—Create a Structure with Sort Key Values used when Sorting Search Results” on page 51 — Create a structure with sort key values.
- "ldap_free_sort_keylist()—Free all Memory used by the Sort Key List” on page 85 — Free all memory used by the sort key list.
- "ldap_create_sort_control()—Create a Sorted Results Control used when Sorting Search Results” on page 49 — Create a sorted results control.
- "ldap_parse_sort_control()—Retrieve Values in a Sorted Results Control” on page 148 — Retrieve values in a sorted results control.
- "ldap_search()—Perform an LDAP Search Operation” on page 164 — Asynchronously search the directory.
- "ldap_search_ext —Asynchronously Search the Directory Using Controls” on page 166 — Asynchronously search the directory with controls.
- "ldap_search_ext_s — Synchronously Search the Directory Using Controls” on page 168 — Synchronously search the directory with controls.
- "ldap_parse_result()—Extract Information from Results” on page 144 — Extract information from results.
- "ldap_init()—Perform an LDAP Initialization Operation” on page 105 — Initializes a session with an LDAP server.
- "ldap_ssl_init —Initialize an SSL Connection.” on page 204 — Initializes an SSL connection
- "ldap_open()—Perform an LDAP Open Operation” on page 135 — Open a connection to an LDAP server
- "ldap_control_free()—Free storage allocated by the LDAP library” on page 33 — Free a single LDAPControl structure.
- "ldap_controls_free()—Free storage allocated by the LDAP library” on page 32 — Free an array of LDAPControl structures.

API introduced: V5R3

```
#include <ldap.h>
int ldap_create_proxyauth_control(
    LDAP *ld,
    char *proxyID,
)
```

Lightweight Directory Access Protocol (LDAP) APIs 47
int *isCritical,
LDAPControl **control)

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

The `create_proxauth_control()` function is used to create the LDAP control that will allow a bind entity to assume a proxy identity. The LDAP control must contain the proxy authorization identity which will be included in LDAP operations to request an operation from the server.

Using the proxy authorization control mechanism, a client can bind to the LDAP directory using its own identity, but is granted proxy authorization rights of another user to access the target directory.

When the IBM Directory Server receives an operation with proxy authorization control, the bind DN is validated against the administrative group and/or the predefined proxy authorization group to determine whether the bind DN should be granted the proxy authorization right. In other words, the bound application client must be a member of the administrative group or proxy authorization group in order to request a proxy authorization operation.

**Authorities and Locks**
No i5/OS authority is required.

**Parameters**

`ld` (Input) Specifies the LDAP pointer returned by previous call to `ldap_init()`, `ldap_ssl_init()` or `ldap_open()`. Must not be NULL.

`proxyID` (Input) Specifies the authorization identity the client will assume. The authorization identity can be specified in two forms:
- `u:user-name` specifies the authorization identity using a user name.
- `dn:distinguished-name` specifies the authorization identity using an LDAP distinguished name.

`isCritical` (Input) Specifies whether the proxy authorization control is critical to the search operation. This should be set to a non-zero value since the proxy authorization control is critical.

`control` (Output) Specifies a pointer to a structure created by the function for the proxy identity control. See [LDAP Controls](#) for more information about server controls. This control should be freed by calling the `ldap_control_free()` function when finished using the control.

**Return Value**

`LDAP_SUCCESS` if the request was successful.

`LDAP_PARAM_ERROR` if a non-valid parameter was passed.

`LDAP_NO_MEMORY` if memory cannot be allocated.

`LDAP_ENCODING_ERROR` if an error occurred when encoding the control.
LDAP_UNAVAILABLE_CRITICAL_EXTENSION
if server does not support proxy authorization and iscritical is set to a non-zero value.

Related Information
- ldap_control_free()—Free storage allocated by the LDAP library” on page 33 — Free a single LDAPControl structure.

API Introduced: V5R4

ldap_create_sort_control()—Create a Sorted Results Control used when Sorting Search Results

Syntax
#include <ldap.h>

int ldap_create_sort_control(
    LDAP *ld, 
    LDAPsortkey **sortKeyList, 
    const char *isCritical, 
    LDAPControl **control)

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

The ldap_create_sort_control() function is used to create a sorted results control used when sorting search results.

See LDAP Sort for usage information about the functions used to perform sorting of entries returned from the server following an LDAP search operation.

Authorities and Locks
No i5/OS authority is required. All authority checking is done by the LDAP server.

Parameters
ld (Input) Specifies the LDAP pointer returned by previous call to ldap_init(), ldap_ssl_init() or ldap_open(). Must not be NULL.

sortKeyList (Input) Specifies the pointer to an array of LDAPsortkey structures, which represent attributes that the server uses to sort returned entries. This is obtained by a previous call to ldap_create_sort_keylist().

isCritical (Input) Specifies the criticality of sort on the search. If the criticality of sort is FALSE, and the server finds a problem with the sort criteria, the search continues but entries returned are not sorted. If the criticality of sort is TRUE, and the server finds a problem with the sort criteria, the search does not continue, no sorting is done, and no entries are returned. If the server does not find any problem with the sort criteria, the search and sort continues and entries are returned sorted.
control
(Output) Specifies the result parameter that is filled in with an allocated array of one control for the sort function. The control must be freed by calling ldap_control_free().

Return Value

LDAP_SUCCESS
if the request was successful.

LDAP_PARAM_ERROR
if ld, sortKeyList or control is NULL.

LDAP_NO_MEMORY
if memory cannot be acquired.

LDAP_ENCODING_ERROR
if an underlying ber encoding function fails.

Example

See Code disclaimer information for information pertaining to code examples.

See the example that uses ldap_create_sort_keylist(), ldap_create_sort_control(), ldap_free_sort_keylist(), and ldap_parse_sort_control() to perform an LDAP sorted search in “ldap_create_sort_keylist()—Create a Structure with Sort Key Values used when Sorting Search Results” on page 51—Create a Structure with Sort Key Values used when Sorting Search Results.

Related Information

- “ldap_create_sort_keylist()—Create a Structure with Sort Key Values used when Sorting Search Results” on page 51 — Create a structure with sort key values.
- “ldap_free_sort_keylist()—Free all Memory used by the Sort Key List” on page 85 — Free all memory used by the sort key list.
- “ldap_parse_sort_control()—Retrieve Values in a Sorted Results Control” on page 148 — Retrieve values in a sorted results control.
- “ldap_create_page_control()—Create a Paged Results Control used when paging search results” on page 42 — Create a paged results control.
- “ldap_parse_page_control()—Retrieve Values in a Paged Results Control” on page 139 — Retrieve values in a paged results control.
- “ldap_search()—Perform an LDAP Search Operation” on page 164 — Asynchronously search the directory.
- “ldap_parse_result()—Extract Information from Results” on page 144 — Extract information from results.
- “ldap_init()—Perform an LDAP Initialization Operation” on page 105 — Initializes a session with an LDAP server.
- “ldap_ssl_init —Initialize an SSL Connection.” on page 204 — Initializes an SSL connection
- “ldap_open()—Perform an LDAP Open Operation” on page 135 — Open a connection to an LDAP server
ldap_create_sort_keylist()—Create a Structure with Sort Key Values used when Sorting Search Results

Syntax
#include <ldap.h>

int ldap_create_sort_keylist(
     LDAPsortkey ***sortKeyList,
     const char *sortString)

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

The ldap_create_sort_keylist() function is used to create a structure with sort key values used when sorting search results.

See LDAP Sort for usage information about the functions used to perform sorting of entries returned from the server following an LDAP search operation.

Authorities and Locks
No i5/OS authority is required. All authority checking is done by the LDAP server.

Parameters
sortKeyList
   (Output) On return will point to an array of LDAPsortkey structures, which represent attributes that the server uses to sort returned entries. Input when used for ldap_create_sort_control() and ldap_free_sort_keylist(). This list must be freed by the caller by calling ldap_free_sort_keylist.

sortString
   (Input) Specifies the string with one or more attributes to be used to sort entries returned by the server. Multiple sort keys are separated with a space character. (e.g. “sn -givenname”)

Return Value
LDAP_SUCCESS
   if the request was successful.

another LDAP error code
   if the keylist could not be created.

Example
See Code disclaimer information for information pertaining to code examples.

The following example uses ldap_create_sort_keylist(), ldap_create_sort_control(), ldap_free_sort_keylist(), and ldap_parse_sort_control() to perform an LDAP sorted search.

/***********************************************************************/
/* File Name: LDAPSortS.c */
/**/
/* Description: An example of how to perform an LDAP Sorted Search */
/* using the following APIs */
/* - ldap_create_sort_keylist() */
/* - ldap_create_sort_control() */
/* - ldap_free_sort_keylist() */
/* - ldap_parse_sort_control() */
/* */
/* How to compile: */
/* Assuming the C source for LDAPSortS.c has been downloaded into */
/* a source physical file MyLDAPLib/QCSRC (with record length of at */
/* least 132 characters) as member LDAPSortS on an iSeries, */
/* program LDAPSortS (+PGM) can be built via the following commands: */
/* */
/* CRTCMOD MODULE(MyLDAPLib/LDAPSORTS) */
/* SRCFILE(MyLDAPLib/QCSRC) */
/* */
/* CRTPGM PGM(MyLDAPLib/LDAPSORTS) */
/* MODULE(MyLDAPLib/LDAPSORTS) */
/* BNDSRVPGM(QSYS/QGLDCLNT) */
/* */
/* Usage: */
/* The input parameters are as follows: */
/* */
/* argv[1] = Search base */
/* argv[2] = filter */
/* argv[3] = search scope (0=base, 1=onelevel, OR 2=subtree) */
/* argv[4] = sort attribute for sort criteria. To sort in reverse */
/* order, use a minus sign (-) as a prefix. */
/* Ex: for ascending order on the sn attribute; use "sn" */
/* for descending order on the sn attribute; use ":-sn" */
/* */
//***********************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <time.h>
#include <ldap.h>

static char ibmid[] = "Copyright IBM Corporation 2003 LICENSED MATERIAL - PROGRAM PROPERTY OF IBM";
define BIND_DN "cn=administrator"
define BIND_PW "adminpwd"

int scope;
LDAP *ld;

int main (int argc, char *argv[])
{
    int l_rc, l_entries, l_port, l_entry_count=0, l_search_count=0, l_errcode=0;
    char *sortString, sortCriticality = 'T', *l_dn, *attrInError = NULL;
    unsigned long sortRC=0;
    LDAPMessage *l_result, *l_entry;
    LDAPsortkey **sortKeyList = NULL;
    LDAPControl *sortControl = NULL, *M_controls[2] = { NULL, NULL }, **returnedControls = NULL;

    /**********************************************************************/
    /* Check input parameters */
    /* */
    /* */
    if (argc < 5)
    {
        printf("The input parameters are as follows:\n");
        printf("  1. Search base\n");
        printf("  2. Filter\n");
        printf("  3. Search Scope (0=base, 1=onelevel, OR 2=subtree)\n");
        printf("  4. Sorting Attribute Ex: for ascending order on the sn attribute; use \"sn\"\n");
        printf(" for descending order on the sn attribute; use \"-sn\"\n");
    }
return 0;
} */
/* *******************************************************************/

/* Set default values: Server and Port. And then parse */
/* input parameters into program variables */
/* */
server = NULL;
l_port = LDAP_PORT;
base = argv[1];
filter = argv[2];
scope = atoi(argv[3]);
sortString = argv[4];
/* */
/* *******************************************************************/

/* Initialize an LDAP session */
/* */
ld = ldap_init(server, l_port);
if (ld == NULL)
{
    printf("==Error==");
    printf(" Init of server %s at port %d failed.\n", server, l_port);
    return 0;
} /* */
/* *******************************************************************/

/* Bind as the ldap administrator */
/* */
l_rc = ldap_simple_bind_s(ld, BIND_DN, BIND_PW);
if (l_rc != LDAP_SUCCESS)
{
    printf("==Error== %s"));
    printf(" Unable to Bind to the LDAP server. Return code is %d.\n", l_rc);
    return 0;
} /* */
/* *******************************************************************/

/* Create the sortKeyList structure with sort key values used */
/* when sorting search result. The structure is needed for the */
/* ldap_create_sort_control API. */
/* */
l_rc = ldap_create_sort_keylist(&sortKeyList, sortString);
if (l_rc != LDAP_SUCCESS)
{
    printf("==Error==");
    printf(" Failure during a create_sort_keylist. Return code is %d.\n", l_rc);
    ldap.unbind(ld);
    return 0;
} /* */
/* *******************************************************************/

/* Create the control for sorting to be used by the search using: */
/* the ld pointer created by ldap_init API, */
/* sortKeyList created by the ldap_create_sort_keylist API. */
/* sort Criticality defined to be True. */
/* The return is a sortControl that is used in the */
/* ldap_search_ext_s API */
l_rc = ldap_create_sort_control(ld, sortKeyList, sortCriticality, &sortControl);
/* */
/*********************************************/
/* Free all Memory used by the Sort Key List */
/* This API must be called after the ldap_create_sort_control() */
/* function has completed */
ldap_free_sort_keylist(sortKeyList);
if (l_rc != LDAP_SUCCESS)
{
    printf("==Error==");
    printf(" Failure during a create_sort_control. Return code is %d.\n",l_rc);
    ldap_unbind(ld);
    return 0;
}
/* */
/*********************************************/
/* Search for entries in the directory using the parameters */
/* */
M_controls[0] = sortControl;
l_rc = ldap_search_ext_s(ld, base, scope, filter, NULL, 0, M_controls, NULL, NULL, 0, &l_result);
if ((l_rc != LDAP_SUCCESS) & (l_rc != LDAP_PARTIAL_RESULTS))
{
    printf("==Error==");
    printf(" Failure during a search. Return code is %d.\n",l_rc);
    printf(" The search parms were:\n");
    printf(" base: %s\n",base);
    printf(" scope: %s\n",scopes[scope]);
    printf(" filter: %s\n",filter);
    ldap_unbind(ld);
    return 0;
}
/* */
/*********************************************/
/* Parse the result to get the returned controls */
l_rc = ldap_parse_result(ld, l_result, &l_errcode, NULL, NULL, NULL, NULL, &returnedControls, LDAP_FALSE);
if ((l_rc != LDAP_SUCCESS) | (l_errcode != LDAP_SUCCESS))
{
    printf("==Error==");
    printf(" Failure during parse result. Return code is %d.\n",l_rc);
    printf(" Error code is %d.\n",l_errcode);
    ldap_unbind(ld);
    return 0;
}
/* */
/*********************************************/
/* Parse the sort control returned, looking for any errors */
if (returnedControls != NULL)
{
    l_rc = ldap_parse_sort_control(ld, returnedControls, &sortRC, &attrInError);
    if ((l_rc != LDAP_SUCCESS) | (sortRC != LDAP_SUCCESS))
    {
        printf("==Error==");
        printf(" Failure during a parse_sort_control. Return code is %d.\n",l_rc);
        printf(" Error code is %d.\n",sortRC);
        ldap_unbind(ld);
        return 0;
    }
}
printf("==Error==");
printf(" Failure during parse sort control. Return code is %d.\n", l_rc);
printf(" Sort return code is %d.\n", sortRC);
printf(" ld: %p\n", ld);
printf(" returnedControls: %p\n", returnedControls);
printf(" attrInError: %p\n", attrInError);
ldap_unbind(ld);
return 0;
} /* */
/**************************************************************************/
/**************************************************************************/
/* Do the required cleanup */ /* */
/* */
ldap_controls_free(returnedControls);
returnedControls = NULL;
if (attrInError != NULL)
{
 free(attrInError);
 attrInError = NULL;
}
} /* */
M_controls[0] = NULL;
ldap_control_free(sortControl);
sortControl = NULL;
/**************************************************************************/
/**************************************************************************/
/* Display the returned result */ /* */
/* */
/* Determine how many entries have been found. */ /* */
l_entries = ldap_count_entries(ld, l_result);
if (l_entries > 0)
{
 l_entry_count = l_entry_count + l_entries;
}
printf(" The search parms were:\n");
printf(" base: %s\n", base);
printf(" scope: %s\n", scopes[scope]);
printf(" filter: %s\n", filter);
printf(" sortAttr: %s\n", sortString);
printf(" The sorted entries returned were:\n");
for ( l_entry = ldap_first_entry(ld, l_result);
 l_entry != NULL;
 l_entry = ldap_next_entry(ld, l_entry) )
{
 l_dn = ldap_get_dn(ld, l_entry);
 printf(" %s:\n", l_dn);
}
/* Free the search results. */
ldap_msgfree(l_result);
printf("\n %d entries found during the search",l_entry_count);
/* */
/**************************************************************************/
/**************************************************************************/
/* Close the LDAP session */ /* */
/* */
ldap_unbind(ld);

return 0;

Related Information

- "ldap_free_sort_keylist()—Free all Memory used by the Sort Key List" on page 85 — Free all memory used by the sort key list.
- "ldap_create_sort_control()—Create a Sorted Results Control used when Sorting Search Results" on page 49 — Create a sorted results control.
- "ldap_parse_sort_control()—Retrieve Values in a Sorted Results Control" on page 148 — Retrieve values in a sorted results control.
- "ldap_create_page_control()—Create a Paged Results Control used when paging search results" on page 42 — Create a paged results control.
- "ldap_parse_page_control()—Retrieve Values in a Paged Results Control" on page 139 — Retrieve values in a paged results control.
- "ldap_search()—Perform an LDAP Search Operation" on page 164 — Asynchronously search the directory.
- "ldap_parse_result()—Extract Information from Results" on page 144 — Extract information from results.

API introduced: V5R3

ldap_default_dn_get() — Retrieve the User’s Default DN

Syntax

#include <ldap.h>

int ldap_default_dn_get(
    char **default_dn,
    char *filename
)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafes: Yes

The ldap_default_dn_get() API is used to retrieve the user’s default DN. To free the returned string, use "ldap_memfree()—Free Memory Allocated by LDAP API” on page 110.

An application stores the default DN on disk by calling "ldap_default_dn_set()—Store the User’s Default DN” on page 57. For i5/OS the default file (used when filename is NULL) where the default DN stored is called ldap_user_info and will be found in the user’s home directory. A user’s home directory is specified in the user’s profile.

Authorities and Locks

The caller must have Execute (*X) authority to each directory in the path name preceding the name of the user information file. The caller must have Read (*R) authority to the user information file.
Parameters

default_dn
(output) Specifies the user’s default Distinguished Name. Free *default_dn with
“ldap_memfree()—Free Memory Allocated by LDAP API” on page 110 when no longer needed.

filename
(Input) Specifies an alternative location for the user’s default Distinguished Name storage. If only
a filename is given for the filename parameter then the file will be checked in the current
directory, otherwise, if a path is given as well as a filename as part of the filename parameter, the
file will be checked following the given path. If filename is NULL, a file called ldap_user_info in
the user’s home directory will be read.

Return Value

LDAP_SUCCESS
if the default DN was retrieved.

another LDAP error code
if the request was not successful.

Error Conditions

The ldap_default_dn_get() API will return an LDAP error code if not successful. See “LDAP Client API
Error Conditions” on page 300 for possible LDAP error code values.

Error Messages

The following message may be sent from this function.

Message ID   Error Message Text
CPF3CF2 E     Error(s) occurred during running of ldap_default_dn_get API.

Related Information

• “ldap_default_dn_set()— Store the User’s Default DN” — Store the User’s Default DN.
• “ldap_enetwork_domain_set()— Store the User’s Default eNetwork Domain Name” on page 66 —
  Store the User’s Default eNetwork Domain Name.
• “ldap_enetwork_domain_get()— Retrieve the User’s Default eNetwork Domain Name” on page 65 —
  Retrieve the User’s Default eNetwork Domain Name.
• “ldap_memfree()—Free Memory Allocated by LDAP API” on page 110 — Free Memory Allocated by
  LDAP API

API introduced: V4R5

Related Information

ldap_default_dn_set()— Store the User’s Default DN

Syntax

#include <ldap.h>

int ldap_default_dn_set(
    char *default_dn,
    char *filename)

Lightweight Directory Access Protocol (LDAP) APIs 57
The `ldap_default_dn_set()` API is used to store the user’s default DN. The DN can be obtained by calling `ldap_default_dn_get()— Retrieve the User’s Default DN” on page 56.

The default DN is stored on disk. For i5/OS the default file the information will be stored in will be called `ldap_user_info` and will be put into the user’s home directory. A user’s home directory is specified in the user’s profile. The home directory must be created prior to calling `ldap_default_dn_set()` and is not created as part of the creation of a user’s profile. It will be stored in the local character set format.

Authorities and Locks
The caller must have Execute (*X) authority to each directory in the path name preceding the name of the user information file. The caller must have Write (*W) authority to the user information file. If the `filename` file doesn’t exist in the directory when calling `ldap_default_dn_set`, the caller must have Write (*W) authority to the file’s parent directory.

Parameters

default_dn
   (input) Specifies the user’s default Distinguished Name.

filename
   (Input) Specifies an alternative location for the user’s default Distinguished Name storage. If only a filename is given for the `filename` parameter then a file will be created in the current directory, otherwise, if a path is given as well as a filename as part of the `filename` parameter, the file will be created following the given path. If `filename` is NULL, a file called `ldap_user_info` will be created into the user’s home directory.

Return Value

LDAP_SUCCESS
   if the request was successful.

another LDAP error code
   if the request was not successful.

Error Conditions

The `ldap_default_dn_set()` API will return an LDAP error code if not successful. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values.

Error Messages

The following message may be sent from this function.

Message ID   Error Message Text
CPF3CF2 E   Error(s) occurred during running of the ldap_default_dn_set API.

Related Information

• “ldap_default_dn_get()— Retrieve the User’s Default DN” on page 56 — Retrieve the User’s Default DN.

• “ldap_enetwork_domain_set()— Store the User’s Default eNetwork Domain Name” on page 66 — Store the User’s Default eNetwork Domain Name.
ldap_delete()—Perform an LDAP Delete Operation

Syntax
#include <ldap.h>

int ldap_delete(
    LDAP *id,
    const char *dn
)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The ldap_delete() routine initiates an asynchronous LDAP operation to delete a leaf entry. The result of the operation can be obtained by a subsequent call to ldap_result()—Retrieve Result of an Asynchronous LDAP Operation on page 156.

Note that the entry to delete must be a leaf entry (that is, it must have no children). Deletion of entire subtrees in a single operation is not supported by LDAP.

Authorities and Locks
No i5/OS authority is required. All authority checking is done by the LDAP server.

Parameters

id (Input) Specifies the LDAP pointer returned by a previous call to ldap_init()—Perform an LDAP Initialization Operation on page 105, ldap_ssl_init—Initialize an SSL Connection on page 204 or ldap_open()—Perform an LDAP Open Operation on page 135.

dn (Input) Specifies the DN of the entry to be deleted.

Return Value
Message ID of the Operation Initiated
If the request was successful.
-1 If the request was not successful.

Error Conditions
If ldap_delete() is not successful, ld_errno will be set to indicate the error. See LDAP Client API Error Conditions on page 300 for possible LDAP error code values. Use ldap_get_errno()—Retrieve Error Information on page 91 function to retrieve the error information.

Error Messages
The following message may be sent from this function.

Message ID   Error Message Text
CPF3CF2 E     Error(s) occurred during running of ldap_delete API.
Related Information

- “ldap_delete_s()—Perform an LDAP Delete Operation (Synchronous)” on page 63 — Synchronous delete an entry.
- “ldap_delete_ext()—Perform an LDAP Delete Operation with Controls” — Asynchronous delete an entry with controls.
- “ldap_delete_ext_s()—Perform an LDAP Delete Operation with Controls” on page 61 — Synchronous delete an entry with controls.

API introduced: V4R3

Top| "Lightweight Directory Access Protocol (LDAP) APIs,” on page 1| APIs by category

ldap_delete_ext()—Perform an LDAP Delete Operation with Controls

Syntax

```c
#include <ldap.h>

int ldap_delete_ext(LDAP *ld,
                    const char *dn,
                    LDAPControl **serverctrls,
                    LDAPControl **clientctrls,
                    int *msgidp)
```

Default Public Authority: *USE

Library Name/Service Program: QSYS/QGLDCLNT

Threadsafe: Yes

The `ldap_delete_ext()` routine initiates an asynchronous LDAP operation to delete a leaf entry with controls.

Note that the entry to delete must be a leaf entry (that is, it must have no children). Deletion of entire subtrees in a single operation is not supported by LDAP.

Authorities and Locks

No i5/OS authority is required. All authority checking is done by the LDAP server.

Parameters

- `ld` (Input) Specifies the LDAP pointer returned by a previous call to "ldap_init()—Perform an LDAP Initialization Operation” on page 105, "ldap_ssl_init —Initialize an SSL Connection,” on page 204, or “ldap_open()—Perform an LDAP Open Operation” on page 135.
- `dn` (Input) Specifies the Distinguished Name (DN) of the entry to be deleted.
- `serverctrls` (Input) Specifies a list of LDAP server controls. This parameter may be set to null. See LDAP Controls for more information about server controls.
- `clientctrls` (Input) Specifies a list of LDAP client controls. This parameter may be set to null. See LDAP Controls for more information about client controls.
- `msgidp` (Output) This result parameter is set to the message id of the request if the `ldap_delete_ext()` call succeeds.
Return Value

LDAP_SUCCESS
if the request was successfully sent, ldap_delete_ext() places the message id of the request in *msgidp. A subsequent call to "ldap_result()—Retrieve Result of an Asynchronous LDAP Operation" on page 156 can be used to obtain the result of the operation. Once the operation has completed, "ldap_result()—Retrieve Result of an Asynchronous LDAP Operation" on page 156 returns a result that contains the status of the operation (in the form of an error code). The error code indicates if the operation completed successfully.

another LDAP error code
if the request was not successfully.

Error Conditions

The ldap_delete_ext() API will return an LDAP error code if not successful. See "LDAP Client API Error Conditions" on page 300 for possible LDAP error code values.

Error Messages

The following message may be sent from this function.

Message ID   Error Message Text
CPF3CF2 E     Error(s) occurred during running of ldap_delete_ext API.

Related Information

• "ldap_delete()—Perform an LDAP Delete Operation" on page 59 — Asynchronous delete an entry.
• "ldap_delete_s()—Perform an LDAP Delete Operation (Synchronous)” on page 63 — Synchronous delete an entry.
• "ldap_delete_ext_s()—Perform an LDAP Delete Operation with Controls” — Synchronous delete an entry with controls.

The ldap_delete_ext() API supports LDAP V3 server controls and client controls.

API introduced: V4R5

ldap_delete_ext_s()—Perform an LDAP Delete Operation with Controls

Syntax

```
#include <ldap.h>

int ldap_delete_ext_s(LDAP *ld, const char *dn, LDAPControl **serverctrls, LDAPControl **clientctrls)
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The ldap_delete_ext_s() routine initiates a synchronous LDAP operation to delete a leaf entry with controls.
Note that the entry to delete must be a leaf entry (that is, it must have no children). Deletion of entire subtrees in a single operation is not supported by LDAP.

**Authorities and Locks**

No i5/OS authority is required. All authority checking is done by the LDAP server.

**Parameters**

- **ld**  
  (Input) Specifies the LDAP pointer returned by a previous call to “ldap_init()—Perform an LDAP Initialization Operation” on page 105, “ldap_ssl_init—Initialize an SSL Connection.” on page 204, or “ldap_open()—Perform an LDAP Open Operation” on page 135.

- **dn**  
  (Input) Specifies the Distinguished Name (DN) of the entry to be deleted.

- **serverctrls**  
  (Input) Specifies a list of LDAP server controls. This parameter may be set to null. See [LDAP Controls](#) for more information about server controls.

- **clientctrls**  
  (Input) Specifies a list of LDAP client controls. This parameter may be set to null. See [LDAP Controls](#) for more information about client controls.

**Return Value**

- **LDAP_SUCCESS**  
  if the request was successful.

- **another LDAP error code**  
  if the request was not successful.

**Error Conditions**

The `ldap_delete_ext_s()` API will return an LDAP error code if not successful. See ["LDAP Client API Error Conditions” on page 300](#) for possible LDAP error code values.

**Error Messages**

The following message may be sent from this function.

- **Message ID** CPF3CF2  
  **Error Message Text** Error(s) occurred during running of `ldap_delete_ext_s` API.

**Related Information**

- "ldap_delete()—Perform an LDAP Delete Operation” on page 59 — Asynchronous delete an entry.
- "ldap_delete_s()—Perform an LDAP Delete Operation (Synchronous)” on page 63 — Synchronous delete an entry.
- "ldap_delete_ext()—Perform an LDAP Delete Operation with Controls” on page 60 — Asynchronous delete an entry with controls.

The `ldap_delete_ext_s()` API supports LDAP V3 server controls and client controls.

API introduced: V4R5
ldap_delete_s()—Perform an LDAP Delete Operation (Synchronous)

Syntax
#include <ldap.h>

int ldap_delete_s(
    LDAP* ld,
    const char* dn)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threading: Yes

The ldap_delete_s() routine initiates a synchronous LDAP operation to delete a leaf entry.

Note that the entry to delete must be a leaf entry (that is, it must have no children). Deletion of entire subtrees in a single operation is not supported by LDAP.

Authorities and Locks
No i5/OS authority is required. All authority checking is done by the LDAP server.

Parameters

ld    (Input) Specifies the LDAP pointer returned by a previous call to "ldap_init()—Perform an LDAP Initialization Operation" on page 105, "ldap_ssl_init—Initialize an SSL Connection," on page 204, or "ldap_open()—Perform an LDAP Open Operation" on page 135.

dn    (Input) Specifies the Distinguished Name (DN) of the entry to be deleted.

Return Value

LDAP_SUCCESS if the request was successful.

another LDAP error code if the request was not successful.

Error Conditions

The ldap_delete_s() will return an LDAP error code if not successful. See "LDAP Client API Error Conditions" on page 300 for possible LDAP error code values.

Error Messages

The following message may be sent from this function.

Message ID    Error Message Text
CPF3CF2 E      Error(s) occurred during running of ldap_delete_s API.

Related Information

• "ldap_delete()—Perform an LDAP Delete Operation" on page 59 — Asynchronous delete an entry.
• "ldap_delete_ext()—Perform an LDAP Delete Operation with Controls" on page 60 — Asynchronous delete an entry with controls.
• "ldap_delete_ext_s()—Perform an LDAP Delete Operation with Controls" on page 61 — Synchronous delete an entry with controls.
The `ldap_dn2ufn()` function takes a distinguished name (DN) and converts it into a "friendlier" representation by removing the attribute type that is associated with each relative distinguished name (RDN). For example, the DN "cn=John Doe,ou=Widget Division,ou=Austin,o=IBM,c=US" would be returned in its "friendlier" form as "John Doe, Widget Division, Austin, IBM, US". Space for the user-friendly name will have been obtained by the API and should be freed by the caller with a call to "ldap_memfree()—Free Memory Allocated by LDAP API" on page 110.

### Authorities and Locks

No i5/OS authority is required.

### Parameters

- **dn** *(Input)* Specifies the DN to be converted (as returned from "ldap_get_dn()—Retrieve the Distinguished Name of an Entry" on page 89).

### Return Value

- **Character String** if the request was successful.
- **NULL** if the request was not successful.

### Error Conditions

If `ldap_dn2ufn()` is not successful, then there was no memory available for the character string.

### Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_dn2ufn API.</td>
</tr>
</tbody>
</table>

### Related Information

- "ldap_get_dn()—Retrieve the Distinguished Name of an Entry" on page 89 — Extract the DN from an entry.
ldap_explode_rdn()—Break a Relative Distinguished Name into Its Components—Break a Relative Distinguished Name into Its Components.

API introduced: V5R1

ldap_enetwork_domain_get()—Retrieve the User’s Default eNetwork Domain Name

Syntax
#include <ldap.h>

int ldap_enetwork_domain_get(
    char **edomain,
    char *filename)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The ldap_enetwork_domain_get() API is used to retrieve the user’s default eNetwork domain name. To free the returned string, use “ldap_memfree()—Free Memory Allocated by LDAP API” on page 110.

The eNetwork domain name (along with the user’s default Domain Name Service (DNS) domain name) is used to identify the user’s LDAP authentication domain. For example, if a user’s eNetwork domain name is "chicago", and the user’s DNS domain is midwest.illinois.com, then information can be published in DNS that associates ldap.chicago.midwest.illinois.com with a collection of LDAP servers (master(s) and replicas). This permits applications to easily find an appropriate LDAP authentication server, by using the “ldap_server_locate()—Locate Suitable LDAP Servers” on page 178 API.

An application stores the eNetwork domain name on disk by calling “ldap_enetwork_domain_set()—Store the User’s Default eNetwork Domain Name” on page 66. For i5/OS the default file where the eNetwork domain name stored is called ldap_user_info and will be found in the user’s home directory. A user’s home directory is specified in the user’s profile.

Authorities and Locks
The caller must have Execute (*X) authority to each directory in the path name preceding the name of the user information file. The caller must have Read (*R) authority to the user information file.

Parameters

edomain
(Output) Specifies the name of the eNetwork domain to which the user belongs.

filename
(Input) Specifies an alternative location for the user’s default eNetwork domain name. If only a filename is given for the filename parameter then the file will be found in the current directory, otherwise, if a path is given as well as a filename as part of the filename parameter, the file will be found by following the given path.

Return Value

LDAP_SUCCESS
if the request was successful.
another LDAP error code
    if the request was not successful.

**Error Conditions**
The `ldap_enetwork_domain_get()` will return an LDAP error code if not successful. See "LDAP Client API Error Conditions" on page 300 for possible LDAP error code values.

**Error Messages**
The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of <code>ldap_enetwork_domain_get</code> API.</td>
</tr>
</tbody>
</table>

**Related Information**
- "ldap_default_dn_set()— Store the User’s Default DN” on page 57 — Store the User’s Default DN.
- "ldap_default_dn_get()— Retrieve the User’s Default DN” on page 56 — Retrieve the User’s Default DN.
- "ldap_enetwork_domain_set()— Store the User’s Default eNetwork Domain Name” — Store the User’s Default eNetwork Domain Name.

API introduced: V4R5

**ldap_enetwork_domain_set()— Store the User’s Default eNetwork Domain Name**

**Syntax**
```
#include <ldap.h>

int ldap_enetwork_domain_set(
    char *edomain,
    char *filename)
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threading Safe: Yes

The `ldap_enetwork_domain_set()` API is used to store the user’s default eNetwork domain name (specified as a NULL terminated string).

The eNetwork domain name (along with the user’s default Domain Name Service (DNS) domain name) is used to identify the user’s LDAP authentication domain. For example, if a user’s eNetwork domain name is “chicago”, and the user’s DNS domain is midwest.illinois.com, then information can be published in DNS that associates ldap.chicago.midwest.illinois.com with a collection of LDAP servers (master(s) and replicas). This permits applications to easily find an appropriate LDAP authentication server, by using the "ldap_server_locate()— Locate Suitable LDAP Servers” on page 178 API.

An application can retrieve the eNetwork domain name by calling "ldap_enetwork_domain_get()— Retrieve the User’s Default eNetwork Domain Name” on page 65.
The eNetwork domain name is stored on disk. For i5/OS the default file the information will be stored in will be called `ldap_user_info` and will be put into the user’s home directory. A user’s home directory is specified in the user’s profile. The home directory must be created prior to calling `ldap_enetwork_domain_set()` and is not created as part of the creation of a user’s profile. It will be stored in the local character set format.

**Authorities and Locks**

The caller must have Execute (*X) authority to each directory in the path name preceding the name of the user information file. The caller must have Write (*W) authority to the user information file. If the file doesn’t exist in the directory, the caller must have Write (*W) authority to the file’s parent directory.

**Parameters**

```plaintext
edomain
(Input) Specifies the name of the eNetwork domain to which the user belongs.
```

```plaintext
filename
(Input) Specifies an alternative location for the user’s default eNetwork domain name. If only a filename is given for the `filename` parameter then a file will be created in the current directory, otherwise, if a path is given as well as a filename as part of the `filename` parameter, the file will be created following the given path.
```

**Return Value**

- LDAP_SUCCESS if the request was successful.
- another LDAP error code if the request was not successful.

**Error Conditions**

The `ldap_enetwork_domain_set()` API will return an LDAP error code if not successful. See "LDAP Client API Error Conditions” on page 300 for possible LDAP error code values.

**Error Messages**

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_enetwork_domain_set API.</td>
</tr>
</tbody>
</table>

**Related Information**

- "ldap_default_dn_set()— Store the User’s Default DN” on page 57 — Store the User’s Default DN.
- "ldap_default_dn_get()— Retrieve the User’s Default DN” on page 56 — Retrieve the User’s Default DN.
- "ldap_enetwork_domain_get()— Retrieve the User’s Default eNetwork Domain Name” on page 65 — Retrieve the User’s Default eNetwork Domain Name.

API introduced: V4R5
ldap_err2string()—Retrieve LDAP Error Message String

Syntax
#include <ldap.h>

char *ldap_err2string(
    int error)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The ldap_err2string() function is used to retrieve the text description corresponding to an LDAP error code.

The text description returned will be provided in English only.

The string returned from ldap_err2string() should not be freed when use of the string is complete.

Authorities and Locks
No i5/OS authority is required.

Parameters
error  (Input) Specifies the LDAP error code returned by a previous call to

“ldap_result2error()—Retrieve LDAP Error Information” on page 158, “ldap_get_errno()—Retrieve Error Information” on page 91, or a synchronous LDAP API.

Return Value
LDAP error description String
a textual description of the LDAP error code.

Error Conditions
The ldap_err2string() API will return “Unknown Error” if the LDAP error code is unknown. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error codes and their description.

Error Messages
The following message may be sent from this function.

Message ID   Error Message Text
CPF3CF2 E    Error(s) occurred during running of ldap_err2string API.

Related Information
• “ldap_get_errno()—Retrieve Error Information” on page 91 — Retrieve Error Code set.
• “ldap_perror()—Print LDAP Error Information” on page 149 — Print an LDAP error indication to standard error.
• “ldap_result2error()—Retrieve LDAP Error Information” on page 158 — Extract LDAP error indication from LDAP result.
**ldap_explode_dn()—Break a Distinguished Name into Its Components**

**Syntax**

```c
#include <ldap.h>

char **ldap_explode_dn(
    const char *dn,
    int notypes)
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The `ldap_explode_dn()` function uses the distinguished name in local codepage returned by "ldap_get_dn()—Retrieve the Distinguished Name of an Entry" on page 89 and breaks it up into its component parts. Each part is known as a Relative Distinguished Name (RDN). If the dn is in UTF8, use "ldap_explode_dn_utf8()—Break a UTF8 codepage Distinguished Name into Its Components" on page 71.

`ldap_explode_dn()` returns a NULL-terminated array, each component of which contains an RDN from the DN. The `notypes` parameter is used to request that only the RDN values be returned, not their types.

For example, the distinguished name `cn=Bob,c=US` would return as either "cn=Bob","c=US",NULL or "Bob","US", NULL depending on whether `notypes` was 0 or 1, respectively. The result can be freed by calling "ldap_value_free()—Free Memory Allocated by ldap_get_values()" on page 223.

**Authorities and Locks**

No i5/OS authority is required.

**Parameters**

- **dn** (Input) Specifies the DN to be exploded (as returned from "ldap_get_dn()—Retrieve the Distinguished Name of an Entry" on page 89).
- **notypes** (Input) Specifies if type information is to be returned with each RDN. If non-zero, the type information will be stripped. If zero, the type information is retained. For example, setting `notypes` to 1 would result in the RDN "cn=Fido" being returned as "Fido".

**Return Value**

- Relative Distinguished Name (RDN) if the request was successful.
- NULL if the request was not successful.

**Error Conditions**

If `ldap_explode_dn()` is not successful, then there was no memory available for either the array or its component parts.

**Error Messages**

The following message may be sent from this function.
Message ID | Error Message Text
---|---
CPF3CF2 E | Error(s) occurred during running of ldap_explode_dn API.

### Related Information

- "ldap_get_dn()—Retrieve the Distinguished Name of an Entry" on page 89 — Extract the DN from an entry.
- "ldap_explode_dn_utf8()—Break a UTF8 codepage Distinguished Name into Its Components" on page 74 — Break a UTF8 Distinguished Name into its components.
- "ldap_explode_rdn()—Break a Relative Distinguished Name into Its Components" on page 72 — Break a Relative Distinguished Name into its components.

API introduced: V4R3

---

**ldap_explode_dns()—Break a DNS-style Distinguished Name into Its Components**

**Syntax**

```c
#include <ldap.h>

char ***ldap_explode_dns(
    const char *dn)
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The `ldap_explode_dns()` function takes a Domain Name System (DNS)-style distinguished name and breaks it up into its component parts.

`ldap_explode_dns()` returns a NULL-terminated array of character strings.

For example, the DNS-style distinguished name rochester.ibm.com would be returned as an array of components "rochester", "ibm", "com", NULL. The result can be freed by calling `ldap_value_free()`—Free Memory Allocated by `ldap_get_values()`" on page 223.

**Authorities and Locks**

No i5/OS authority is required.

**Parameters**

- `dn` (Input) Specifies the DNS-style DN to be exploded.

**Return Value**

- An array of character strings. if the request was successful.
- NULL if the request was not successful.
Error Conditions
If `ldap_explode_dns()` is not successful, no memory is available for the array or its components.

Error Messages
The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_explode_dns API.</td>
</tr>
</tbody>
</table>

Related Information
- “ldap_explode_dn()—Break a Distinguished Name into Its Components” on page 69 — Break a Distinguished Name into its components.
- “ldap_explode_dn_utf8()—Break a UTF8 codepage Distinguished Name into Its Components” — Break a UTF8 codepage Distinguished Name into its components.
- “ldap_explode_rdn()—Break a Relative Distinguished Name into Its Components” on page 72 — Break a Relative Distinguished Name into its components.
- “ldap_explode_rdn_utf8()—Break a UTF8 codepage Relative Distinguished Name into Its Components” on page 74 — Break a UTF8 codepage Relative Distinguished Name into Its Components.

API introduced: V5R1

```c
#include <ldap.h>

char **ldap_explode_dn_utf8(
    char *dn,
    int notypes
);```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The `ldap_explode_dn_utf8` function uses the distinguished name in UTF8 characters returned by “ldap_get_dn()—Retrieve the Distinguished Name of an Entry” on page 89 and breaks it up into its component parts. Each part is known as a Relative Distinguished Name (RDN). If the dn is in local codepage, use “ldap_explode_dn()—Break a Distinguished Name into Its Components” on page 69.

`ldap_explode_dn_utf8()` returns a NULL-terminated array, each component of which contains an RDN from the DN. The `notypes` parameter is used to request that only the RDN values be returned, not their types.

For example, the distinguished name cn=Bob,c=US would return as either "cn=Bob","c=US",NULL or "Bob","US", NULL depending on whether `notypes` was 0 or 1, respectively. The result can be freed by calling “ldap_value_free()—Free Memory Allocated by ldap_get_values()” on page 223.
Authorities and Locks
No i5/OS authority is required.

Parameters

\texttt{dn} (Input) The DN to be exploded in UTF8 codepage (as returned from \texttt{ldap_get_dn()—Retrieve the Distinguished Name of an Entry} on page 89).

\texttt{notypes} (Input) Whether type information is to be returned with each RDN. If non-zero, the type information is stripped. If zero, the type information is retained. For example, setting \texttt{notypes} to 1 would result in the RDN "cn=Fido" being returned as "Fido".

Return Value
Relative Distinguished Name (RDN)
The request was successful.

\texttt{NULL} The request was not successful. The \texttt{ldap_get_errno()—Retrieve Error Information} on page 91 API can be used to obtain the error code.

Error Conditions
If \texttt{ldap_explode_dn_utf8} is not successful, \texttt{ld_errno} is set to indicate the error. See \texttt{"LDAP Client API Error Conditions"} on page 300 for possible LDAP error code values. Use the \texttt{ldap_get_errno()—Retrieve Error Information} on page 91 function to retrieve the error information.

Error Messages
The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of \texttt{ldap_explode_dn_utf8} API.</td>
</tr>
</tbody>
</table>

Related Information

- \texttt{"ldap_explode_dn()—Break a Distinguished Name into Its Components"} on page 69 — Break a Distinguished Name into Its Components.
- \texttt{"ldap_explode_rdn()—Break a Relative Distinguished Name into Its Components"} — Break a Relative Distinguished Name into Its Components.
- \texttt{"ldap_explode_rdn_utf8()—Break a UTF8 codepage Relative Distinguished Name into Its Components"} on page 74 — Break a UTF8 codepage Relative Distinguished Name into Its Components.

API introduced: V5R1
The `ldap_explode_rdn()` function uses the relative distinguished name (RDN) in the local CCSID (as returned by `ldap_explode_dn()`—Break a Distinguished Name into Its Components” on page 69, for example) and breaks it up into its component parts. If the RDN is in UTF8, use “ldap_explode_rdn_utf8()—Break a UTF8 codepage Relative Distinguished Name into Its Components” on page 74.

`ldap_explode_rdn()` returns a NULL-terminated array of character strings. The `notypes` parameter is used to request that only the component values be returned, not their types.

For example, the RDN "ou=Research+cn=Bob" would return as either ["ou=Research", "cn=Bob", NULL] or ["Research", "Bob", NULL], depending on whether `notypes` was 0 or 1, respectively. The result can be freed by calling “ldap_value_free()—Free Memory Allocated by ldap_get_values()” on page 223.

**Authorities and Locks**

No i5/OS authority is required.

**Parameters**

- **rdn**  
  (Input) Specifies the RDN to be exploded (perhaps as returned by `ldap_explode_dn()`—Break a Distinguished Name into Its Components” on page 69). Multiple RDNs can be concatenated using a plus sign (‘+’).

- **notypes**  
  (Input) Specifies if type information is to be returned with each RDN. If non-zero, the type information will be stripped. If zero, the type information is retained. For example, setting `notypes` to 1 would result in the RDN "cn=Fido" being returned as "Fido".

**Return Value**

- Components of Relative Distinguished Name (RDN) if the request was successful.
- NULL if the request was not successful.

**Error Conditions**

If `ldap_explode_rdn()` is not successful, then there was no memory available for either the array or its component parts.

**Error Messages**

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of <code>ldap_explode_rdn</code> API.</td>
</tr>
</tbody>
</table>

**Related Information**

- “ldap_explode_dn()—Break a Distinguished Name into Its Components” on page 69 — Break a Distinguished Name into its components.
- “ldap_explode_rdn_utf8()—Break a UTF8 codepage Relative Distinguished Name into Its Components” on page 74 — Break a UTF8 Relative Distinguished Name into its components.
ldap_explode_rdn_utf8()—Break a UTF8 codepage Relative Distinguished Name into Its Components

Syntax

#include <ldap.h>

char **ldap_explode_rdn_utf8(
    char *rdn,
    int notypes)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The ldap_explode_rdn_utf8() function uses the relative distinguished name (RDN) in UTF8 characters (as returned by ldap_explode_dn_utf8()—Break a UTF8 codepage Distinguished Name into Its Components” on page 71 for example) and breaks it up into its component parts. If the RDN is in local codepage, use ldap_explode_rdn()—Break a Relative Distinguished Name into Its Components” on page 72.

ldap_explode_rdn_utf8() returns a NULL-terminated array of character strings. The notypes parameter is used to request that only the component values be returned, not their types.

For example, the RDN "ou=Research+cn=Bob" would return as either {"ou=Research", "cn=Bob", NULL} or{"Research","Bob", NULL}, depending on whether notypes was 0 or 1, respectively. The result can be freed by calling ldap_value_free()—Free Memory Allocated by ldap_get_values()” on page 223.

Authorities and Locks

No i5/OS authority is required.

Parameters

rdn (Input) The RDN to be exploded (perhaps as returned by ldap_explode_dn_utf8()—Break a UTF8 codepage Distinguished Name into Its Components” on page 71). Multiple RDNs can be concatenated using a plus sign (+).

notypes (Input) Whether type information is to be returned with each RDN. If non-zero, the type information is stripped. If zero, the type information is retained. For example, setting notypes to 1 would result in the RDN "cn=Fido" being returned as "Fido".

Return Value

Components of Relative Distinguished Name (RDN)
The request was successful.

NULL The request was not successful. The ldap_get_errno()—Retrieve Error Information” on page 91 API can be used to obtain the error code.
Error Conditions

If `ldap_explode_rdn_utf8()` is not successful, `ld_errno` will be set to indicate the error. See "LDAP Client API Error Conditions" on page 300 for possible LDAP error code values. Use the "ldap_get_errno()—Retrieve Error Information" on page 91 function to retrieve the error information.

Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_explode_rdn_utf8 API.</td>
</tr>
</tbody>
</table>

Related Information

- "ldap_explode_dn()—Break a Distinguished Name into Its Components" on page 69 — Break a Distinguished Name into Its Components.
- "ldap_explode_dn_utf8()—Break a UTF8 codepage Distinguished Name into Its Components" on page 71 — Break a UTF8 codepage Distinguished Name into Its Components.
- "ldap_explode_rdn()—Break a Relative Distinguished Name into Its Components" on page 72 — Break a Distinguished Name into Its Components.

API introduced: V5R1
Authorities and Locks
No i5/OS authority is required. All authority checking is done by the LDAP server.

Parameters

ld  (Input) Specifies the LDAP pointer returned by a previous call to "ldap_init()—Perform an LDAP Initialization Operation” on page 105, "ldap_ssl_init —Initialize an SSL Connection.” on page 204, or “ldap_open()—Perform an LDAP Open Operation” on page 135.

reqoid (Input) Specifies the dotted-OID text string that identifies the extended operation to be performed by the server.

reqdata (Input) Specifies the arbitrary data required by the extended operation (if NULL, no data is sent to the server).

serverctrls  (Input) Specifies a list of LDAP server controls. This parameter may be set to null. See LDAP Controls for more information about server controls.

clientctrls  (Input) Specifies a list of LDAP client controls. This parameter may be set to null. See LDAP Controls for more information about client controls.

msgidp  (Output) This result parameter is set to the message id of the request if the ldap_extended_operation() call succeeds.

Return Value

LDAP_SUCCESS if the request was successful. ldap_extended_operation() places the message id of the request in *msgidp. To check the result of this operation, call "ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156 and "ldap_parse_extended_result()—Parse extended result” on page 138 APIs. The server may also return an OID and result data.

another LDAP error code if the request was not successful.

Error Conditions
If ldap_extended_operation() is not successful, will return a -1 instead of a valid msgid, setting the session error in the LD structure, which can be obtained by using "ldap_get_errno()—Retrieve Error Information” on page 91.

Error Messages
The following message may be sent from this function.

Message ID   Error Message Text
CPF3CF2 E    Error(s) occurred during running of ldap_extended_operation API.
Related Information

- "ldap_add_ext()—Perform an LDAP Add Operation with Controls" on page 11 — Asynchronously add an entry with controls.
- "ldap_add_ext_s()—Perform an LDAP Add Operation with Controls (Synchronous)” on page 13 — Synchronously add an entry with controls.
- "ldap_compare_ext()—Perform an LDAP Compare Operation with Controls” on page 28 — Asynchronous compare to a directory entry with controls.
- "ldap_compare_ext_s()—Perform an LDAP Compare Operation with Controls (Synchronous)” on page 30 — Synchronous compare to a directory entry with controls.
- "ldap_delete_ext()—Perform an LDAP Delete Operation with Controls” on page 60 — Asynchronous delete an entry with controls.
- "ldap_delete_ext_s()—Perform an LDAP Delete Operation with Controls” on page 61 — Synchronous delete an entry with controls.
- "ldap_modify_ext()—Perform an LDAP Modify Entry Request with Controls” on page 113 — Asynchronously modify an entry with controls.
- "ldap_modify_ext_s()—Perform an LDAP Modify Entry Request with Controls” on page 116 — Synchronously modify an entry with controls.
- "ldap_parse_extended_result()—Parse extended result” on page 138 — Parse extended result.
- "ldap_sasl_bind()—Perform an LDAP SASL Bind Request” on page 159 — Asynchronously bind to the directory using the Simple Authentication Security Layer (SASL).
- "ldap_sasl_bind_s()—Perform an LDAP SASL Bind Request (Synchronous)” on page 161 — Synchronously bind to the directory using the Simple Authentication Security Layer (SASL).
- "ldap_search_ext —Asynchronously Search the Directory Using Controls” on page 166 — Asynchronously search the directory with controls.
- "ldap_search_ext_s — Synchronously Search the Directory Using Controls” on page 168 — Synchronously search the directory with controls.
- "ldap_rename()—Asynchronously Rename an Entry” on page 152 — Asynchronously rename an entry with controls.
- "ldap_rename_s()—Synchronously Rename an Entry” on page 154 — Synchronously rename an entry with controls.
- "ldap_unbind_ext()—Perform an LDAP Unbind Request” on page 211 — Unbind with controls.

The ldap_extended_operation() API supports LDAP V3 server controls and client controls.

API introduced: V5R1

#include <ldap.h>

int ldap_extended_operation_s(
  LDAP *ld,
  const char *reqoid,
  const struct berval *reqdata,
  LDAPControl **serverctrls,
  LDAPControl **clientctrls,
  char **retoidp,
  struct berval **retdatap)

Lightweight Directory Access Protocol (LDAP) APIs 77
The `ldap_extended_operation_s()` function is used to perform a synchronous LDAP extended operation, which returns LDAP_SUCCESS if the extended operation completed successfully, or an LDAP error code if not. The `retoid` and `retdata` parameters are filled in with the Object IDentifier (OID) and data from the response. If no OID or data was returned, these parameters are set to NULL, respectively.

If the LDAP server does not support the extended operation, the operation will fail. To determine if the requisite extended operation is supported by the server, get the rootDSE of the LDAP server and check for the supportedExtension attribute. If the values for this attribute include the object identifier of your extended operation, then the server supports the extended operation. If the supportedExtension attribute is not present in the rootDSE, then the server is not configured to support any extended operations.

** Authorities and Locks  

No i5/OS authority is required. All authority checking is done by the LDAP server.

** Parameters  

* `ld` (Input) Specifies the LDAP pointer returned by a previous call to "ldap_init()—Perform an LDAP Initialization Operation” on page 105, "ldap_ssl_init—Initialize an SSL Connection.” on page 204, or "ldap_open()—Perform an LDAP Open Operation” on page 135.

* `reqoid` (Input) Specifies the dotted-OID text string that identifies the extended operation to be performed by the server.

* `reqdata` (Input) Specifies the arbitrary data required by the extended operation (if NULL, no data is sent to the server).

* `serverctrls` (Input) Specifies a list of LDAP server controls. This parameter may be set to null. See LDAP Controls for more information about server controls.

* `clientctrls` (Input) Specifies a list of LDAP client controls. This parameter may be set to null. See LDAP Controls for more information about client controls.

* `retoidp` (Output) This result parameter is set to point to a character string that is set to an allocated, dotted-OID text string returned from the server. This string should be disposed of using the "ldap_memfree()—Free Memory Allocated by LDAP API” on page 110 API. If no OID is returned, `retoidp` is set to NULL.

* `retdatp` (Output) This result parameter is set to a pointer to a berval structure pointer that is set to an allocated copy of the data returned by the server. This struct berval should be disposed of using `ber_bvfree()`. If no data is returned, `retdatp` is set to NULL.

** Return Value  

* LDAP_SUCCESS  

  if the request was successful.

* another LDAP error code  

  if the request was not successful.

** Error Conditions  

If `ldap_extended_operation_s()` is not successful, it will return the LDAP error code resulting from the operation.
Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2</td>
<td>Error(s) occurred during running of ldap_extended_operation_s API.</td>
</tr>
</tbody>
</table>

Related Information

- "ldap_add_ext()—Perform an LDAP Add Operation with Controls” on page 11 — Asynchronously add an entry with controls.
- "ldap_add_ext_s()—Perform an LDAP Add Operation with Controls (Synchronous)” on page 13 — Synchronously add an entry with controls.
- "ldap_compare_ext()—Perform an LDAP Compare Operation with Controls” on page 28 — Asynchronous compare to a directory entry with controls.
- "ldap_compare_ext_s()—Perform an LDAP Compare Operation with Controls (Synchronous)” on page 30 — Synchronous compare to a directory entry with controls.
- "ldap_delete_ext()—Perform an LDAP Delete Operation with Controls” on page 60 — Asynchronous delete an entry with controls.
- "ldap_delete_ext_s()—Perform an LDAP Delete Operation with Controls” on page 61 — Synchronous delete an entry with controls.
- "ldap_modify_ext()—Perform an LDAP Modify Entry Request with Controls” on page 113 — Asynchronously modify an entry with controls.
- "ldap_modify_ext_s()—Perform an LDAP Modify Entry Request with Controls” on page 116 — Synchronously modify an entry with controls.
- "ldap_sasl_bind()—Perform an LDAP SASL Bind Request” on page 159 — Asynchronously bind to the directory using the Simple Authentication Security Layer (SASL).
- "ldap_sasl_bind_s()—Perform an LDAP SASL Bind Request (Synchronous)” on page 161 — Synchronously bind to the directory using the Simple Authentication Security Layer (SASL).
- "ldap_search_ext — Asynchronously Search the Directory Using Controls” on page 166 — Asynchronously search the directory with controls.
- "ldap_search_ext_s — Synchronously Search the Directory Using Controls” on page 168 — Synchronously search the directory with controls.
- "ldap_rename()—Asynchronously Rename an Entry” on page 152 — Asynchronously rename an entry with controls.
- "ldap_rename_s()—Synchronously Rename an Entry” on page 154 — Synchronously rename an entry with controls.
- "ldap_unbind_ext()—Perform an LDAP Unbind Request” on page 211 — Unbind with controls.

The ldap_extended_operation_s() API supports LDAP V3 server controls and client controls.

API introduced: V5R1

Syntax

Lightweight Directory Access Protocol (LDAP) APIs 79
#include <ldap.h>

char *ldap_first_attribute(
  LDAP  *ld,
  LDAPMessage  *entry,
  BerElement  **berptr)

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafes: Yes

The `ldap_first_attribute()` function returns the first attribute in an entry. The `ldap_first_attribute()` and 
“`ldap_next_attribute()—Retrieve Next Attribute in an Entry” on page 129” functions are used to step through the attributes in an LDAP entry.

`ldap_first_attribute()` takes an entry returned by “`ldap_first_entry()—Retrieve First LDAP Entry” on page 81” or “`ldap_next_entry()—Retrieve Next LDAP Entry” on page 131 and returns a pointer to a buffer containing a null terminated string that is the first attribute type in the entry. This buffer must be freed when its use is completed using “`ldap_memfree()—Free Memory Allocated by LDAP API” on page 110. 
*berptr also must be freed when its use is completed using “`ldap_ber_free()—Free storage allocated by the LDAP library” on page 22.”

**Authorities and Locks**
No i5/OS authority is required.

**Parameters**

`ld`  (Input) The LDAP pointer returned by a previous call to “`ldap_init()—Perform an LDAP Initialization Operation” on page 105.”

“`ldap_ssl_init —Initialize an SSL Connection.” on page 204. 
“`ldap_app_ssl_init_np —Initialize an SSL Connection’ on page 18 or “`ldap_open()—Perform an LDAP Open Operation” on page 135.

`entry`  (Input) The attribute information as returned by “`ldap_first_entry()—Retrieve First LDAP Entry” on page 81” or “`ldap_next_entry()—Retrieve Next LDAP Entry” on page 131;

`berptr`  (Output) A pointer to a BerElement that will be allocated to keep track of the current position. It is an input and output parameter for subsequent calls to “`ldap_next_attribute()—Retrieve Next Attribute in an Entry” on page 129.” The BerElement structure is opaque to the application. Free *berptr when its use is completed using ber_free.

**Return Value**

Pointer to a buffer containing the first attribute type in the entry
  if the request was successful.

NULL if the request was not successful.

**Error Conditions**

If `ldap_first_attribute()` is not successful, NULL is returned, and `ld_errno` will be set to indicate the error. See “`LDAP Client API Error Conditions” on page 300 for possible LDAP error code values. Use “`ldap_get_errno()—Retrieve Error Information” on page 91” function to retrieve the error information.

**Error Messages**

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_first_attribute API.</td>
</tr>
</tbody>
</table>
**Related Information**

- "ldap_first_entry()—Retrieve First LDAP Entry" — Retrieve first LDAP entry.
- "ldap_next_entry()—Retrieve Next LDAP Entry" on page 131 — Retrieve next LDAP entry.
- "ldap_count_attributes()—Retrieve Count of Attributes for an LDAP Entry" on page 35 — Retrieve count of attributes for an LDAP entry.
- "ldap_next_attribute()—Retrieve Next Attribute in an Entry" on page 129 — Return next attribute name in an entry.
- "ldap_get_values()—Retrieve a Set of Attribute Values from an Entry" on page 102 — Retrieve a set of attribute values from an entry.
- "ldap_get_values_len()—Retrieve a Set of Binary Attribute Values" on page 104 — Retrieve a set of binary attribute values.

API introduced: V4R3

---

**ldap_first_entry()—Retrieve First LDAP Entry**

**Syntax**

```c
#include <ldap.h>

LDAPMessage *ldap_first_entry(
    LDAP *ld,
    LDAPMessage *result)
```

Default Public Authority: *USE

Library Name/Service Program: QSYS/QGLDCLNT

Threadsafe: Yes

The `ldap_first_entry()` function takes the result from a call to "ldap_result()—Retrieve Result of an Asynchronous LDAP Operation" on page 156, "ldap_search_s()—Perform an LDAP Search Operation (Synchronous)" on page 171, or "ldap_search_st()—Perform an LDAP Search Operation (Timed Synchronous)" on page 173 and returns a pointer to the first entry in the result.

The `ldap_first_entry()`, "ldap_next_entry()—Retrieve Next LDAP Entry" on page 131, and "ldap_count_entries()—Retrieve Count of LDAP Entries" on page 36 functions are used to parse results received from "ldap_result()—Retrieve Result of an Asynchronous LDAP Operation" on page 156 or the synchronous LDAP search functions "ldap_search_s()—Perform an LDAP Search Operation (Synchronous)" on page 171 and "ldap_search_st()—Perform an LDAP Search Operation (Timed Synchronous)" on page 173.

---

**Authorities and Locks**

No i5/OS authority is required.

**Parameters**

- **Id** (Input) The LDAP pointer returned by a previous call to "ldap_init()—Perform an LDAP Initialization Operation” on page 105, "ldap_ssl_init —Initialize an SSL Connection.” on page 204, or "ldap_open()—Perform an LDAP Open Operation” on page 135.
- **result** (Input) The result returned by a call to “ldap_result()—Retrieve Result of an Asynchronous LDAP Operation.”
Return Value

Pointer to the next entry in the result

if the request was successful.

NULL if the request was not successful.

Error Conditions

If `ldap_first_entry()` is not successful, NULL is returned, `ld_errno` will be set to indicate the error. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values. Use "ldap_get_errno()—Retrieve Error Information” on page 91 function to retrieve the error information.

Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_first_entry API.</td>
</tr>
</tbody>
</table>

Related Information

- "ldap_next_entry()—Retrieve Next LDAP Entry” on page 131 — Return next entry in a chain of search results.
- "ldap_count_entries()—Retrieve Count of LDAP Entries” on page 36 — Return number of entries in a chain of search results.
- "ldap_get_entry_controls_np()—Extract Server Controls from an Entry” on page 90 — Extract server controls from an entry.
- "ldap_first_reference()—Retrieve First Continuation Reference in a Chain of Search Results” on page 84 — Return first continuation reference in a chain of search results.
- "ldap_next_reference()—Retrieve the next Continuation Reference in a Chain of Search Results” on page 134 — Return next continuation reference in a chain of search results.
- "ldap_count_references()—Count continuation references in a chain of search results” on page 39 — Return number of continuation reference in a chain of search results.
- "ldap_parse_reference_np()—Extract Information from a Continuation Reference” on page 143 — Extract information from a continuation reference.
- "ldap_first_message()—Retrieve First LDAP Message” — Retrieve first LDAP message.
- "ldap_next_message()—Retrieve Next LDAP Message” on page 132 — Retrieve next LDAP message.
- "ldap_msgfree()—Free LDAP Result Message” on page 125 — Free LDAP result message.
- "ldap_msgtype()—Retrieve the Type of an LDAP Message” on page 128 — Retrieve Type of an LDAP Message

API introduced: V4R3
#include <ldap.h>

LDAPMessage *ldap_first_message(LDAP *ld, LDAPMessage *result)

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

The `ldap_first_message()` routine is used to step through the list of messages in a result chain, as returned by the `ldap_result()` function. It is used to return a pointer to the first message in the list.

**Authorities and Locks**
No i5/OS authority is required.

**Parameters**
- **ld** (Input) The LDAP pointer returned by a previous call to `ldap_init()`—Perform an LDAP Initialization Operation” on page 105, `ldap_ssl_init—Initialize an SSL Connection.” on page 204 or “ldap_open()—Perform an LDAP Open Operation” on page 135.
- **result** (Input) The result returned by a call to “ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156 or one of the synchronous search routines “ldap_search_s()—Perform an LDAP Search Operation (Synchronous)” on page 171, “ldap_search_st()—Perform an LDAP Search Operation (Timed Synchronous)” on page 173, or “ldap_search_ext_s — Synchronously Search the Directory Using Controls” on page 168.

**Return Value**
LDAPMessage *
Pointer to the first message.

NULL when no message exists in the result set or if an error occurs.

**Error Conditions**
If `ldap_first_message()` is not successful, ld_errno will be set to indicate the error. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values. Use `ldap_get_errno()—Retrieve Error Information” on page 91 function to retrieve the error information.

**Error Messages**
The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_first_message API.</td>
</tr>
</tbody>
</table>

**Related Information**
- “ldap_count_messages()—Count messages in a result chain” on page 38 — Return the number of message in a result chain.
- “ldap_first_entry()—Retrieve First LDAP Entry” on page 81 — Retrieve first LDAP entry.
- “ldap_first_reference()—Retrieve First Continuation Reference in a Chain of Search Results” on page 84 — Return first continuation reference in a chain of search results.
- “ldap_msgfree()—Free LDAP Result Message” on page 125 — Free LDAP result message.
• “ldap_msgid()—Retrieve the Message ID Associated with an LDAP Message” on page 126 — Retrieve message ID associated with an LDAP message.
• “ldap_msgtype()—Retrieve the Type of an LDAP Message” on page 128 — Retrieve type of an LDAP message.
• “ldap_next_message()—Retrieve Next LDAP Message” on page 132 — Retrieve next LDAP message.
• “ldap_result2error()—Retrieve LDAP Error Information” on page 158 — Retrieve LDAP error information

API introduced: V4R5

### ldap_first_reference()—Retrieve First Continuation Reference in a Chain of Search Results

**Syntax**

```c
#include <ldap.h>
LDAPMessage *ldap_first_reference(LDAP *ld, LDAPMessage *result)
```

**Library Name/Service Program:** QSYS/QGLDCLNT

**Default Public Authority:** *USE

**Threadsafe:** Yes

The `ldap_first_reference()` is used to return the first continuation reference from the search result chain.

### Authorities and Locks

No i5/OS authority is required.

### Parameters

- **ld** (Input) The LDAP pointer returned by a previous call to “ldap_init()—Perform an LDAP Initialization Operation” on page 105, “ldap_ssl_init — Initialize an SSL Connection,” on page 204, or “ldap_open()—Perform an LDAP Open Operation” on page 135.

- **result** (Input) The result returned by a call to “ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156 or one of the synchronous search routines (“ldap_search_s()—Perform an LDAP Search Operation (Synchronous)” on page 171, “ldap_search_st()—Perform an LDAP Search Operation (Timed Synchronous)” on page 173, or “ldap_search_ext_s — Synchronously Search the Directory Using Controls” on page 168).

### Return Value

`LDAPMessage *`

Pointer to the first continuation reference. The pointer returned from `ldap_first_reference()` should be supplied on a subsequent call to “ldap_next_reference()—Retrieve the next Continuation Reference in a Chain of Search Results” on page 134 to get the next continuation reference.

**NULL** when no more continuation references exist in the result set to be returned.
**Error Conditions**

If `ldap_first_reference()` is not successful, `ld_errno` will be set to indicate the error. See "LDAP Client API Error Conditions" on page 300 for possible LDAP error code values. Use "ldap_get_errno()—Retrieve Error Information" on page 91 function to retrieve the error information.

**Error Messages**

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_first_reference API.</td>
</tr>
</tbody>
</table>

**Related Information**

- "ldap_first_entry()—Retrieve First LDAP Entry” on page 81 — Return first entry in a chain of search results.
- "ldap_next_entry()—Retrieve Next LDAP Entry” on page 131 — Return next entry in a chain of search results.
- "ldap_count_entries()—Retrieve Count of LDAP Entries” on page 36 — Return number of entry in a chain of search results.
- "ldap_get_entry_controls_np()—Extract Server Controls from an Entry” on page 90 — Extract server controls from an entry.
- "ldap_count_references()—Count continuation references in a chain of search results” on page 39 — Return the number of continuation reference in a chain of search results.
- "ldap_next_reference()—Retrieve the next Continuation Reference in a Chain of Search Results” on page 134 — Return next continuation reference in a chain of search results.
- "ldap_parse_reference_np()—Extract Information from a Continuation Reference” on page 143 — Extract information from a continuation reference.

API introduced: V4R5

---

**ldap_free_sort_keylist()—Free all Memory used by the Sort Key List**

Syntax

```c
#include <ldap.h>

void ldap_free_sort_keylist(
    LDAPsortkey **sortKeyList)
```

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

The `ldap_free_sort_keylist()` function is used to free all the memory used by the sort key list. This function must be called after the `ldap_create_sort_control()` function has completed.

See [LDAP Sort](#) for usage information about the functions used to perform sorting of entries returned from the server following an LDAP search operation.

**Authorities and Locks**

No i5/OS authority is required. All authority checking is done by the LDAP server.
Parameters

sortKeyList
(Input) Specifies the pointer to an array of LDAP sortkey structures, which represent attributes that the server uses to sort returned entries. Input when used for ldap_create_sort_control() and ldap_free_sort_keylist().

Return Value

NONE

Example

See Code disclaimer information for information pertaining to code examples.

See the example that uses ldap_create_sort_keylist(), ldap_create_sort_control(), ldap_free_sort_keylist(), and ldap_parse_sort_control() to perform an LDAP sorted search in "ldap_create_sort_keylist()—Create a Structure with Sort Key Values used when Sorting Search Results" on page 51—Create a Structure with Sort Key Values used when Sorting Search Results.

Related Information

- "ldap_create_sort_keylist()—Create a Structure with Sort Key Values used when Sorting Search Results" on page 51—Create a structure with sort key values.
- "ldap_create_sort_control()—Create a Sorted Results Control used when Sorting Search Results" on page 49—Create a sorted results control.
- "ldap_parse_sort_control()—Retrieve Values in a Sorted Results Control” on page 148—Retrieve values in a sorted results control.
- "ldap_create_page_control()—Create a Paged Results Control used when paging search results” on page 42—Create a paged results control.
- "ldap_parse_page_control()—Retrieve Values in a Paged Results Control” on page 139—Retrieve values in a paged results control.
- "ldap_search()—Perform an LDAP Search Operation” on page 164—Asynchronously search the directory.
- "ldap_parse_result()—Extract Information from Results” on page 144—Extract information from results.

API introduced: V5R3

Syntax

```c
#include <ldap.h>

typedef struct ldap_url_desc {
    char    *lud_host;    /* LDAP host to contact */
    int      lud_port;    /* port on host */
    char    *lud_dn;      /* base for search */
    char    **lud_attrs;  /* NULL-terminate list of attributes */
} ldap_url_desc;
```

ldap_free_urldesc()—Free an LDAP URL Description
int lud_scope;  /* a valid LDAP_SCOPE... value */
char *lud_filter;  /* LDAP search filter */
char *lud_string;  /* for internal use only */
}

LDAPURLDesc;

void ldap_free_urldesc(
   LDAPURLDesc *ludp)
{
   // Library Name/Service Program: QSYS/QGLDCLNT
   // Default Public Authority: *USE
   // Threadsafe: Yes

   The ldap_free_urldesc() function is called to free an LDAP URL description that was obtained from a call to the “ldap_url_parse()—Parse an LDAP URL” on page 214 function.

   **Authorities and Locks**
   No i5/OS authority is required.

   **Parameters**
   
   **ludp**  (Input) Points to the LDAP URL description, as returned by “ldap_url_parse()—Parse an LDAP URL” on page 214.

   **Return Value**
   None.

   **Error Conditions**
   The ldap_free_urldesc() API does not return an error code.

   **Error Messages**
   The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_free_urldesc API.</td>
</tr>
</tbody>
</table>

   **Related Information**

   - “ldap_is_ldap_url()—Verify LDAP URL” on page 108 — Check a URL string to see if it is an LDAP URL.
   - “ldap_url_parse()—Parse an LDAP URL” on page 214 — Break up an LDAP URL string into its components.
   - “ldap_url_search()—Perform an LDAP URL Search Operation” on page 217 — Asynchronously search using an LDAP URL.
   - “ldap_url_search_s() — Perform an LDAP URL Search Operation (Synchronous)” on page 219 — Synchronously search using an LDAP URL.
   - “ldap_url_search_st()—Perform an LDAP URL Search Operation (Timed Synchronous)” on page 221 — Synchronously search using an LDAP URL and a timeout.

   API introduced: V4R3
ldap_get_bind_controls()—Get LDAP Bind Controls

Syntax

```c
#include <ldap.h>

int ldap_get_bind_controls(
    LDAP *ld,
    LDAPControl ***bind_controls)
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The `ldap_get_bind_controls()` function allows a client using Simple Authentication Security Layer (SASL) binds to access controls returned by the server during a previous bind request. For example, this API could be used to get password policy response controls from a bind request (see "ldap_parse_pwdpolicy_response()—Obtain the LDAP Password Policy Error or Warning Codes" on page 141).

Authorities and Locks
No i5/OS authority is required.

Parameters

`ld` (Input) The LDAP pointer used in a previous call to “ldap_sasl_bind_s()—Perform an LDAP SASL Bind Request (Synchronous)” on page 161. Must not be NULL.

`bind_controls` (Output) On return, `bind_controls` will point to a NULL-terminated array of LDAPControl structures, or NULL if no controls were returned by the server. The caller must free the returned controls by using “ldap_controls_free()—Free storage allocated by the LDAP library” on page 32.

Return Value

`LDAP_SUCCESS` if the request was successful.

`LDAP_NO_MEMORY` if a memory allocation error occurred.

`LDAP_PARAM_ERROR` if `ld` or `bind_controls` are NULL.

Related Information

- "ldap_controls_free()—Free storage allocated by the LDAP library” on page 32 — Free an array of LDAPControl structures.
- "ldap_parse_pwdpolicy_response()—Obtain the LDAP Password Policy Error or Warning Codes” on page 141 — Obtain Error and Warning Codes from the Password Policy Response Control.
- "ldap_sasl_bind_s()—Perform an LDAP SASL Bind Request (Synchronous)” on page 161 — Perform an LDAP SASL Bind Request.
ldap_get_dn()—Retrieve the Distinguished Name of an Entry

Syntax
#include <ldap.h>

char *ldap_get_dn(
    LDAP *ld,
    LDAPMessage *entry)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The `ldap_get_dn()` function takes an entry as returned by “ldap_first_entry()—Retrieve First LDAP Entry” on page 81 or “ldap_next_entry()—Retrieve Next LDAP Entry” on page 131 and returns a copy of the entry’s Distinguished Name (DN). Memory for the DN will have been allocated and should be freed by a call to “ldap_memfree()—Free Memory Allocated by LDAP API” on page 110.

Authorities and Locks
No i5/OS authority is required.

Parameters

- `ld` (Input) Specifies the LDAP pointer returned by a previous call to “ldap_init()—Perform an LDAP Initialization Operation” on page 105, “ldap_ssl_init—Initialize an SSL Connection.” on page 204, or “ldap_open()—Perform an LDAP Open Operation” on page 135.
- `entry` (Input) The entry whose dn is to be retrieved, as returned by Specifies the LDAP pointer returned by a previous call to “ldap_first_entry()—Retrieve First LDAP Entry” on page 81 or “ldap_next_entry()—Retrieve Next LDAP Entry” on page 131.

Return Value

- Copy of the entry’s DN if the request was successful.
- NULL if the request was not successful.

Error Conditions

If `ldap_get_dn()` is not successful, `ld_errno` will be set to indicate the error. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values. Use “ldap_get_errno()—Retrieve Error Information” on page 91 function to retrieve the error information.

Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_get_dn API.</td>
</tr>
</tbody>
</table>

Related Information

- “ldap_explode_dn()—Break a Distinguished Name into Its Components” on page 69 — Convert a DN into its component parts.
**ldap_explode_dn_utf8()**—Break a UTF8 codepage Distinguished Name into Its Components

— Break a UTF8 codepage Distinguished Name into its components

API introduced: V4R5

---

**ldap_get_entry_controls_np()**—Extract Server Controls from an Entry

Syntax

```c
#include <ldap.h>

int ldap_get_entry_controls_np(
    LDAP *ld,
    LDAPMessage *entry,
    LDAPControl ***serverctrlsp)
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The `ldap_get_entry_controls_np()` routine is used to retrieve an array of server controls returned in an individual entry in a chain of search results.

Note the suffix "_np" which shows the API is in a preliminary implementation, and is not documented in the Internet Draft. The Internet community may standardize this API in the future.

**Authorities and Locks**

No i5/OS authority is required.

**Parameters**

- **ld** (Input) Specifies the LDAP pointer returned by a previous call to "ldap_init()—Perform an LDAP Initialization Operation” on page 105 or "ldap_ssl_init —Initialize an SSL Connection,” on page 204, or "ldap_open()—Perform an LDAP Open Operation” on page 135.

- **entry** (Input) Specifies a pointer to an entry returned on a previous call to "ldap_first_entry()—Retrieve First LDAP Entry” on page 81 or "ldap_next_entry()—Retrieve Next LDAP Entry” on page 131.

- **serverctrlsp** (Input) Specifies a pointer to a result parameter that is filled in with an allocated array of controls copied out of the entry. The control array should be freed by calling “ldap_controls_free()—Free storage allocated by the LDAP library” on page 32.

**Return Value**

- LDAP_SUCCESS if the call was successful
- another LDAP error code if the call was not successful.

**Error Conditions**

The `ldap_get_entry_controls_np()` API will return LDAP error code if not successful. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values.
Error Messages
The following message may be sent from this function.

Message ID | Error Message Text
---|---
CPF3CF2 E | Error(s) occurred during running of ldap_get_entry_controls_np API.

Related Information
- "ldap_first_entry()—Retrieve First LDAP Entry” on page 81 — Return first entry in a chain of search results.
- "ldap_next_entry()—Retrieve Next LDAP Entry” on page 131 — Return next entry in a chain of search results.
- "ldap_count_entries()—Retrieve Count of LDAP Entries” on page 36 — Return the number of entry in a chain of search results.
- "ldap_first_reference()—Retrieve First Continuation Reference in a Chain of Search Results” on page 84 — Return first continuation reference in a chain of search results.
- "ldap_next_reference()—Retrieve the next Continuation Reference in a Chain of Search Results” on page 134 — Return next continuation reference in a chain of search results.
- "ldap_count_references()—Count continuation references in a chain of search results” on page 39 — Return number of continuation reference in a chain of search results.
- "ldap_parse_reference_np()—Extract Information from a Continuation Reference” on page 143 — Extract information from a continuation reference.

API introduced: V4R5

ldap_get_errno()—Retrieve Error Information
Syntax
```c
#include <ldap.h>

int ldap_get_errno(
    LDAP *ld)
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The `ldap_get_errno()` function retrieves information about the most recent error that occurred for an LDAP operation. This function can be called for any LDAP API that does not return an error.

The “ldap_get_lerror()—Retrieve Error Information” on page 93 API returns more error information than `ldap_get_errno()`.

Authorities and Locks
No i5/OS authority is required.

Parameters
`ld` (Input) Specifies the LDAP pointer returned by a previous call to “ldap_init()—Perform an LDAP
Initialization Operation” on page 105, “ldap_ssl_init — Initialize an SSL Connection.” on page 204, or “ldap_open()—Perform an LDAP Open Operation” on page 135.

Return Value

LDAP error code

See “LDAP Client API Error Conditions” on page 300 for possible LDAP error codes values.

Error Messages

The following message may be sent from this function.

Message ID        Error Message Text
CPF3CF2 E          Error(s) occurred during running of ldap_get_errno API.

Related Information

- “ldap_err2string()—Retrieve LDAP Error Message String” on page 68 — Convert LDAP error indication to a string.
- “ldap_get_lderrno()—Retrieve Error Information” on page 93 — Retrieve Error Information.
- “ldap_perror()—Print LDAP Error Information” on page 149 — Print an LDAP error indication to standard error.
- “ldap_result2error()—Retrieve LDAP Error Information” on page 158 — Extract LDAP error indication from LDAP result.

API introduced: V4R3

ldap_get_iconv_local_codepage() — Get the Active LDAP Code Page

Syntax

#include <ldap.h>

char *
ldap_get_iconv_local_codepage ( )

Default Public Authority: USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: No

The ldap_get_iconv_local_codepage() API is used to obtain the active LDAP code page. It returns the value of a global variable ldap_global_codepage set by ldap_set_iconv_local_codepage() — Set the Active LDAP Code Page” on page 185. To free the returned string, use “ldap_memfree()—Free Memory Allocated by LDAP API” on page 110.

Authorities and Locks

No i5/OS authority is required.

Parameters

No parameter are passed to ldap_get_iconv_local_codepage().
Return Value

LDAP Code page
   if the request was successful.
NULL    if the request was not successful.

Error Conditions

If ldap_get_iconv_local_codepage() is not successful, it returns NULL.

Error Messages

The following message may be sent from this function.

Message ID  Error Message Text
CPF3CF2 E    Error(s) occurred during running of ldap_get_iconv_local_codepage API.

Related Information

- “ldap_xlate_local_to_utf8()— Convert String From the Local Code Page to UTF-8 Encoding” on page 228 — Convert String From the Local to UTF-8 Code Page.
- “ldap_xlate_utf8_to_local() — Convert String From the UTF-8 Encoding to Local Code Page” on page 231 — Convert String From UTF-8 to Local Code Page.
- “ldap_xlate_local_to_unicode()— Convert String From the Local Code Page to UCS-2 (or UNICODE) Encoding” on page 226 — Convert String From the Local to UCS-2 Code Page.
- “ldap_xlate_unicode_to_local() — Convert String From the UCS-2 (or UNICODE) Encoding to Local Code Page” on page 229 — Convert String From UCS-2 to Local Code Page.
- “ldap_set_iconv_local_codepage() — Set the Active LDAP Code Page” on page 185 — Set the Active LDAP Code Page.
- “ldap_set_iconv_local_charset()— Set the Active LDAP Character Set” on page 183 — Set the Active LDAP Character set.
- “ldap_set_locale() — Change the Locale Used by LDAP” on page 188 — Change the Locale Used by LDAP.
- “ldap_get_locale()— Get Active LDAP Locale” on page 95 — Get the Locale Used by LDAP.

API introduced: V4R5

Syntax

#include <ldap.h>

int ldap_get_lderrno(
   LDAP *ld, 
   char **dn, 
   char **errmsg)

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threading Safe: Yes
The **ldap_get_lderrno**() function retrieves information about the most recent error that occurred for an LDAP operation. This function can be called for any LDAP API that does not return an error.

When an error occurs at the LDAP server, the server returns both an LDAP result code and a message containing any additional information about the error from the server. If the error occurred because an entry specified by a Distinguished Name (DN) could not be found, the server may also return the portion of the DN that identifies an existing entry. Use **ldap_get_lderrno**() to obtain both the message containing error information and the matched DN.

**Authorities and Locks**

No i5/OS authority is required.

**Parameters**

- **ld** *(Input)* Specifies the LDAP pointer returned by a previous call to **ldap_init**—Perform an LDAP Initialization Operation” on page 105, **ldap_ssl_init**—Initialize an SSL Connection.” on page 204, or **ldap_open**—Perform an LDAP Open Operation” on page 135.

- **dn** *(Output)* The distinguished name (DN) that identifies an existing entry, indicating how much of the name in the request was recognized by the server. The DN is returned when an LDAP_NO_SUCH_OBJECT error is returned from the server on some previous operation. The matched DN string should be freed by calling **ldap_memfree**—Free Memory Allocated by LDAP API” on page 110.

- **errmsg** *(Output)* The text of the error message, as returned from the server. The error message string should be freed by calling **ldap_memfree**—Free Memory Allocated by LDAP API” on page 110.

**Return Value**

**LDAP error code**

See **LDAP Client API Error Conditions** on page 300 for possible LDAP error codes values.

**Error Messages**

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_get_lderrno API.</td>
</tr>
</tbody>
</table>

**Related Information**

- **“ldap_err2string”**—Retrieve LDAP Error Message String” on page 68 — Convert LDAP error indication to a string.
- **“ldap_get_errno”**—Retrieve Error Information” on page 91 — Obtain information from most recent error.
- **“ldap_perror”**—Print LDAP Error Information” on page 149 — Print an LDAP error indication to standard error.
- **“ldap_result2error”**—Retrieve LDAP Error Information” on page 158 — Extract LDAP error indication from LDAP result.
- **“ldap_set_lderrno”** — Set Error Information” on page 186 — Set Error Information
ldap_get_locale()— Get Active LDAP Locale

Syntax
#include <ldap.h>

char *ldap_get_locale() 

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: No

The ldap_get_locale() API is used to obtain the active LDAP locale. To free the returned string, use “ldap_memfree()—Free Memory Allocated by LDAP API” on page 110.

Authorities and Locks
No i5/OS authority is required.

Parameters
No parameters are passed to ldap_get_locale()

Return Value
Active LDAP Locale
if the request was successful.
NULL if the request was not successful.

Error Conditions
If ldap_get_locale() is not successful, it returns NULL.

Error Messages
The following message may be sent from this function.

Message ID | Error Message Text
--- | ---
CPF3CF2 E | Error(s) occurred during running of ldap_get_locale API.

Related Information
- “ldap_xlate_local_to_utf8()—Convert String From the Local Code Page to UTF-8 Encoding” on page 228 — Convert String From the Local to UTF-8 Code Page.
- “ldap_xlate_utf8_to_local() — Convert String From the UTF-8 Encoding to Local Code Page” on page 231 — Convert String From UTF-8 to Local Code Page.
- “ldap_xlate_local_to_unicode()—Convert String From the Local Code Page to UCS-2 (or UNICODE) Encoding” on page 226 — Convert String From the Local to UCS-2 Code Page.
- “ldap_xlate_unicode_to_local() — Convert String From the UCS-2 (or UNICODE) Encoding to Local Code Page” on page 229 — Convert String From UCS-2 to Local Code Page.
- “ldap_get_iconv_local_codepage()—Get the Active LDAP Code Page” on page 92 — Get the Active LDAP Code Page.
• “ldap_set_iconv_local_codepage() — Set the Active LDAP Code Page” on page 185 — Set the Active LDAP Code Page.
• “ldap_set_iconv_local_charset()— Set the Active LDAP Character Set” on page 183 — Set the Active LDAP Character set.
• “ldap_set_locale() — Change the Locale Used by LDAP” on page 188 — Change the Locale Used by LDAP.

API introduced: V4R5

*Lightweight Directory Access Protocol (LDAP) APIs," on page 1 | APIs by category

---

## ldap_get_option()—Retrieve LDAP Options

Syntax
#include <ldap.h>

```c
int ldap_get_option(
    LDAP *ld,
    int optionToGet,
    void *optionValue)
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafes: Yes

The `ldap_get_option()` function is used to query settings associated with the specified LDAP connection.

### Authorities and Locks

No i5/OS authority is required.

### Parameters

**ld** (Input) The LDAP pointer returned by a previous call to `ldap_init()`—Perform an LDAP Initialization Operation” on page 105, `ldap_ssl —Initialize an SSL Connection.” on page 204, or `ldap_open()”—Perform an LDAP Open Operation” on page 135. If a NULL ld is passed in, the default value for the option is retrieved.

**optionToGet**

(Input) The option value that is to be queried on the `ldap_get_option()` call. See below for the list of supported options.

**optionValue**

(Input) The address of the storage in which to return the queried value using `ldap_get_option()`.

The following session settings can be get using the `ldap_get_option()` API:

- **LDAP_OPT_SIZELIMIT**
  - maximum number of entries that can be returned on a search operation
- **LDAP_OPT_TIMELIMIT**
  - maximum number of seconds to wait for search results.
- **LDAP_OPT_REFHOPLIMIT**
  - maximum number of referrals in a sequence that the client can follow rules for following aliases at the server.
- **LDAP_OPT_DEREF**
  - whether or not referrals should be followed by the client.
- **LDAP_OPT_DEBUG**
  - debug options.
- **LDAP_OPT_SSL_CIPHER**
  - SSL ciphers to use.
- **LDAP_OPT_SSL_TIMEOUT**
  - SSL timeout for refreshing session keys
- **LDAP_OPT_REBIND_FN**
  - address of application’s setrebindproc procedure.
LDAP_OPT_PROTOCOL_VERSION  LDAP protocol version to use (V2 or V3).
LDAP_OPT_SERVER_CONTROLS  default server controls.
LDAP_OPT_CLIENT_CONTROLS  default client library controls.
LDAP_OPT_UTF8_IO  mode for converting string data between the local code page and UTF-8
LDAP_OPT_HOST_NAME  current host name
LDAP_OPT_ERROR_NUMBER  error number
LDAP_OPT_ERROR_STRING  error string
LDAP_OPT_EXT_ERROR  extended error code
LDAP_OPT_EXT_GSS_ERR  GSSAPI extended error code

Additional details on specific options for `ldap_get_option()` are provided in the following sections.

**LDAP_OPT_SIZELIMIT**
Specifies the maximum number of entries that can be returned on a search operation. Note: the actual size limit for operations is also bounded by the maximum number of entries that the server is configured to return. Thus, the actual size limit will be the lesser of the value specified on this option and the value configured in the LDAP server. The default sizelimit is unlimited, specified with a value of zero (thus deferring to the sizelimit setting of the LDAP server).

Examples:
```
sizelimit=50;
ldap_set_option( ld, LDAP_OPT_SIZELIMIT, &sizelimit);
ldap_get_option( ld, LDAP_OPT_SIZELIMIT, &sizelimit );
```

**LDAP_OPT_TIMELIMIT**
Specifies the number of seconds to wait for search results. Note: the actual time limit for operations is also bounded by the maximum time that the server is configured to allow. Thus, the actual time limit will be the lesser of the value specified on this option and the value configured in the LDAP server. The default is unlimited (specified with a value of zero).

Examples:
```
timevalue=50;
ldap_set_option( ld, LDAP_OPT_TIMELIMIT, &timevalue);
ldap_get_option( ld, LDAP_OPT_TIMELIMIT, &timevalue );
```

**LDAP_OPT_REFHOPLIMIT**
Specifies the maximum number of hops that the client library will take when chasing referrals. The default is 5.

Examples:
```
hoplimit=7;
ldap_set_option( ld, LDAP_OPT_REFHOPLIMIT, &hoplimit);
ldap_get_option( ld, LDAP_OPT_REFHOPLIMIT, &hoplimit );
```

**LDAP_OPT_DEREF**
Specifies alternative rules for following aliases at the server. The default is `LDAP_DEREF_NEVER`.

Supported values:
- `LDAP_DEREF_NEVER` 0
- `LDAP_DEREF_SEARCHING` 1
- `LDAP_DEREF_FINDING` 2
Examples:
int deref = LDAP_DEREF_NEVER;
ldap_set_option(ld, LDAP_OPT_DEREF, &deref);
ldap_get_option(ld, LDAP_OPT_DEREF, &deref);

**LDAP_OPT_REFERRALS**
Specifies whether the LDAP library will automatically follow referrals returned by LDAP servers or not. It can be set to one of the constants `LDAP_OPT_ON` or `LDAP_OPT_OFF`. By default, the LDAP client will follow referrals.

Examples:
int value;
ldap_set_option(ld, LDAP_OPT_REFFERALS, (void*)LDAP_OPT_ON);
ldap_get_option(ld, LDAP_OPT_REFFERALS, &value);

**LDAP_OPT_DEBUG**
Specifies a bit-map that indicates the level of debug trace for the LDAP library.

Supported values:

<table>
<thead>
<tr>
<th>LDAP_DEBUG_OFF</th>
<th>0x000</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAP_DEBUG_TRACE</td>
<td>0x001</td>
</tr>
<tr>
<td>LDAP_DEBUG_PACKETS</td>
<td>0x002</td>
</tr>
<tr>
<td>LDAP_DEBUGARGS</td>
<td>0x004</td>
</tr>
<tr>
<td>LDAP_DEBUGCONNS</td>
<td>0x008</td>
</tr>
<tr>
<td>LDAP_DEBUGBER</td>
<td>0x010</td>
</tr>
<tr>
<td>LDAP_DEBUGFILTER</td>
<td>0x020</td>
</tr>
<tr>
<td>LDAP_DEBUGCONFIG</td>
<td>0x040</td>
</tr>
<tr>
<td>LDAP_DEBUGACL</td>
<td>0x080</td>
</tr>
<tr>
<td>LDAP_DEBUGSTATS</td>
<td>0x100</td>
</tr>
<tr>
<td>LDAP_DEBUGSTATS2</td>
<td>0x200</td>
</tr>
<tr>
<td>LDAP_DEBUGSHELL</td>
<td>0x400</td>
</tr>
<tr>
<td>LDAP_DEBUGPARSE</td>
<td>0x800</td>
</tr>
<tr>
<td>LDAP_DEBUGANY</td>
<td>0xffff</td>
</tr>
</tbody>
</table>

Examples:
int value;
int debugvalue = LDAP_DEBUG_TRACE | LDAP_DEBUG_PACKETS;
ldap_set_option(ld, LDAP_OPT_DEBUG, &debugvalue);
ldap_get_option(ld, LDAP_OPT_DEBUG, &value);

**LDAP_OPT_SSL_CIPHER**
Specifies a set of one or more ciphers to be used when negotiating the cipher algorithm with the LDAP server. The first cipher in the list that is common with the list of ciphers supported by the server is chosen. For the export version of the library, the value used is "0306". For the domestic version of the library, the default value is "05040A090306". Note that the cipher string supported by the export version of the LDAP client library is fixed and cannot be modified.

Supported ciphers:
LDAP_SSL_RC4_MD5_EX 03
LDAP_SSL_RC2_MD5_EX 06
LDAP_SSL_RC4_SHA_US 05 (Non-export only)
LDAP_SSL_RC4_MDC5_US 04 (Non-export only)
LDAP_SSL_DES_SHA_US 09 (Non-export only)
LDAP_SSL_3DES_SHA_US 0A (Non-export only)
LDAP_SSL_AES_SHA_US 2F (Non-export only)

Examples:
char *setcipher = "2F090A";
char *getcipher;
ldap_setoption(1d, LDAP_OPT_SSL_CIPHER, setcipher);
ldap_getoption(1d, LDAP_OPT_SSL_CIPHER, &getcipher);

Use "ldap_memfree()—Free Memory Allocated by LDAP API" on page 110 to free the memory returned by the call to ldap_getoption().

LDAP_OPT_SSL_TIMEOUT
Specifies in seconds the SSL inactivity timer. After the specified seconds, in which no SSL activity has occurred, the SSL connection will be refreshed with new session keys. A smaller value may help increase security, but will have a small impact on performance. The default SSL timeout value is 43200 seconds.

Examples:
value = 100;
ldap_set_option(1d, LDAP_OPT_SSL_TIMEOUT, &value);
ldap_get_option(1d, LDAP_OPT_SSL_TIMEOUT, &value)

LDAP_OPT_REBIND_FN
Specifies the address of a routine to be called by the LDAP library when the need arises to authenticate a connection with another LDAP server. This can occur, for example, when the LDAP library is chasing a referral. If a routine is not defined, referrals will always be chased using the anonymous identity. A default routine is not defined.

Examples:
extern LDAPRebindProc proc_address;
LDAPRebindProc value;
ldap_setoption(1d, LDAP_OPT_REBIND_FN, &proc_address);
ldap_getoption(1d, LDAP_OPT_REBIND_FN, &value);

LDAP_OPT_PROTOCOL_VERSION
Specifies the LDAP protocol to be used by the LDAP client library when connecting to an LDAP server. Also used to determine which LDAP protocol is being used for the connection. For an application that uses "ldap_init()—Perform an LDAP Initialization Operation" on page 105 to create the LDAP connection the default value of this option will be LDAP_VERSION3 for communicating with the LDAP server. The default value of this option will be LDAP_VERSION2 if the application uses the deprecated "ldap_open()—Perform an LDAP Open Operation" on page 135 API. In either case, the LDAP_OPT_PROTOCOL_VERSION option can be used with "ldap_setoption() — Set LDAP Options" on page 189 to change the default. The LDAP protocol version should be reset prior to issuing the bind (or any operation that causes an implicit bind).

Examples:
version2 = LDAP_VERSION2;
version3 = LDAP_VERSION3;
/* Example for Version 3 application setting version to version 2 */
ldap_setoption(1d, LDAP_OPT_PROTOCOL_VERSION, &version2);
/* Example of Version 2 application setting version to version 3 */
ldap_set_option( ld, LDAP_OPT_PROTOCOL_VERSION, &version3);
ldap_get_option( ld, LDAP_OPT_PROTOCOL_VERSION, &value);

The value returned by `ldap_get_option()` when `LDAP_OPT_PROTOCOL_VERSION` is specified can be used to determine how parameters should be passed to the `ldap_set_option()` — Set LDAP Options” on page 189 call. The easiest way to work with this compatibility feature is to guarantee that calls to “ldap_set_option() — Set LDAP Options” on page 189 are all performed while LDAP_OPT_PROTOCOL_VERSION is set to the same value. If this cannot be guaranteed by the application, then follow the format of the example below when coding the call to “ldap_set_option() — Set LDAP Options” on page 189

Examples:

```c
int sizeLimit=100;

int protocolVersion;

ldap_get_option( ld, LDAP_OPT_PROTOCOL_VERSION, &protocolVersion );

if ( protocolVersion == LDAP_VERSION2 ) {
    ldap_set_option( ld, LDAP_OPT_SIZELIMIT, (void *)&sizeLimit );
} else { /* the protocol version is LDAP_VERSION3 */
    ldap_set_option( ld, LDAP_OPT_SIZELIMIT, &sizeLimit );
}
```

**LDAP_OPT_SERVER_CONTROLS**

Specifies a default list of server controls to be sent with each request. The default list can be overridden by specifying a server control, or list of server controls, on specific APIs. By default, there are no settings for Server Controls.

**Example:**

```c
ldap_set_option( ld, LDAP_OPT_SERVER_CONTROLS, &ctrlp);
```

**LDAP_OPT_CLIENT_CONTROLS**

Specifies a default list of client controls to be processed by the client library with each request. Since client controls are not defined for this version of the library, the `ldap_set_option()` API can be used to define a set of default, non-critical client controls. If one or more client controls in the set is critical, the entire list is rejected with a return code of `LDAP_UNAVAILABLE_CRITICAL_EXTENSION`.

**LDAP_OPT_UTF8_IO**

Specifies whether the LDAP library will automatically convert string data to and from the local code page. It can be set to one of the constants `LDAP_UTF8_XLATE_ON` or `LDAP_UTF8_XLATE_OFF`. By default, the LDAP library will convert string data.

When conversion is disabled, the LDAP library assumes that data received from the application by LDAP APIs is already represented in UTF-8. Similarly, the LDAP library assumes that the application is prepared to receive string data from the LDAP library represented in UTF-8 (or as binary).

When `LDAP_UTF8_XLATE_ON` is set (the default), the LDAP library assumes that string data received from the application by LDAP APIs is in the default (or explicitly designated) code page. Similarly, all string data returned from the LDAP library (back to the application) is converted to the designated local code page.

**Notes:**

1. Only string data supplied on connection-based APIs will be translated (that is, only those APIs that include an `ld` will be subject to translation).
2. Translation of strings from a UTF-8 encoding to local code page may result in loss of data when one
or more characters in the UTF-8 encoding cannot be represented in the local code page. When this
occurs, a substitution character replaces any UTF-8 characters that cannot be converted to the local
code page.

Example:

```c
int value;
ldap_get_option( ld, LDAP_OPT_UTF8_IO, &value);
```

**LDAP_OPT_HOST_NAME**

This is a read-only option that returns a pointer to the hostname for the original connection (as specified
on “ldap_init()—Perform an LDAP Initialization Operation” on page 105, “ldap_open()—Perform an
LDAP Open Operation” on page 135, or “ldap_ssl_init — Initialize an SSL Connection.” on page 204).

Example:

```c
char *hostname;
ldap_get_option( ld, LDAP_OPT_HOST_NAME, &hostname);
```

Use “ldap_memfree()—Free Memory Allocated by LDAP API” on page 110 to free the memory returned
by the call to `ldap_get_option()`.

**LDAP_OPT_ERROR_NUMBER**

This is a read-only option that returns the error code associated with the most recent LDAP error that
occurred for the specified LDAP connection.

Example:

```c
int error;
ldap_get_option( ld, LDAP_OPT_ERROR_NUMBER, &error);
```

**LDAP_OPT_ERROR_STRING**

This is a read-only option that returns the text message associated with the most recent LDAP error that
occurred for the specified LDAP connection.

Example:

```c
char *error_string;
ldap_get_option( ld, LDAP_OPT_ERROR_STRING, &error_string);
```

Use “ldap_memfree()—Free Memory Allocated by LDAP API” on page 110 to free memory returned by
the call to `ldap_get_option()`.

**LDAP_OPT_EXT_ERROR**

This is a read-only option that returns the extended error code. For example, if an SSL error occurred
when attempting to call an “ldap_search_s()—Perform an LDAP Search Operation (Synchronous)” on
page 171 API, the actual SSL error can be obtained by using `LDAP_OPT_EXT_ERROR`.

Example:

```c
int exterror;
ldap_get_option( ld, LDAP_OPT_EXT_ERROR, &exterror);
```

Returns errors reported by the SSL library.
LDAP_OPT_EXT_GSS_ERR
This is a read-only option that returns the extended error code from SASL binds using the GSSAPI mechanism.

Example:
```c
int gsserror;
ldap_get_option( ld, LDAP_OPT_EXT_GSS_ERR, &gsserror);
```

Return Value
LDAP_SUCCESS
if the request was successful.

another LDAP error code
if the request was not successful.

Error Conditions
The `ldap_get_option()` API will return an LDAP error code if not successful. See “LDAP Client API Error Conditions” on page 300 for possible values for LDAP error codes.

Error Messages
The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_get_option API.</td>
</tr>
</tbody>
</table>

Related Information
- “[ldap_init]—Perform an LDAP Initialization Operation” on page 105 — Initializes a session with an LDAP server.
- “[ldap_set_option] — Set LDAP Options” on page 189 — Set an option associated with an LDAP descriptor.
- “[ldap_version — Obtain LDAP Version and SSL Cipher Information” on page 225 — Obtain LDAP version and SSL cipher information.

API introduced: V4R3
The `ldap_get_values()` function is used to retrieve attribute values from an LDAP entry as returned by
"ldap_first_entry()—Retrieve First LDAP Entry” on page 81 or "ldap_next_entry()—Retrieve Next LDAP Entry” on page 131. `ldap_get_values()` uses the entry and the attribute attr whose values are wanted and returns a NULL-terminated array of the attribute’s values. The returned array should be freed with "ldap_value_free()—Free Memory Allocated by ldap_get_values()” on page 223 when it is no longer needed.

Use "ldap_get_values_len()—Retrieve a Set of Binary Attribute Values” on page 104 to get binary attribute values.

Authorities and Locks
No i5/OS authority is required.

Parameters

`ld` (Input) Specifies the LDAP pointer returned by a previous call to "ldap_init()—Perform an LDAP Initialization Operation” on page 105, "ldap_ssl_init —Initialize an SSL Connection.” on page 204, or "ldap_open()—Perform an LDAP Open Operation” on page 135.

`entry` (Input) Specifies an LDAP entry as returned from "ldap_first_entry()—Retrieve First LDAP Entry” on page 81 or "ldap_next_entry()—Retrieve Next LDAP Entry” on page 131.

`attr` (Input) Specifies the attribute whose values are desired.

Return Value

Array of Values
  if the request was successful.

NULL  if the request was not successful.

Error Conditions

The `ldap_get_values()` API will return NULL and set the `ld_errno` error code, if not successful. See "LDAP Client API Error Conditions” on page 300 for possible LDAP error code values.

Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_get_values API.</td>
</tr>
</tbody>
</table>

Related Information

- "ldap_get_values_len()—Retrieve a Set of Binary Attribute Values” on page 104 — Return an attribute’s binary values.
- "ldap_count_values()—Retrieve Count of Attribute Values” on page 40 — Return number of values.
- "ldap_count_values_len()—Retrieve Count of Binary Attribute Values” on page 41 — Return number of binary values.
- "ldap_value_free()—Free Memory Allocated by ldap_get_values()” on page 223 — Free memory allocated by ldap_get_values().
- "ldap_value_free_len()—Free Memory Allocated by ldap_get_values_len()” on page 224 — Free memory allocated by ldap_get_values_len().
API introduced: V4R3

ldap_get_values_len()—Retrieve a Set of Binary Attribute Values

Syntax
#include <ldap.h>

struct berval {
    unsigned long bv_len;
    char *bv_val;
};

struct berval **ldap_get_values_len(
    LDAP *ld,
    LDAPMessage *entry,
    const char *attr)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The ldap_get_values_len() function is used to retrieve attribute values that are binary in nature from an LDAP entry as returned by “ldap_first_entry()—Retrieve First LDAP Entry” on page 81 or “ldap_next_entry()—Retrieve Next LDAP Entry” on page 131.

The ldap_get_values_len() API uses the same parameters as “ldap_get_values()—Retrieve a Set of Attribute Values from an Entry” on page 102, but returns a NULL-terminated array of pointers to berval structures, each containing the length of and a pointer to a value. Use “ldap_value_free_len()—Free Memory Allocated by ldap_get_values_len()” on page 224 to free the returned attribute values when they are no longer needed.

Authorities and Locks
No i5/OS authority is required.

Parameters
ld (Input) Specifies the LDAP pointer returned by a previous call to “ldap_init()—Perform an LDAP Initialization Operation” on page 105, “ldap_ssl_init — Initialize an SSL Connection.” on page 204, or “ldap_open()—Perform an LDAP Open Operation” on page 135.

entry (Input) Specifies an LDAP entry as returned from “ldap_first_entry()—Retrieve First LDAP Entry” on page 81 or “ldap_next_entry()—Retrieve Next LDAP Entry” on page 131.

attr (Input) Specifies the attribute whose values are desired.

Return Value
NULL-terminated array of pointers to berval structures
if the request was successful.

NULL if the request was not successful.

Error Conditions
The ldap_get_values_len() API will return NULL and set the ld_errno error code if not successful. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values.
Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_get_values_len API.</td>
</tr>
</tbody>
</table>

Related Information

- "ldap_get_values()—Retrieve a Set of Attribute Values from an Entry” on page 102 — Return an attribute’s values.
- "ldap_count_values()—Retrieve Count of Attribute Values” on page 40 — Return number of values.
- "ldap_count_values_len()—Retrieve Count of Binary Attribute Values” on page 41 — Return number of binary values.
- "ldap_value_free()—Free Memory Allocated by ldap_get_values()” on page 223 — Free memory allocated by ldap_get_values().
- "ldap_value_free_len()—Free Memory Allocated by ldap_get_values_len()” on page 224 — Free memory allocated by ldap_get_values_len().

API introduced: V4R3

Top | “Lightweight Directory Access Protocol (LDAP) APIs,” on page 1 | APIs by category

ldap_init()—Perform an LDAP Initialization Operation

Syntax

```c
#include <ldap.h>

LDAP *ldap_init(
    char *host,
    int port)
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The `ldap_init()` API is used to allocate an LDAP structure, which is used to identify the connection and to maintain per-connection information.

The `ldap_init()` API returns a pointer to an LDAP structure, which should be passed to subsequent calls to other LDAP functions such as `ldap_bind()` and `ldap_search()`.

`ldap_init()` initializes a session with an LDAP server. The server is not actually contacted until an operation is performed that requires it, allowing various options to be set after initialization, but before actually contacting the host. It allocates an LDAP structure which is used to identify the connection and maintain per-connection information. Although still supported, the use of "ldap_open()—Perform an LDAP Open Operation” on page 135 is deprecated. Use of `ldap_init()` instead of ”ldap_open()—Perform an LDAP Open Operation” on page 135 is recommended.

Authorities and Locks

No i5/OS authority is required.
Parameters

**host**  (Input) Several methods are supported for specifying one or more target LDAP servers, including the following:

*Explicit Host List* Specifies the name of the host on which the LDAP server is running. The *host* parameter may contain a blank-separated list of hosts to try to connect to, and each host may optionally be of the form *host:*port. If present, the *port* overrides the *port* parameter.

The following are typical examples:

```c
ld=ldap_init ("server1", ldap_port);
ld=ldap_init ("server2:1200", ldap_port);
ld=ldap_init ("server1:800 server2:2000 server3", ldap_port);
```

*Localhost* If the *host* parameter is NULL, the LDAP server will be assumed to be running on the local host.

*Default Hosts* If the *host* parameter is set to **LDAP_URL_PREFIX** ("ldap://") the LDAP library will attempt to locate one or more default LDAP servers, with non-SSL ports, using the SecureWay [ldap_server_locate()— Locate Suitable LDAP Servers](#) on page 178 function. The port specified on the call is ignored, since *ldap_server_locate()* returns the port.

For example, the following two are equivalent:

```c
ld=ldap_init ("ldap://", ldap_port);
ld=ldap_init (LDAP_URL_PREFIX, LDAP_PORT);
```

If more than one default server is located, the list is processed in sequence, until an active server is found.

The LDAP URL can include a Distinguished Name (DN), used as a filter for selecting candidate LDAP servers based on the server’s suffix (or suffixes). If the most significant portion of the DN is an exact match with a server’s suffix (after normalizing for case), the server is added to the list of candidate servers. For example, the following will only return default LDAP servers that have a suffix that supports the specified DN:

```c
ld=ldap_init ("1ldap:///cn=fred, dc=austin, dc=ibm, dc=com", LDAP_PORT);
```

In this case, a server that has a suffix of "dc=austin, dc=ibm, dc=com" would match. If more than one default server is located, the list is processed in sequence, until an active server is found.

If the LDAP URL contains a host name and optional port, the host is used to create the connection. No attempt is made to locate the default server(s), and the DN, if present, is ignored.

For example, the following two are equivalent:

```c
ld=ldap_init ("1ldap://myserver", LDAP_PORT);
ld=ldap_init ("myserver", LDAP_PORT);
```
Local Socket

If the host parameter is prefixed with "/", the host parameter is assumed to be the name of a UNIX socket (that is, socket family is AF_UNIX) and port is ignored. Use of a UNIX socket requires the LDAP server to be running on the local host. In addition, the LDAP server must be listening on the specified UNIX socket. The i5/OS Secureway Directory Services server listens on the /tmp/s.slapd local socket, in addition to any configured TCP/IP ports.

For example:
ld=ldap_init ("/tmp/s.slapd", ldap_port);

port

Specifies the port number to which to connect. If the default IANA-assigned port of 389 is desired, LDAP_PORT should be specified.

Return Value

Pointer to an LDAP structure

if the request was successful.

NULL if the request was not successful.

Error Conditions

The ldap_init() API will return NULL if not successful.

Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2E</td>
<td>Error(s) occurred during running of ldap_init API</td>
</tr>
</tbody>
</table>

Related Information

- "ldap_open()—Perform an LDAP Open Operation” on page 135 — Open a connection to an LDAP server (deprecated).
- "ldap_ssl_init —Initialize an SSL Connection.” on page 204 — Initializes an SSL Connection
- "ldap_set_option() — Set LDAP Options” on page 189 — Set an option associated with an LDAP descriptor.
- "ldap_get_option()—Retrieve LDAP Options” on page 96 — Get an option associated with an LDAP descriptor.
- "ldap_version — Obtain LDAP Version and SSL Cipher Information” on page 225 — Obtain LDAP version and SSL cipher information.

API introduced: V4R3
ldap_insert_control()—Insert a Control in the Llist of LDAP Server Controls

Syntax
#include <ldap.h>

int ldap_insert_control(
    LDAPControl *newControl,
    LDAPControl ***ctrlList)

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

The ldap_insert_control() function is used to insert a control in the list of LDAP server controls.

Note: The function will allocate space in the list for the control, but will not allocate the actual control.

Authorities and Locks
No i5/OS authority is required. All authority checking is done by the LDAP server.

Parameters

newControl
(Input) Specifies the LDAP server control to be inserted. See LDAP Controls for more information about server controls.

ctrlList
(Input) Specifies a list of LDAP server controls. See LDAP Controls for more information about server controls.

Return Value

LDAP_SUCCESS
if the request was successful.

LDAP_NO_MEMORY
if the control could not be inserted.

Related Information

- “ldap_add_control()—Create a Control and Insert it into the List of LDAP Server Controls” on page 10 — Add a new LDAP server control.
- “ldap_remove_control()—Remove a Control from the List of LDAP Server Controls” on page 151 — Remove a control from the list of LDAP server controls.
- “ldap_copy_controls()—Make a Copy of the List of LDAP Server Controls” on page 34 — Synchronously add an entry with controls.

API introduced: V5R3

ldap_is_ldap_url()—Verify LDAP URL

Syntax
#include <ldap.h>

int ldap_is_ldap_url(
    char *url)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The ldap_is_ldap_url() function is used to check a string to determine if it could be an LDAP URL. It can be used as a quick check for an LDAP URL.

Authorities and Locks
No i5/OS authority is required.

Parameters
url  (Input) Specifies a pointer to the URL string.

Return Value
non-zero  if url begins with "ldap://" or "ldaps://".
0  if url is not an LDAP URL.

Related Information
• "ldap_is_ldap_url_utf8()—Verify UTF-8 codepage LDAP URL" — Verify UTF-8 codepage LDAP URL.
• "ldap_url_parse()—Parse an LDAP URL” on page 214 — Break up an LDAP URL string into its components.
• "ldap_url_search()—Perform an LDAP URL Search Operation” on page 217 — Asynchronously search using an LDAP URL.
• "ldap_url_search_st() — Perform an LDAP URL Search Operation (Synchronous)” on page 219 — Synchronously search using an LDAP URL.
• "ldap_url_search_st()—Perform an LDAP URL Search Operation (Timed Synchronous)” on page 221 — Synchronously search using an LDAP URL and a timeout.

API introduced: V4R3

Syntax
#include <ldap.h>

int ldap_is_ldap_url_utf8(
    char *url)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The ldap_is_ldap_url_utf8() function is used to check a UTF-8 codepage string to determine if it could be an LDAP URL. It can be used as a quick check for an LDAP URL.
Authorities and Locks
No i5/OS authority is required.

Parameters
url (Input) Specifies a pointer to the URL string.

Return Value
non-zero
   if url begins with "ldap://" or "ldaps://".
0     if url is not an LDAP URL.

Related Information
- "ldap_is_ldap_url()—Verify LDAP URL” on page 108 — Verify LDAP URL.
- "ldap_url_parse_utf8()—Parse a UTF8 codepage LDAP URL string” on page 215 — Break up a UTF-8 codepage LDAP URL string into its components.
- "ldap_url_search()—Perform an LDAP URL Search Operation” on page 217 — Asynchronously search using an LDAP URL.
- "ldap_url_search_s() — Perform an LDAP URL Search Operation (Synchronous)” on page 219 — Synchronously search using an LDAP URL.
- "ldap_url_search_st()—Perform an LDAP URL Search Operation (Timed Synchronous)” on page 221 — Synchronously search using an LDAP URL and a timeout.

API introduced: V5R4

ldap_memfree()—Free Memory Allocated by LDAP API

Syntax
#include <ldap.h>
void ldap_memfree(  
    char *mem)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The ldap_memfree() API is used to free storage that is allocated by some of the LDAP APIs. Refer to the specific LDAP API documentation to see which memory free API to use for any memory allocated.

Authorities and Locks
No i5/OS authority is required.

Parameters
mem (Input) Specifies the address of storage that was allocated by the LDAP library.
Return Value
NONE

Error Conditions
The ldap_memfree() API does not return an error code.

Error Messages
The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_memfree API.</td>
</tr>
</tbody>
</table>

Related Information
- "ldap_ber_free()—Free storage allocated by the LDAP library” on page 22 — Free the BerElement structure.
- "ldap_control_free()—Free storage allocated by the LDAP library” on page 33 — Free a single LDAPControl structure.
- "ldap_controls_free()—Free storage allocated by the LDAP library” on page 32 — Free an array of LDAPControl structures.
- "ldap_free_urldesc()—Free an LDAP URL Description” on page 86 — Free an LDAP URL Description
- "ldap_mods_free()—Free LDAP Modify Storage” on page 123 — Free an array of pointers to mod structures.
- "ldap_msgfree()—Free LDAP Result Message” on page 125 — Free the LDAPMessage structure.
- "ldap_server_free_list()—Free the List of LDAP Servers” on page 177 — Free the List of LDAP Servers
- "ldap_value_free()—Free Memory Allocated by ldap_get_values()” on page 223 — Free memory allocated by ldap_get_values
- "ldap_value_free_len()—Free Memory Allocated by ldap_get_values_len()” on page 224 — Free Memory Allocated by ldap_get_values_len

API introduced: V4R3

ldap_modify()—Perform an LDAP Modify Entry Request

Syntax
#include <ldap.h>

typedef struct ldapmod {
    int mod_op;
    char *mod_type;
    union {
        char **modv_strvals;
        struct berval **modv_bvals;
    } mod_vals;
} LDAPMod;

#define mod_values mod_vals.modv_strvals
#define mod_bvalues mod_vals.modv_bvals
int ldap_modify(  
    LDAP *ld,  
    const char *dn,  
    LDAPMod **mods)  

Default Public Authority: *USE  
Library Name/Service Program: QSYS/QGLDCLNT  
Threadsafe: Yes

The `ldap_modify()` API is an asynchronous request. The result of the operation can be obtained by a subsequent call to “ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156.

The `mod_op` field is used to specify the type of modification to perform and should be one of the following:

- **LDAP_MOD_ADD**: 0x00
- **LDAP_MOD_DELETE**: 0x01
- **LDAP_MOD_REPLACE**: 0x02

This field also indicates the type of values included in the `mod_vals` union. For binary data, you must also bitwise OR the operation type with **LDAP_MOD_BVALUES** (0x80). This indicates that the values are specified in a NULL-terminated array of `struct berval` structures. Otherwise, the `mod_values` will be used (that is, the values are assumed to be a NULL-terminated array of NULL-terminated character strings).

The `mod_type` field specifies the name of attribute to add, delete, or replace.

The `mod_vals` field specifies a pointer to a NULL-terminated array of values to add, modify or delete respectively. Only one of the `mod_values` or `mod_bvalues` variants should be used, with `mod_bvalues` being selected by ORing the `mod_op` field with the constant **LDAP_MOD_BVALUES**. `mod_values` is a NULL-terminated array of NULL-terminated strings and `mod_bvalues` is a NULL-terminated array of `berval` structures that can be used to pass binary values such as images.

For **LDAP_MOD_ADD** modifications, the given values are added to the entry, creating the attribute if necessary.

For **LDAP_MOD_DELETE** modifications, the given values are deleted from the entry, removing the attribute if no values remain. If the entire attribute is to be deleted, the `mod_values` field should be set to `NULL`. The server will return an error if the attribute doesn’t exist.

For **LDAP_MOD_REPLACE** modifications, the attribute will have the listed values after the modification, having been created if necessary, or removed if the `mod_values` field is `NULL`. The server will NOT return an error if the value doesn’t exist.

All modifications are performed in the order in which they are listed.

**Authorities and Locks**

No i5/OS authority is required. All authority checking is done by the LDAP server.

**Parameters**

- **ld** (Input) Specifies the LDAP pointer returned by a previous call to “ldap_init()—Perform an LDAP Initialization Operation” on page 105, “ldap_ssl_init — Initialize an SSL Connection.” on page 204, or “ldap_open() — Perform an LDAP Open Operation” on page 135.
- **dn** (Input) Specifies the Distinguished Name (DN) of the entry to be modified.
mods  (Input) Specifies a NULL-terminated array of modifications to make to the entry. Each element of the mods array is a pointer to an LDAPMod structure.

Return Value
Message ID of the Operation Initiated
if the request was successful. A subsequent call to “ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156, can be used to obtain the result of the modify.
-1 if the request was not successful.

Error Conditions
If ldap_modify() is not successful, ld_errno will be set to indicate the error. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values. Use “ldap_get_errno()—Retrieve Error Information” on page 91 function to retrieve the error information.

Error Messages
The following message may be sent from this function.

Message ID   Error Message Text
CPF3CF2 E    Error(s) occurred during running of ldap_modify API.

Related Information
• ”ldap_add()—Perform an LDAP Add Operation” on page 9 — Asynchronously add an entry.
• ”ldap_delete()—Perform an LDAP Delete Operation” on page 59 — Perform an LDAP Delete Operation.
• ”ldap_modify_s()—Perform an LDAP Modify Entry Request (Synchronous)” on page 118 — Synchronous modify to a directory entry.
• ”ldap_modify_ext()—Perform an LDAP Modify Entry Request with Controls” — Asynchronous modify to a directory entry with controls.
• ”ldap_modify_ext_s()—Perform an LDAP Modify Entry Request with Controls” on page 116 — Synchronous modify to a directory entry with controls.
• ”ldap_modrdn()—Perform an LDAP Modify RDN Request” on page 120 — Asynchronously modify the RDN of an entry.
• ”ldap_modrdn_s()—Perform an LDAP Modify RDN Request (Synchronous)” on page 122 — Synchronously modify the RDN of an entry.

API introduced: V4R3

ldap_modify_ext()—Perform an LDAP Modify Entry Request with Controls

Syntax
#include <ldap.h>

typedef struct ldapmod {
    int mod_op;
    char *mod_type;
    union {
        char **modv_strvals;
    }
} ldapmod;

Lightweight Directory Access Protocol (LDAP) APIs 113
```
struct berval **modv_bvals;
} mod_vals;
} LDAPMod;

#define mod_values mod_vals.modv_strvals
#define mod_bvalues mod_vals.modv_bvals

int ldap_modify_ext( LDAP *ld, const char *dn, LDAPMod **mods, LDAPControl **serverctrls, LDAPControl **clientctrls, int *msgidp)
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The `ldap_modify_ext()` routine initiates an asynchronous modify operation with controls. `dn` is the Distinguished name of the entry to modify, and `mods` is a NULL-terminated array of modifications to make to the entry. Each element of the `mods` array is a pointer to an LDAPMod structure.

The `mod_op` field is used to specify the type of modification to perform and should be one of the following:

- `LDAP_MOD_ADD` 0x00
- `LDAP_MOD_DELETE` 0x01
- `LDAP_MOD_REPLACE` 0x02

This field also indicates the type of values included in the `mod_vals` union. For binary data, you must also bitwise OR the operation type with `LDAP_MOD_BVALUES` (0x80). This indicates that the values are specified in a NULL-terminated array of struct berval structures. Otherwise, the `mod_values` will be used (that is, the values are assumed to be a NULL-terminated array of NULL-terminated character strings).

The `mod_type` field specifies the name of attribute to add, delete, or replace.

The `mod_vals` field specifies a pointer to a NULL-terminated array of values to add, replace, or delete. Only one of the `mod_values` or `mod_bvalues` variants should be used, with `mod_bvalues` being selected by ORing the `mod_op` field with the constant `LDAP_MOD_BVALUES`. `mod_values` is a NULL-terminated array of NULL-terminated strings and `mod_bvalues` is a NULL-terminated array of berval structures that can be used to pass binary values such as images.

For `LDAP_MOD_ADD` modifications, the given values are added to the entry, creating the attribute if necessary.

For `LDAP_MOD_DELETE` modifications, the given values are deleted from the entry, removing the attribute if no values remain. If the entire attribute is to be deleted, the `mod_values` field should be set to NULL. The server will return an error if the attribute doesn’t exist.

For `LDAP_MOD_REPLACE` modifications, the attribute will have the listed values after the modification, having been created if necessary, or removed if the `mod_vals` field is NULL. The server should NOT return an error if the value doesn’t exist.

All modifications are performed in the order in which they are listed.
Authorities and Locks

No i5/OS authority is required. All authority checking is done by the LDAP server.

Parameters

ld  (Input) Specifies the LDAP pointer returned by a previous call to “ldap_init()—Perform an LDAP Initialization Operation” on page 105, “ldap_ssl_init—Initialize an SSL Connection.” on page 204, or “ldap_open()—Perform an LDAP Open Operation” on page 135.

dn  (Input) Specifies the Distinguished Name of the entry to be modified.

mods  (Input) Specifies a NULL-terminated array of modifications to make to the entry. Each element of the mods array is a pointer to an LDAPMod structure.

serverctrls  (Input) Specifies a list of LDAP server controls. This parameter may be set to NULL. See LDAP Controls for more information about server controls.

clientctrls  (Input) Specifies a list of LDAP client controls. This parameter may be set to NULL. See LDAP Controls for more information about client controls.

msgidp  (output) This result parameter is set to the message id of the request if the ldap_modify_ext() call succeeds.

Return Value

LDAP_SUCCESS

if the request was successfully sent. If successful, ldap_modify_ext() places the message id of the request in *msgidp. A subsequent call to “ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156 can be used to obtain the result of the operation. Once the operation has completed, ldap_result() returns a result that contains the status of the operation (in the form of an error code). The error code indicates whether or not the operation completed successfully. The “ldap_parse_result()—Extract Information from Results” on page 144 API is used to check the error code in the result.

another LDAP error code

if the request was not successful.

Error Conditions

The ldap_modify_ext() API will return an LDAP error code if not successful. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values.

Error Messages

The following message may be sent from this function.

Message ID     Error Message Text
CPF3CF2 E     Error(s) occurred during running of ldap_modify_ext API.

Related Information

• “ldap_add_ext()—Perform an LDAP Add Operation with Controls” on page 11 — Asynchronously add an entry with controls.

• “ldap_delete_ext()—Perform an LDAP Delete Operation with Controls” on page 60 — Perform an LDAP delete operation with controls.
ldap_modify_ext_s()—Perform an LDAP Modify Entry Request with Controls

Syntax

```c
#include <ldap.h>

typedef struct ldapmod {
    int mod_op;
    char *mod_type;
    union {
        char **modv_strvals;
        struct berval **modv_bvals;
    } mod_vals;
} LDAPMod;

#define mod_values mod_vals.modv_strvals
#define mod_bvalues mod_vals.modv_bvals

int ldap_modify_ext_s(
    LDAP *ld,
    const char *dn,
    LDAPMod **mods,
    LDAPControl **serverctrls,
    LDAPControl **clientctrls)
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The `ldap_modify_ext_s()` API initiates a synchronous modify operation with controls. `dn` is the Distinguished name of the entry to modify, and `mods` is a NULL-terminated array of modifications to make to the entry. Each element of the mods array is a pointer to an LDAPMod structure.

The `mod_op` field is used to specify the type of modification to perform and should be one of the following:

- `LDAP_MOD_ADD` 0x00
- `LDAP_MOD_DELETE` 0x01
- `LDAP_MOD_REPLACE` 0x02
This field also indicates the type of values included in the mod_vals union. For binary data, you must also bitwise OR the operation type with LDAP_MOD_BVALUES (0x80). This indicates that the values are specified in a NULL-terminated array of struct berval structures. Otherwise, the mod_values will be used (that is, the values are assumed to be a NULL-terminated array of NULL-terminated character strings).

The mod_type field specifies the name of attribute to add, delete, or replace.

The mod_vals field specifies a pointer to a NULL-terminated array of values to add, modify or delete respectively. Only one of the mod_values or mod_bvalues variants should be used, with mod_bvalues being selected by ORing the mod_op field with the constant LDAP_MOD_BVALUES. mod_values is a NULL-terminated array of NULL-terminated strings and mod_bvalues is a NULL-terminated array of berval structures that can be used to pass binary values such as images.

For LDAP_MOD_ADD modifications, the given values are added to the entry, creating the attribute if necessary.

For LDAP_MOD_DELETE modifications, the given values are deleted from the entry, removing the attribute if no values remain. If the entire attribute is to be deleted, the mod_values field should be set to NULL. The server will return an error if the attribute doesn’t exist.

For LDAP_MOD_REPLACE modifications, the attribute will have the listed values after the modification, having been created if necessary, or removed if the mod_values field is NULL. The server will NOT return an error if the value doesn’t exist.

All modifications are performed in the order in which they are listed.

Authorities and Locks
No i5/OS authority is required. All authority checking is done by the LDAP server.

Parameters

\textit{ld} \quad (Input) Specifies the LDAP pointer returned by a previous call to "\texttt{ldap_init()}—Perform an LDAP Initialization Operation” on page 105, "\texttt{ldap_ssl_init}—Initialize an SSL Connection.” on page 204, or "\texttt{ldap_open()}—Perform an LDAP Open Operation” on page 135.

\textit{dn} \quad (Input) Specifies the Distinguished Name of the entry to be modified.

\textit{mods} \quad (Input) Specifies a NULL-terminated array of modifications to make to the entry. Each element of the mods array is a pointer to an LDAPMod structure.

\textit{serverctrls} \quad (Input) Specifies a list of LDAP server controls. This parameter may be set to null. See LDAP Controls for more information about server controls.

\textit{clientctrls} \quad (Input) Specifies a list of LDAP client controls. This parameter may be set to null. See LDAP Controls for more information about client controls.

Return Value

LDAP_SUCCESS
if the request was successfully sent.

LDAP error code
if the request was not successfully sent.
Error Conditions
The `ldap_modify_ext_s()` will return an LDAP error code if not successful. See "LDAP Client API Error Conditions" on page 300 for possible LDAP error code values.

Error Messages
The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_modify_ext_s API.</td>
</tr>
</tbody>
</table>

Related Information
- "ldap_add_ext_s()—Perform an LDAP Add Operation with Controls (Synchronous)“ on page 13 — Synchronously add an entry with controls.
- "ldap_delete_ext_s()—Perform an LDAP Delete Operation with Controls” on page 61 — Perform an LDAP Delete Operation with Controls (Synchronous)
- "ldap_modify()—Perform an LDAP Modify Entry Request“ on page 111 — Asynchronous modify to a directory entry.
- "ldap_modify_s()—Perform an LDAP Modify Entry Request (Synchronous)“ — Synchronous modify to a directory entry.
- "ldap_modify_ext()—Perform an LDAP Modify Entry Request with Controls” on page 113 — Asynchronous modify to a directory entry with controls.
- "ldap_modrdn()—Perform an LDAP Modify RDN Request“ on page 120 — Asynchronously modify the RDN of an entry.
- "ldap_modrdn_s()—Perform an LDAP Modify RDN Request (Synchronous)“ on page 122 — Synchronously modify the RDN of an entry.

API introduced: V4R5

_Supported by IBM Systems - iSeries: Lightweight Directory Access Protocol (LDAP) APIs._

**ldap_modify_s()—Perform an LDAP Modify Entry Request (Synchronous)**

Syntax
```c
#include <ldap.h>

typedef struct ldapmod {
    int mod_op;
    char *mod_type;
    union {
        char **modv_strvals;
        struct berval **modv_bvals;
    } mod_vals;
} LDAPMod;

#define mod_values mod_vals.modv_strvals
#define mod_bvalues mod_vals.modv_bvals

int ldap_modify_s(
    LDAP *ld,
    const char *dn,
    LDAPMod **mods)
```
The `ldap_modify_s()` performs a synchronous request.

The `mod_op` field is used to specify the type of modification to perform and should be one of the following:

- `LDAP_MOD_ADD` 0x00
- `LDAP_MOD_DELETE` 0x01
- `LDAP_MOD_REPLACE` 0x02

This field also indicates the type of values included in the `mod_vals` union. For binary data, you must also bitwise OR the operation type with `LDAP_MOD_BVALUES` (0x80). This indicates that the values are specified in a NULL-terminated array of struct `berval` structures. Otherwise, the `mod_values` will be used (that is, the values are assumed to be a NULL-terminated array of NULL-terminated character strings).

The `mod_type` field specifies the name of attribute to add, delete, or replace.

The `mod_vals` field specifies a pointer to a NULL-terminated array of values to add, modify or delete respectively. Only one of the `mod_values` or `mod_bvalues` variants should be used, with `mod_bvalues` being selected by ORing the `mod_op` field with the constant `LDAP_MOD_BVALUES`. `mod_values` is a NULL-terminated array of NULL-terminated strings and `mod_bvalues` is a NULL-terminated array of berval structures that can be used to pass binary values such as images.

For `LDAP_MOD_ADD` modifications, the given values are added to the entry, creating the attribute if necessary.

For `LDAP_MOD_DELETE` modifications, the given values are deleted from the entry, removing the attribute if no values remain. If the entire attribute is to be deleted, the `mod_values` field should be set to NULL. The server will return an error if the attribute doesn’t exist.

For `LDAP_MOD_REPLACE` modifications, the attribute will have the listed values after the modification, having been created if necessary, or removed if the `mod_values` field is NULL. The server will NOT return an error if the value doesn’t exist.

All modifications are performed in the order in which they are listed.

**Authorities and Locks**

No i5/OS authority is required. All authority checking is done by the LDAP server.

**Parameters**

- `ld` (Input) Specifies the LDAP pointer returned by a previous call to "`ldap_init()—Perform an LDAP Initialization Operation" on page 105" or "`ldap_ssl_init —Initialize an SSL Connection." on page 204, or "`ldap_open()—Perform an LDAP Open Operation" on page 135."
- `dn` (Input) Specifies the Distinguished Name of the entry to be modified.
- `mods` (Input) Specifies a NULL-terminated array of modifications to make to the entry. Each element of the `mods` array is a pointer to an LDAPMod structure.
Return Value
LDAP_SUCCESS
if the request was successful.

another LDAP error
if the request was not successful.

Error Conditions
The ldap_modify_s() API will return an LDAP error code if not successful. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values.

Error Messages
The following message may be sent from this function.

Message ID   Error Message Text
CPF3CF2 E    Error(s) occurred during running of ldap_modify_s API.

Related Information
- “ldap_add_s()—Perform an LDAP Add Operation (Synchronous)” on page 14 — Perform an LDAP add operation (synchronous).
- “ldap_delete_s()—Perform an LDAP Delete Operation (Synchronous)” on page 63 — Perform an LDAP delete operation (synchronous).
- “ldap_modify_s()—Perform an LDAP Modify Entry Request (Synchronous)” on page 118 — Perform an LDAP modify entry request.
- “ldap_modify_ext()—Perform an LDAP Modify Entry Request with Controls” on page 113 — Asynchronous modify to a directory entry with controls.
- “ldap_modify_ext_s()—Perform an LDAP Modify Entry Request with Controls” on page 116 — Synchronous modify to a directory entry with controls.
- “ldap_modrdn()—Perform an LDAP Modify RDN Request” — Asynchronously modify the RDN of an entry.
- “ldap_modrdn_s()—Perform an LDAP Modify RDN Request (Synchronous)” on page 122 — Synchronously modify the RDN of an entry.

API introduced: V4R3
The `ldap_modrdn()` function is used to perform an LDAP modify relative distinguished name (RDN) operation. The function takes the distinguished name of the entry whose RDN is to be changed, and `newrdn`, the new RDN to give the entry. The `deleteoldrdn` parameter is used as a boolean value to indicate whether the old RDN values should be deleted from the entry or not.

`ldap_modrdn()` performs an asynchronous request. The result of the operation can be obtained by a subsequent call to “ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156.

In LDAP V2, the `ldap_modrdn()` and “ldap_modrdn_s()—Perform an LDAP Modify RDN Request (Synchronous)” on page 122 APIs were used to change the name of an LDAP entry. They could only be used to change the least significant component of a name (the RDN or relative distinguished name). LDAP V3 provides the Modify DN protocol operation that allows more general name change access. The “ldap_rename()—Asynchronously Rename an Entry” on page 152 and “ldap_rename_s()—Synchronously Rename an Entry” on page 154 routines are used to change the name of an entry, and the use of the `ldap_modrdn()` and “ldap_modrdn_s()—Perform an LDAP Modify RDN Request (Synchronous)” on page 122 routines are deprecated.

**Authorities and Locks**

No i5/OS authority is required. All authority checking is done by the LDAP server.

**Parameters**

```c
ld (Input) Specifies the LDAP pointer returned by a previous call to "ldap_init()—Perform an LDAP Initialization Operation” on page 105, "ldap_ssl_init—Initialize an SSL Connection.” on page 204, or "ldap_open()—Perform an LDAP Open Operation” on page 135.
```

```c
dn (Input) Specifies the DN of the entry whose RDN is to be changed.
```

```c
newrdn (Input) Specifies the new RDN to be given to the entry.
```

```c
deleteoldrdn (Input) Specifies a boolean value. When set to 1, the old RDN value is to be deleted from the entry. When set to 0, the old RDN value should be retained as a non-distinguished value.
```

**Return Value**

Message ID of the Operation Initiated

If the request was successful. A subsequent call to “ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156 can be used to obtain the result of the modify.

-1 if the request was not successful.

**Error Conditions**

If `ldap_modrdn()` is not successful, `ld_errno` will be set to indicate the error. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values. Use the “ldap_get_errno()—Retrieve Error Information” on page 91 function to retrieve the error information.

**Error Messages**

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_modrdn API.</td>
</tr>
</tbody>
</table>
Related Information

- "ldap_add()—Perform an LDAP Add Operation" on page 9 — Perform an LDAP add operation.
- "ldap_delete()—Perform an LDAP Delete Operation" on page 59 — Perform an LDAP delete operation.
- "ldap_modify()—Perform an LDAP Modify Entry Request" on page 111 — Asynchronous modify to a directory entry.
- "ldap_modrdn_s()—Perform an LDAP Modify RDN Request (Synchronous)" — Synchronously modify the RDN of an entry.
- "ldap_rename()—Asynchronously Rename an Entry" on page 152 — Asynchronously rename an entry.

API introduced: V4R3

```
#include <ldap.h>

int ldap_modrdn_s(  
    LDAP *ld,  
    const char *dn,  
    const char *newrdn,  
    int deleteoldrdn)
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafte: Yes

The `ldap_modrdn_s()` function is used to perform an LDAP modify relative distinguished name (RDN) operation. The function takes the distinguished name of the entry whose RDN is to be changed, and `newrdn`, the new RDN to give the entry. The `deleteoldrdn` parameter is used as a boolean value to indicate whether the old RDN values should be deleted from the entry or not.

`ldap_modrdn_s()` performs a synchronous request.

In LDAP V2, the `ldap_modrdn()` and "ldap_modrdn_s()—Perform an LDAP Modify RDN Request (Synchronous)" APIs were used to change the name of an LDAP entry. They could only be used to change the least significant component of a name (the RDN or relative distinguished name). LDAP V3 provides the Modify DN protocol operation that allows more general name change access. The "ldap_rename()—Asynchronously Rename an Entry" on page 152 and "ldap_rename_s()—Synchronously Rename an Entry" on page 154 routines are used to change the name of an entry, and the use of the `ldap_modrdn()` and "ldap_modrdn_s()—Perform an LDAP Modify RDN Request (Synchronous)" routines are deprecated.

Authorities and Locks

No i5/OS authority is required. All authority checking is done by the LDAP server.

Parameters

- `ld` (Input) Specifies the LDAP pointer returned by a previous call to "ldap_init()—Perform an LDAP Initialization Operation" on page 105, "ldap_ssl_init—Initialize an SSL Connection." on page 204, or "ldap_open()—Perform an LDAP Open Operation" on page 135.
Specifies the DN of the entry whose RDN is to be changed.

Specifies the new RDN to be given to the entry.

Specifies a boolean value. When set to 1, the old RDN value is to be deleted from the entry. When set to 0, the old RDN value should be retained as a non-distinguished value.

Return Value
LDAP_SUCCESS
  if the request was successful.

another LDAP error
  if the request was not successful.

Error Conditions
The ldap_modrdn_s() will return an LDAP error code if not successful. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values.

Error Messages
The following message may be sent from this function.

Message ID   Error Message Text
CPF3CF2 E   Error(s) occurred during running of ldap_modrdn_s API.

Related Information
- “ldap_add()—Perform an LDAP Add Operation” on page 9 — Perform an LDAP add operation.
- “ldap_delete()—Perform an LDAP Delete Operation” on page 59 — Perform an LDAP delete operation.
- “ldap_modify()—Perform an LDAP Modify Entry Request” on page 111 — Asynchronous modify to a directory entry.
- “ldap_modrdn()—Perform an LDAP Modify RDN Request” on page 120 — Asynchronously modify the RDN of an entry.
- “ldap_rename_s()—Synchronously Rename an Entry” on page 154 — Synchronously rename an entry.

API introduced: V4R3

**ldap_mods_free()**—Free LDAP Modify Storage

Syntax
#include <ldap.h>

void ldap.mods_free(  
    LDAPMod **mods,     
    int freemods)
The `ldap_mods_free()` function is used to free storage associated with the "ldap_modify()—Perform an LDAP Modify Entry Request" on page 111 and related LDAP APIs.

`ldap_mods_free()` can be used to free each element of a NULL-terminated array of modification structures. If `freemods` is nonzero, the `mods` pointer itself is freed, otherwise freeing `mods` is left to the caller.

**Authorities and Locks**

No i5/OS authority is required.

**Parameters**

- `mods` (Input) Specifies a NULL-terminated array of modifications to make to the entry. Each element of the `mods` array is a pointer to an LDAPMod structure.

- `freemods` (Input) Specifies whether or not the `mods` pointer is to be freed in addition to the NULL-terminated array of LDAPMod structures.

**Return Value**

None

**Error Conditions**

The `ldap_mods_free()` API does not return an error code.

**Error Messages**

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_mods_free API.</td>
</tr>
</tbody>
</table>

**Related Information**

- "ldap_ber_free()—Free storage allocated by the LDAP library“ on page 22 — Free the BerElement structure.
- "ldap_control_free()—Free storage allocated by the LDAP library“ on page 33 — Free a single LDAPControl structure.
- "ldap_controlls_free()—Free storage allocated by the LDAP library“ on page 32 — Free an array of LDAPControl structures.
- "ldap_free_urldesc()—Free an LDAP URL Description“ on page 86 — Free an LDAP URL Description
- "ldap_mods_free()—Free LDAP Modify Storage” on page 123 — Free an array of pointers to mod structures.
- "ldap_memfree()—Free Memory Allocated by LDAP API” on page 110 — Free storage allocated by the LDAP client library.
- "ldap_modify()—Perform an LDAP Modify Entry Request” on page 111 — Perform an LDAP modify entry request.
- "ldap_msgfree()—Free LDAP Result Message” on page 125 — Free the LDAPMessage structure.
ldap_msgfree()—Free LDAP Result Message

Syntax
#include <ldap.h>

int ldap_msgfree(
    LDAPMessage *msg)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The ldap_msgfree() routine is used to free the memory allocated for an LDAP message by
“ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156,
“ldap_search_s()—Perform an LDAP Search Operation (Synchronous)” on page 171, “ldap_search_ext_s — Synchronously Search the Directory Using Controls” on page 168 or “ldap_search_st()—Perform an LDAP Search Operation (Timed Synchronous)” on page 173. It takes a pointer to the result to be freed and returns the type of the message it freed.

Authorities and Locks
No i5/OS authority is required.

Parameters
msg (Input) Specifies pointer to the memory allocated for an LDAP message by
“ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156,
“ldap_search_s()—Perform an LDAP Search Operation (Synchronous)” on page 171, “ldap_search_ext_s — Synchronously Search the Directory Using Controls” on page 168 or “ldap_search_st()—Perform an LDAP Search Operation (Timed Synchronous)” on page 173.

Return Values
Message Type
the type of the message freed.
ZERO if the input pointer to LDAPMessage structure is NULL.

Error Conditions
The ldap_msgfree() API returns ZERO if the input pointer to LDAPMessage structure is NULL.

Error Messages
The following message may be sent from this function.
Message ID     Error Message Text
CPF3CF2 E     Error(s) occurred during running of ldap_msgfree API.

Related Information

• "ldap_ber_free()—Free storage allocated by the LDAP library” on page 22 — Free the BerElement structure.
• "ldap_control_free()—Free storage allocated by the LDAP library” on page 33 — Free a single LDAPControl structure.
• "ldap_controls_free()—Free storage allocated by the LDAP library” on page 32 — Free an array of LDAPControl structures.
• "ldap_free_urldesc()—Free an LDAP URL Description” on page 86 — Free an LDAP URL Description
• "ldap_memfree()—Free Memory Allocated by LDAP API” on page 110 — Free storage allocated by the LDAP client library.
• "ldap_mods_free()—Free LDAP Modify Storage” on page 123 — Free an array of pointers to mod structures.
• "ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156 — Retrieve result of an asynchronous LDAP operation.
• "ldap_search_ext_s — Synchronously Search the Directory Using Controls” on page 168 — Synchronously search the directory using controls.
• "ldap_search_s()—Perform an LDAP Search Operation (Synchronous)” on page 171 — Perform an LDAP search operation (synchronous).
• "ldap_search_st()—Perform an LDAP Search Operation (Timed Synchronous)” on page 173 — Perform an LDAP search operation (timed synchronous).
• "ldap_server_free_list()—Free the List of LDAP Servers” on page 177 — Free the List of LDAP Servers
• "ldap_value_free()—Free Memory Allocated by ldap_get_values()” on page 223 — Free memory allocated by ldap_get_values
• "ldap_value_free_len()—Free Memory Allocated by ldap_get_values_len()” on page 224 — Free Memory Allocated by ldap_get_values_len

API introduced: V4R3

ldap_msgid()—Retrieve the Message ID Associated with an LDAP Message

Syntax
#include <ldap.h>

int ldap_msgid(
    LDAPMessage *msg)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The ldap_msgid() routine returns the message ID associated with an LDAP message. Use ldap_msgid() to match the result(s) of an asynchronous operation with the original operation.
Authorities and Locks
No i5/OS authority is required.

Parameters
msg  (Input) Specifies a pointer to a result, as returned from “ldap_first_message()—Retrieve First LDAP Message” on page 82, “ldap_next_message()—Retrieve Next LDAP Message” on page 132, “ldap_first_entry()—Retrieve First LDAP Entry” on page 81, “ldap_next_entry()—Retrieve Next LDAP Entry” on page 131, “ldap_first_reference()—Retrieve First Continuation Reference in a Chain of Search Results” on page 84, or “ldap_next_reference()—Retrieve the next Continuation Reference in a Chain of Search Results” on page 134.

Return Value
Message ID
if the call was successful.
ZERO  if the input pointer to LDAPMessage structure is NULL.

Error Conditions
ldap_msgid() returns ZERO if the input pointer to LDAPMessage structure is NULL.

Error Messages
The following message may be sent from this function.

Message ID      Error Message Text
CPF3CF2   E      Error(s) occurred during running of ldap_msgid API.

Related Information
• “ldap_add()—Perform an LDAP Add Operation” on page 9 — Perform an LDAP add operation.
• “ldap_add_ext()—Perform an LDAP Add Operation with Controls” on page 11 — Perform an LDAP add operation with controls.
• “ldap_bind()—Perform an LDAP Bind Request” on page 23 — Perform an LDAP bind request.
• “ldap_compare()—Perform an LDAP Compare Operation” on page 26 — Perform an LDAP compare operation.
• “ldap_compare_ext()—Perform an LDAP Compare Operation with Controls” on page 28 — Perform an LDAP compare operation with controls.
• “ldap_delete()—Perform an LDAP Delete Operation” on page 59 — Perform an LDAP delete operation.
• “ldap_delete_ext()—Perform an LDAP Delete Operation with Controls” on page 60 — Perform an LDAP delete operation with controls.
• “ldap_extended_operation()—Perform extended operations.” on page 75 — Perform extended operations.
• “ldap_first_entry()—Retrieve First LDAP Entry” on page 81 — Retrieve first LDAP entry.
• “ldap_first_message()—Retrieve First LDAP Message” on page 82 — Retrieve First LDAP message.
• “ldap_first_reference()—Retrieve First Continuation Reference in a Chain of Search Results” on page 84 — Return first continuation reference in a chain of search results.
• “ldap_modify()—Perform an LDAP Modify Entry Request” on page 111 — Perform an LDAP modify entry request.
• “ldap_modify_ext()—Perform an LDAP Modify Entry Request with Controls” on page 113 — Perform an LDAP modify entry request with controls.
• “ldap_modrdn()—Perform an LDAP Modify RDN Request” on page 120 — Perform an LDAP modify RDN request.
• “ldap_msgtype()—Retrieve the Type of an LDAP Message” — Returns the type of an LDAP message.
• “ldap_next_entry()—Retrieve Next LDAP Entry” on page 131 — Retrieve next LDAP entry.
• “ldap_next_message()—Retrieve Next LDAP Message” on page 132 — Retrieve Next LDAP message.
• “ldap_next_reference()—Retrieve the next Continuation Reference in a Chain of Search Results” on page 134 — Retrieve next continuation reference in a chain of search results.
• “ldap_rename()—Asynchronously Rename an Entry” on page 152 — Asynchronously rename an entry.
• “ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156 — Wait for result from an asynchronous operation.
• “ldap_sasl_bind()—Perform an LDAP SASL Bind Request” on page 159 — Perform an LDAP SASL bind request.
• “ldap_search()—Perform an LDAP Search Operation” on page 164 — Perform an LDAP search operation.
• “ldap_search_ext —Asynchronously Search the Directory Using Controls” on page 166 — Asynchronously search the directory using controls.
• “ldap_simple_bind()—Perform a Simple LDAP Bind Request” on page 196 — Perform a simple LDAP bind request.

API introduced: V4R5

Top | “Lightweight Directory Access Protocol (LDAP) APIs,” on page 1 | APIs by category

**ldap_msgtype()—Retrieve the Type of an LDAP Message**

Syntax

```c
#include <ldap.h>

int ldap_msgtype(
    LDAPMessage    *msg)
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The `ldap_msgtype()` API returns the type of an LDAP message.

**Authorities and Locks**

No i5/OS authority is required.

**Parameters**

`msg` (Input) Specifies a pointer to a result, as returned from

- “ldap_first_message()—Retrieve First LDAP Message” on page 82
- “ldap_next_message()—Retrieve Next LDAP Message” on page 132
- “ldap_first_entry()—Retrieve First LDAP Entry” on page 81
- “ldap_next_entry()—Retrieve Next LDAP Entry” on page 131
- “ldap_first_reference()—Retrieve First Continuation Reference in a Chain of Search Results” on page 84
- “ldap_next_reference()—Retrieve the next Continuation Reference in a Chain of Search Results” on page 134
Return Value

Message Type
if the call was successful. Message types are as follows:

- LDAP_RES_BIND (0x61): Result of an LDAP bind operation.
- LDAP_RES_SEARCH_ENTRY (0x64): An entry.
- LDAP_RES_SEARCH_RESULT (0x65): Result of an LDAP search operation (LDAP v3).
- LDAP_RES_MODIFY (0x67): Result of an LDAP modify operation.
- LDAP_RES_ADD (0x69): Result of an LDAP add operation.
- LDAP_RES_DELETE (0x6b): Result of an LDAP delete operation.
- LDAP_RES_MODRDN (0x6d): Result of an LDAP modrdn operation.
- LDAP_RES_COMPARE (0x6f): Result of an LDAP compare operation.
- LDAP_RES_SEARCH_REFERENCE (0x73): A search reference.
- LDAP_RES_EXTENDED (0x78): Result of an LDAP extended operation (LDAP v3).
- LDAP_RES_REFERRAL (0xa3): A referral.

ZERO if the input pointer to LDAPMessage structure is NULL.

Error Conditions
The ldap_msgtype() API returns ZERO if the input pointer to LDAPMessage structure is NULL.

Error Messages
The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_msgtype API.</td>
</tr>
</tbody>
</table>

Related Information

- "ldap_first_entry()—Retrieve First LDAP Entry” on page 81 — Retrieve first LDAP entry.
- "ldap_first_message()—Retrieve First LDAP Message” on page 82 — Retrieve first LDAP message.
- "ldap_first_reference()—Retrieve First Continuation Reference in a Chain of Search Results” on page 84 — Retrieve first continuation reference in a chain of search results.
- "ldap_msgid()—Retrieve the Message ID Associated with an LDAP Message” on page 126 — Returns the ID of an LDAP message.
- "ldap_next_message()—Retrieve Next LDAP Message” on page 132 — Retrieve next LDAP message.
- "ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156 — Wait for result from an asynchronous operation.

API introduced: V4R5

ldap_next_attribute()—Retrieve Next Attribute in an Entry

Syntax
The `ldap_next_attribute()` function returns the next attribute in an entry.

The `ldap_next_attribute()` function takes an entry returned by "`ldap_first_entry()`—Retrieve First LDAP Entry" on page 81 or "`ldap_next_entry()`—Retrieve Next LDAP Entry" on page 131 and returns a pointer to a buffer containing the next attribute type in the entry. This string must be freed when its use is completed using "`ldap_memfree()`—Free Memory Allocated by LDAP API" on page 110.

The "`ldap_first_attribute()`—Retrieve First Attribute in an Entry" on page 79 and `ldap_next_attribute()` functions are used to step through the attributes in an LDAP entry.

**Authorities and Locks**

No i5/OS authority is required.

**Parameters**

- **ld** (Input) Specifies the LDAP pointer returned by a previous call to “`ldap_init()`—Perform an LDAP Initialization Operation” on page 105, “`ldap_ssl_init`—Initialize an SSL Connection.” on page 204, or "`ldap_open()`—Perform an LDAP Open Operation” on page 135.

- **entry** (Input) The attribute information as returned by "`ldap_first_entry()`—Retrieve First LDAP Entry” on page 81 or "`ldap_next_entry()`—Retrieve Next LDAP Entry” on page 131.

- **berptr** (Input/Output) This parameter specifies a pointer to a BerElement that was allocated by "`ldap_first_attribute()`—Retrieve First Attribute in an Entry” on page 79 to keep track of the current position. The BerElement structure is opaque to the application. The caller should free `berptr` using "`ldap_ber_free()`—Free storage allocated by the LDAP library” on page 22 when finished.

**Return Value**

- Pointer to a buffer containing the next attribute type in the entry if the request was successful.

- NULL When there are no attributes left to be retrieved.

**Error Conditions**

If `ldap_next_attribute()` is not successful, NULL is returned, and `ld_errno` will be set to indicate the error. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values. Use "`ldap_get_errno()`—Retrieve Error Information” on page 91 function to retrieve the error information. It is left to the user to free outstanding BerElements using "`ldap_first_entry()`—Retrieve First LDAP Entry” on page 81.

**Error Messages**

The following message may be sent from this function.
Error(s) occurred during running of ldap_next_attribute API.

Related Information

- "ldap_first_attribute()—Retrieve First Attribute in an Entry" on page 79 — Retrieve first attribute in an entry.
- "ldap_first_entry()—Retrieve First LDAP Entry” on page 81 — Retrieve first LDAP entry.
- "ldap_next_entry()—Retrieve Next LDAP Entry” — Retrieve next LDAP entry.
- "ldap_count_attributes()—Retrieve Count of Attributes for an LDAP Entry” on page 35 — Retrieve count of attributes for an LDAP entry.
- "ldap_get_values()—Retrieve a Set of Attribute Values from an Entry” on page 102 — Retrieve a set of attribute values from an entry.
- "ldap_get_values_len()—Retrieve a Set of Binary Attribute Values” on page 104 — Retrieve a set of binary attribute values.

API introduced: V4R3

ldap_next_entry()—Retrieve Next LDAP Entry

Syntax

#include <ldap.h>

LDAPMessage *ldap_next_entry(
    LDAP *ld,
    LDAPMessage *entry)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The ldap_next_entry() function takes the result from a previous call to "ldap_first_entry()—Retrieve First LDAP Entry” on page 81 or ldap_next_entry() and returns a pointer to the next entry in a chain of results.

The entry returned by ldap_next_entry() can be used by functions such as "ldap_get_dn()—Retrieve the Distinguished Name of an Entry” on page 89, "ldap_first_attribute()—Retrieve First Attribute in an Entry” on page 79 and "ldap_get_values()—Retrieve a Set of Attribute Values from an Entry” on page 102 as well as other functions to obtain additional information about the entry.

Authorities and Locks
No i5/OS authority is required.

Parameters

ld (Input) Specifies the LDAP pointer returned by a previous call to "ldap_init()—Perform an LDAP Initialization Operation” on page 105, "ldap_ssl_init — Initialize an SSL Connection.” on page 204, or "ldap_open()—Perform an LDAP Open Operation” on page 135.

entry (Input) Specifies a pointer to an entry returned on a previous call to "ldap_first_entry()—Retrieve First LDAP Entry” on page 81 or ldap_next_entry().
Return Value

Pointer to the next entry in the result if the request was successful.

NULL When there are no attributes left to be retrieved.

Error Conditions

If ldap_next_entry() is not successful, NULL is returned, ld_errno will be set to indicate the error. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values. Use ”ldap_get_errno()—Retrieve Error Information” on page 91 function to retrieve the error information.

Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_next_entry API.</td>
</tr>
</tbody>
</table>

Related Information

- ”ldap_first_entry()—Retrieve First LDAP Entry” on page 81 — Return first entry in a chain of search results.
- ”ldap_count_entries()—Retrieve Count of LDAP Entries” on page 36 — Return number of entries in a chain of search results.
- ”ldap_get_entry_controls_np()—Extract Server Controls from an Entry” on page 90 — Extract server controls from an entry.
- ”ldap_first_reference()—Retrieve First Continuation Reference in a Chain of Search Results” on page 84 — Return first continuation reference in a chain of search results.
- ”ldap_count_references()—Count continuation references in a chain of search results” on page 39 — Return number of continuation reference in a chain of search results.
- ”ldap_parse_reference_np()—Extract Information from a Continuation Reference” on page 143 — Extract information from a continuation reference.

API introduced: V4R3

---

ldap_next_message()—Retrieve Next LDAP Message

Syntax

#include <ldap.h>

LDAPMessage *ldap_next_message(
    LDAP *ld,
    LDAPMessage *msg)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes
The `ldap_next_message()` function is used to step through the list of messages in a result chain, as returned by "ldap_result()—Retrieve Result of an Asynchronous LDAP Operation" on page 156 and "ldap_first_message()—Retrieve First LDAP Message" on page 82. It is used to return a pointer to the next message from the list.

**Authorities and Locks**
No i5/OS authority is required.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ld</code></td>
<td>(Input) Specifies the LDAP pointer returned by a previous call to &quot;ldap_init()—Perform an LDAP Initialization Operation&quot; on page 105, &quot;ldap_ssl_init—Initialize an SSL Connection.&quot; on page 204, or &quot;ldap_open()—Perform an LDAP Open Operation&quot; on page 135.</td>
</tr>
<tr>
<td><code>msg</code></td>
<td>(Input) Specifies the message returned by a previous call to &quot;ldap_first_message()—Retrieve First LDAP Message&quot; on page 82 or <code>ldap_next_message()</code>.</td>
</tr>
</tbody>
</table>

**Return Value**

`LDAPMessage *`<br>pointer to the next message in list.<br>
`NULL` when no more messages exist in the result set to be returned or if an error occurs.

**Error Conditions**

If `ldap_next_message()` is not successful, `ld_errno` will be set to indicate the error. See "LDAP Client API Error Conditions" on page 300 for possible LDAP error code values. Use the "ldap_get_errno()—Retrieve Error Information" on page 91 function to retrieve the error information.

**Error Messages**

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of <code>ldap_next_message</code> API.</td>
</tr>
</tbody>
</table>

**Related Information**

- "ldap_count_messages()—Count messages in a result chain” on page 38 — Return the number of messages in a result chain.
- "ldap_first_entry()—Retrieve First LDAP Entry” on page 81 — Retrieve first LDAP entry.
- "ldap_first_message()—Retrieve First LDAP Message” on page 82 — Retrieve first LDAP message.
- "ldap_first_reference()—Retrieve First Continuation Reference in a Chain of Search Results” on page 84 — Return first continuation reference in a chain of search results.
- "ldap_msgfree()—Free LDAP Result Message” on page 125 — Free LDAP Result Message.
- "ldap_msgid()—Retrieve the Message ID Associated with an LDAP Message” on page 126 — Retrieve Message ID Associated with an LDAP Message.
- "ldap_msgtype()—Retrieve the Type of an LDAP Message” on page 128 — Retrieve Type of an LDAP Message.
- "ldap_result2error()—Retrieve LDAP Error Information” on page 158 — Retrieve LDAP Error Information.
API introduced: V4R5

```
Top| "Lightweight Directory Access Protocol (LDAP) APIs," on page 1 | APIs by category
```

### ldap_next_reference()—Retrieve the next Continuation Reference in a Chain of Search Results

**Syntax**

```
#include <ldap.h>

LDAPMessage *ldap_next_reference(
    LDAP *ld,
    LDAPMessage *result)
```

**Default Public Authority:** *USE  
**Library Name/Service Program:** QSYS/QGLDCLNT  
**Threadsafe:** Yes

The `ldap_next_reference()` function is used to return the next continuation reference from the search result chain.

### Authorities and Locks

No i5/OS authority is required.

### Parameters

- **ld** *(Input)* Specifies the LDAP pointer returned by a previous call to ["ldap_init()—Perform an LDAP Initialization Operation" on page 105](#) or ["ldap_ssl_init — Initialize an SSL Connection." on page 204](#), or ["ldap_open()—Perform an LDAP Open Operation" on page 135](#).

- **result** *(Input)* Specifies the result returned by a call to ldap_result() or one of the synchronous search routines ["ldap_search_s()—Perform an LDAP Search Operation (Synchronous)" on page 171](#), ["ldap_search_st()—Perform an LDAP Search Operation (Timed Synchronous)" on page 173](#), or ["ldap_search_ext_s — Synchronously Search the Directory Using Controls" on page 168](#).

- **ref** *(Input)* Specifies a pointer to a search continuation reference returned on a previous call to ["ldap_first_reference()—Retrieve First Continuation Reference in a Chain of Search Results" on page 84](#) or [ldap_next_reference()](#).

### Return Value

- `LDAPMessage *` pointer to the next continuation reference.

- `NULL` when no more continuation references exist in the result set to be returned.

### Error Conditions

If `ldap_next_reference()` is not successful, `ld_errno` will be set to indicate the error. See ["LDAP Client API Error Conditions" on page 300](#) for possible LDAP error code values. Use the ["ldap_get_errno()—Retrieve Error Information" on page 91](#) function to retrieve the error information.

### Error Messages

The following message may be sent from this function
Error(s) occurred during running of ldap_next_reference API.

Related Information

- "ldap_first_entry()—Retrieve First LDAP Entry” on page 81 — Return first entry in a chain of search results.
- "ldap_next_entry()—Retrieve Next LDAP Entry” on page 131 — Return next entry in a chain of search results.
- "ldap_count_entries()—Retrieve Count of LDAP Entries” on page 36 — Return number of entry in a chain of search results.
- "ldap_get_entry_controls_np()—Extract Server Controls from an Entry” on page 90 — Extract server controls from an entry.
- "ldap_count_references()—Count continuation references in a chain of search results” on page 39 — Return the number of continuation reference in a chain of search results.
- "ldap_first_reference()—Retrieve First Continuation Reference in a Chain of Search Results” on page 84 — Return first continuation reference in a chain of search results.
- "ldap_parse_reference_np()—Extract Information from a Continuation Reference” on page 143 — Extract information from a continuation reference.

API introduced: V4R5

ldap_open()—Perform an LDAP Open Operation

Syntax

#include <ldap.h>

LDAP *ldap_open(
    char  *host,
    int    port)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The ldap_open() function opens a connection to an LDAP server and allocates an LDAP structure, which is used to identify the connection and to maintain per-connection information.

The ldap_open() function returns a pointer to an LDAP structure, which should be passed to subsequent calls to other LDAP functions such as "ldap_bind()—Perform an LDAP Bind Request” on page 23 and "ldap_search()—Perform an LDAP Search Operation” on page 164.

Although still supported, the use of ldap_open() is deprecated. The ldap_open() API allocates an LDAP structure and opens a connection to the LDAP server. Use of "ldap_init()—Perform an LDAP Initialization Operation” on page 105 instead of ldap_open() is recommended.

As a rule of thumb, the LDAP application is typically running as LDAP version 2 when it uses ldap_open() to create the LDAP connection. The LDAP application is typically running as LDAP version 3 when it uses ldap_init() to create the LDAP connection. However, it was possible with the LDAP V2 API to call ldap_init() so that there may be cases where this rule of thumb is not true.
Authorities and Locks
No i5/OS authority is required.

Parameters

**host**  (Input) Several methods are supported for specifying one or more target LDAP servers, including the following:

*Explicit Host List* Specifies the name of the host on which the LDAP server is running. The *host* parameter may contain a blank-separated list of hosts to try to connect to, and each host may optionally be of the form *host:port*. If present, the *port* overrides the *port* parameter.

The following are typical examples:

```c
ld=ldap_open("server1", ldap_port);
ld=ldap_open("server2:1200", ldap_port);
ld=ldap_open("server1:800 server2:2000 server3", ldap_port);
```

*Localhost* If the *host* parameter is **NULL**, the LDAP server will be assumed to be running on the local host.

*Default Hosts* If the *host* parameter is set to **LDAP_URL_PREFIX** ("ldap://") the LDAP library will attempt to locate one or more default LDAP servers, with non-SSL ports, using the SecureWay function. The port specified on the call is ignored, since `ldap_server_locate()` returns the port.

For example, the following two are equivalent:

```c
ld=ldap_open("ldap://", ldap_port);
ld=ldap_open(LDAP_URL_PREFIX, LDAP_PORT);
```

If more than one default server is located, the list is processed in sequence, until an active server is found.

The LDAP URL can include a Distinguished Name (DN), used as a filter for selecting candidate LDAP servers based on the server's suffix (or suffixes). If the most significant portion of the DN is an exact match with a server's suffix (after normalizing for case), the server is added to the list of candidate servers. For example, the following will only return default LDAP servers that have a suffix that supports the specified DN:

```c
ld=ldap_open("ldap:///cn=fred, dc=austin, dc=ibm, dc=com", LDAP_PORT);
```

In this case, a server that has a suffix of "dc=austin, dc=ibm, dc=com" would match. If more than one default server is located, the list is processed in sequence, until an active server is found.

If the LDAP URL contains a host name and optional port, the host is used to create the connection. No attempt is made to locate the default server(s), and the DN, if present, is ignored.

For example, the following two are equivalent:

```c
ld=ldap_open("ldap://myserver", LDAP_PORT);
ld=ldap_open("myserver", LDAP_PORT);
```
If the host parameter is prefixed with "/", the host parameter is assumed to be the name of a UNIX socket (that is, socket family is AF_UNIX) and port is ignored. Use of a UNIX socket requires the LDAP server to be running on the local host. In addition, the LDAP server must be listening on the specified UNIX socket. The i5/OS SecureWay Directory Services server listens on the /tmp/s.slapd local socket, in addition to any configured TCP/IP ports.

For example:
```
ld=ldap_open("/tmp/s.slapd", ldap_port);
```

If a specified host is prefixed with "privport://", then the LDAP library will use the rresvport() function to attempt to obtain one of the reserved ports (512 through 1023), instead of an "ephemeral" port. The search for a reserved port starts at 1023 and stops at 512. If a reserved port cannot be obtained, the function call will fail.

For example:
```
ld=ldap_open( "privport://server1,ldap_port" );
ld=ldap_open( "privport://server2:1200", ldap_port );
ld=ldap_open( "privport://server1:800 server2:2000 privport://server3", ldap_port );
```

**port**  
(Input) Specifies the TCP port number the server is listening on. If the default IANA-assigned port of 389 is desired, LDAP_PORT should be specified. To use the default SSL port 636 for SSL connections, use LDAPS_PORT.

**Return Value**

Pointer to an LDAP structure
  
  if the request was successful.

NULL  
if the request was not successful.

**Error Conditions**

The ldap_open() API will return NULL and set the ld_errno error code, if not successful. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error codes values.

**Error Messages**

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_open API.</td>
</tr>
</tbody>
</table>

**Related Information**

- “ldap_init()—Perform an LDAP Initialization Operation” on page 105 — Initializes a session with an LDAP server.
- “ldap_set_option() — Set LDAP Options” on page 189 — Set an option associated with an LDAP descriptor.
- “ldap_get_option()—Retrieve LDAP Options” on page 96 — Get an option associated with an LDAP descriptor.
ldap_parse_extended_result()—Parse extended result

Syntax
#include <ldap.h>

int ldap_parse_extended_result(
    LDAP *ld,
    LDAPMessage *res,
    char **resultoidp,
    struct berval **resultdatap,
    int freeit)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The ldap_parse_extended_result() function is used to parse the result of an extended operation initiated by ldap_extended_operation()—Perform extended operations.” on page 75.

Authorities and Locks
No i5/OS authority is required.

Parameters

Id (Input) Specifies the LDAP pointer returned by a previous call to ldap_init()—Perform an LDAP Initialization Operation” on page 105, ldap_ssl_init—Initialize an SSL Connection.” on page 204, or ldap_open()—Perform an LDAP Open Operation” on page 135.

res (Input) Specifies the result of an LDAP operation as returned by ldap_first_message()—Retrieve First LDAP Message” on page 82 or ldap_next_message()—Retrieve Next LDAP Message” on page 132 where the message type is LDAP_RES_EXTENDED.

resultoidp (Input) This result parameter specifies a pointer which is set to point to an allocated, dotted-OID text string returned from the server. This string should be disposed of using ldap_memfree()—Free Memory Allocated by LDAP API” on page 110 API. If no OID is returned, *resultoidp is set to NULL.

resultdatap (Input) This result parameter specifies a pointer to a berval structure pointer that is set to an allocated copy of the data returned by the server. This struct berval should be disposed of using ber_bvfree(). If no data is returned, *resultdatap is set to NULL.

freeit (Input) Specifies a boolean value that determines if the LDAP result (as specified by res) is to be freed. Any non-zero value will result in res being freed after the requested information is extracted. Alternatively, the ldap_msgfree()—Free LDAP Result Message” on page 125 API can be used to free the result at a later time.
Return Value

LDAP_SUCCESS
  if the request was successful.

another LDAP error code
  if the request was not successful.

Error Conditions

If `ldap_extended_result()` is not successful, `ld_errno` will be set to indicate the error. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values. “ldap_get_errno()—Retrieve Error Information” on page 91 function to retrieve the error information.

Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of <code>ldap_parse_extended_result</code> API.</td>
</tr>
</tbody>
</table>

Related Information

- “`ldap_extended_operation()`—Perform extended operation” on page 75 — Perform extended operation.
- “`ldap_extended_operation_s()`—Perform extended operations synchronously” on page 77 — Perform extended operations synchronously.
- “`ldap_first_message()`—Retrieve First LDAP Message” on page 82 — Retrieve first LDAP message.
- “`ldap_msgtype()`—Retrieve the Type of an LDAP Message” on page 128 — Retrieve the type of an LDAP message.
- “`ldap_next_message()`—Retrieve Next LDAP Message” on page 132 — Retrieve next LDAP message.
- “`ldap_result()`—Retrieve Result of an Asynchronous LDAP Operation” on page 156 — Retrieve Result of an Asynchronous LDAP Operation.

The `ldap_parse_extended_result()` API supports LDAP V3 server controls and client controls.

API introduced: V5R1

Syntax

```c
#include <ldap.h>

int ldap_parse_page_control(
  LDAP *ld,
  LDAPControl **serverControls,
  unsigned long *totalCount,
  struct berval **cookie)
```

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes
The `ldap_parse_page_control()` function is used to retrieve the values in a paged results control used when paging search results.

See [LDAP Paged Results](#) for usage information about the functions used to perform paging of entries returned from the server following an LDAP search operation.

Note: serverControls, and cookie must be freed by the caller.

### Authorities and Locks

No i5/OS authority is required. All authority checking is done by the LDAP server.

### Parameters

- **ld** (Input) Specifies the LDAP pointer returned by previous call to `ldap_init()`, `ldap_ssl_init()` or `ldap_open()`. Must not be NULL.

- **serverControls** (Input) Specifies a list of LDAP server controls. See [LDAP Controls](#) for more information about server controls. These controls are returned to the client when calling the `ldap_parse_result()` function on the set of results returned by the server. ServerControls can be freed immediately after the call to `ldap_parse_page_control()` with a call to `ldap_controls_free()`.

- **totalCount** (Output) Specifies the estimate of the total number of entries for this search. The value can be zero if the estimate cannot be provided.

- **cookie** (Output) Specifies an opaque structure returned by the server. The cookie should be passed into `ldap_parse_page_control()` as a NULL pointer. Upon return the value of the cookie indicates if there are more pages of results for this search stored on the server. An empty cookie upon return indicates that all results have been returned. Otherwise, this value must be passed into a subsequent call to `ldap_create_page_control` to get the next page of results. Thereafter, the cookie must be freed with a call to `ber_bvfree()`.

### Example

See [Code disclaimer information](#) for information pertaining to code examples.

See the example that uses `ldap_create_page_control()` and `ldap_parse_page_control()` to perform an LDAP sorted search in [“ldap_create_page_control()—Create a Paged Results Control used when paging search results” on page 42](#)—Create a Paged Results Control used when paging search results.

### Return Value

- **LDAP_SUCCESS** if the request was successful.

- another LDAP error code if the control could not be parsed.

### Related Information

- [“ldap_create_page_control()—Create a Paged Results Control used when paging search results” on page 42](#)—Create a paged results control.
• “ldap_create_sort_keylist()” — Create a Structure with Sort Key Values used when Sorting Search Results” on page 51 — Create a structure with sort key values.

• “ldap_free_sort_keylist()” — Free all Memory used by the Sort Key List” on page 85 — Free all memory used by the sort key list.

• “ldap_create_sort_control()” — Create a Sorted Results Control used when Sorting Search Results” on page 49 — Create a sorted results control.

• “ldap_parse_sort_control()” — Retrieve Values in a Sorted Results Control” on page 148 — Retrieve values in a sorted results control.

• “ldap_search()” — Perform an LDAP Search Operation” on page 164 — Asynchronously search the directory.

• “ldap_search_ext” — Asynchronously Search the Directory Using Controls” on page 166 — Asynchronously search the directory with controls.

• “ldap_search_ext_s” — Synchronously Search the Directory Using Controls” on page 168 — Synchronously search the directory with controls.

• “ldap_parse_result()” — Extract Information from Results” on page 144 — Extract information from results.

• “ldap_init()” — Perform an LDAP Initialization Operation” on page 105 — Initializes a session with an LDAP server.

• “ldap_ssl_init” — Initialize an SSL Connection.” on page 204 — Initializes an SSL connection

• “ldap_open()” — Perform an LDAP Open Operation” on page 135 — Open a connection to an LDAP server

• “ldap_control_free()” — Free storage allocated by the LDAP library” on page 33 — Free a single LDAPControl structure.

• “ldap_controls_free()” — Free storage allocated by the LDAP library” on page 32 — Free an array of LDAPControl structures.

API introduced: V5R3

#include <ldap.h>

int ldap_parse_pwdpolicy_response(
    LDAPControl **serverControls,
    int *controlerr,
    int *controlwarn,
    int *controlres)

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

The ldap_parse_pwdpolicy_response() function is used to obtain the LDAP password policy error or warning codes from the password policy response control associated with an LDAPMessage message. The two possible warnings are timebeforeexpiration and graceloginsremaining. For warnings, the function can also be used to obtain the LDAP password policy warning result code (the number of seconds value) associated with the warning code.
Authorities and Locks

No i5/OS authority is required. All authority checking is done by the LDAP server.

Parameters

serverControls

(Input) Specifies an array of LDAPCONTROL pointers returned by a previous call to
"ldap_parse_result()—Extract Information from Results" on page 144. See [LDAP Controls] for more information about server controls.

cntrollerr

(Input) Specifies a pointer to the result parameter that is filled in with the LDAP Password Policy error code. This can be used as input to "ldap_pwdpolicy_err2string()—Convert the Numeric LDAP Password Policy Error or Warning Code into a String" on page 150 to obtain a text description of the error. Possible controlerr values:

- LDAP_PASSWORD_EXPIRED 0x03
- LDAP_ACCOUNT_LOCKED 0x04
- LDAP_CHANGE_AFTER_RESET 0x05
- LDAP_TIME_BEFORE_EXPIRE 0x06
- LDAP_NEED_OLD_PASSWORD 0x07
- LDAP_INVALID_PASS_SYNTAX 0x08
- LDAP_PASSWORD_TOO_SHORT 0x09
- LDAP_PASSWORD_TOO_YOUNG 0x0A
- LDAP_PASSWORD_IN_HISTORY 0x0B

controlwarn

(Input) Specifies a pointer to the result parameter that is filled in with the LDAP Password Policy warning code. This can be used as input to "ldap_pwdpolicy_err2string()—Convert the Numeric LDAP Password Policy Error or Warning Code into a String" on page 150 to obtain a text description of the warning. Possible controlwarn values:

- LDAP_TIME_BEFORE_EXPIRE 0x01
- LDAP_GRACE_LOGINS 0x02

controlres

(Input) Specifies a pointer to the result parameter that is filled in with the LDAP Password Policy warning result value.

Error Conditions

The ldap_parse_pwdpolicy_response() function returns an LDAP error code if it encounters an error parsing the result. See ["LDAP Client API Error Conditions" on page 300] for possible LDAP error code values. The error code indicates if the operation completed successfully.

Related Information

- "ldap_pwdpolicy_err2string()—Convert the Numeric LDAP Password Policy Error or Warning Code into a String" on page 150 — Convert the numeric LDAP Password Policy error or warning code into a string.
- "ldap_parse_result()—Extract Information from Results" on page 144 — Extract information from results.
ldap_parse_reference_np()—Extract Information from a Continuation Reference

Syntax
#include <ldap.h>

int ldap_parse_reference_np(LDAP *ld, LDAPMessage *ref, char **referralsp, LDAPControl **serverctrlsp, int freeit)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The ldap_parse_reference_np() function is used to retrieve the list of alternate servers returned in an individual continuation reference in a chain of search results. This routine is also used to obtain an array of server controls returned in the continuation reference.

Note the suffix "_np" which shows the API is in a preliminary implementation, and is not documented in the Internet Draft. The internet community may standardize this API in the future.

Authorities and Locks
No i5/OS authority is required.

Parameters

Id (Input) Specifies the LDAP pointer returned by a previous call to "ldap_init()—Perform an LDAP Initialization Operation” on page 105, "ldap_ssl_init —Initialize an SSL Connection,” on page 204, or "ldap_open()—Perform an LDAP Open Operation” on page 135.

ref (Input) Specifies a pointer to a search continuation reference returned on a previous call to "ldap_first_reference()—Retrieve First Continuation Reference in a Chain of Search Results” on page 84 or "ldap_next_reference()—Retrieve the next Continuation Reference in a Chain of Search Results” on page 134.

referralsp (Output) Specifies a pointer to a result parameter that is filled in with the contents of the referrals field from the LDAPMessage ref, indicating zero or more alternate LDAP servers where the request should be retried. The referrals array should be freed by calling "ldap_value_free()—Free Memory Allocated by ldap_get_values()” on page 223. NULL may be supplied for this parameter to ignore the referrals field.

serverctrlsp (Input) Specifies a pointer to a result parameter that is filled in with an allocated array of controls copied out of the LDAPMessage ref. The control array should be freed by calling "ldap_controls_free()—Free storage allocated by the LDAP library” on page 32.

freeit (Input) Specifies a boolean value that determines if the LDAP result chain (as specified by ref) is to be freed. Any non-zero value will result in the LDAP result chain being freed after the requested information is extracted. Alternatively, the "ldap_msgfree()—Free LDAP Result Message” on page 125 API can be used to free the LDAP result chain at a later time.
Return Value
LDAP_SUCCESS
   if the call was successful.
another LDAP error code
   if the call was not successful.

Error Conditions
The ldap_parse_reference_np() function will return an LDAP error code if not successful. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values.

Error Messages
The following message may be sent from this function.

Message ID   | Error Message Text
--------------|--------------------------------------------------
CPF3CF2 E     | Error(s) occurred during running of ldap_parse_reference_np API.

Related Information
- “ldap_first_entry()—Retrieve First LDAP Entry” on page 81 — Return first entry in a chain of search results.
- “ldap_next_entry()—Retrieve Next LDAP Entry” on page 131 — Return next entry in a chain of search results.
- “ldap_count_entries()—Retrieve Count of LDAP Entries” on page 36 — Return number of entry in a chain of search results.
- “ldap_get_entry_controls_np()—Extract Server Controls from an Entry” on page 90 — Extract server controls from an entry.
- “ldap_count_references()—Count continuation references in a chain of search results” on page 39 — Return the number of continuation reference in a chain of search results.
- “ldap_first_reference()—Retrieve First Continuation Reference in a Chain of Search Results” on page 84 — Return first continuation reference in a chain of search results.
- “ldap_next_reference()—Retrieve the next Continuation Reference in a Chain of Search Results” on page 134 — Return next continuation reference in a chain of search results.

API introduced: V4R5

Syntax
#include <ldap.h>

int ldap_parse_result(
   LDAP *ld,
   LDAPMessage *res,
   int *errcodep,
   char **matcheddnp,
   char **errmsgp,
   char **referralsp,
   LDAPControl **servctrlsp,
   int freeit)

144 IBM Systems - iSeries: Lightweight Directory Access Protocol (LDAP) APIs
Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The ldap_parse_result() routine is used to:

- Obtain the LDAP error code field associated with an LDAPMessage res.
- Obtain the portion of the DN that the server recognizes for a failed operation.
- Obtain the text error message associated with the error code returned in an LDAPMessage res.
- Obtain the list of alternate servers from the referrals field.
- Obtain the array of controls that may be returned by the server.

**Authorities and Locks**

No i5/OS authority is required.

**Parameters**

\(ld\) (Input) Specifies the LDAP pointer returned by a previous call to "ldap_init()—Perform an LDAP Initialization Operation" on page 105, "ldap_ssl_init—Initialize an SSL Connection." on page 204, or "ldap_open()—Perform an LDAP Open Operation" on page 135.

\(res\) (Input) Specifies the result of an LDAP operation as returned by "ldap_result()—Retrieve Result of an Asynchronous LDAP Operation" on page 156 or one of the synchronous LDAP API operation calls.

\(errcodep\) (Output) Specifies a pointer to the result parameter that will be filled in with the LDAP error code field from the LDAPMessage res. The LDAPResult message is produced by the LDAP server, and indicates the outcome of the operation. NULL can be specified for errcodep if the error code is to be ignored.

\(matcheddn\) (Output) Specifies a pointer to a result parameter. When LDAP_NO_SUCH_OBJECT is returned as the LDAP error code, this result parameter will be filled in with a Distinguished Name indicating how much of the name in the request was recognized by the server. NULL can be specified for matcheddn if the matched DN is to be ignored. The matched DN string should be freed by calling "ldap_memfree()—Free Memory Allocated by LDAP API" on page 110.

\(errmsg\) (Output) Specifies a pointer to a result parameter that is filled in with the contents of the error message from the LDAPMessage res. The error message string should be freed by calling "ldap_memfree()—Free Memory Allocated by LDAP API" on page 110.

\(referrals\) (Output) Specifies a pointer to a result parameter that is filled in with the contents of the referrals field from the LDAPMessage res, indicating zero or more alternate LDAP servers where the request should be retried. The referrals array should be freed by calling "ldap_value_free()—Free Memory Allocated by ldap_get_values()" on page 223. NULL may be supplied for this parameter to ignore the referrals field.

\(serverctrlsp\) (Output) Specifies a pointer to a result parameter that is filled in with an allocated array of controls copied out of the LDAPMessage res. The control array should be freed by calling "ldap_controls_free()—Free storage allocated by the LDAP library" on page 32.

\(freecit\) (Input) Specifies a boolean value that determines if the LDAP result chain (as specified by res) is to be freed. Any non-zero value will result in the LDAP result chain being freed after the requested information is extracted. Alternatively, the "ldap_msgfree()—Free LDAP Result Message" on page 125 API can be used to free the LDAP result chain at a later time.
Return Value

LDAP_SUCCESS
    if the result was successfully located and parsed.

another LDAP error code
    if not successfully parsed.

Error Conditions

The ldap_parse_result() function will return an LDAP error code if not successful. See "LDAP Client API Error Conditions" on page 300 for possible LDAP error code values.

Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_parse_result API.</td>
</tr>
</tbody>
</table>

Related Information

- “ldap_first_message()—Retrieve First LDAP Message” on page 82 — Retrieve first LDAP message.
- “ldap_next_message()—Retrieve Next LDAP Message” on page 132 — Retrieve next LDAP message.
- “ldap_parse_extended_result()—Parse extended result” on page 138 — Parse extended result.
- “ldap_parse_sasl_bind_result()—Extract Server Credentials from SASL Bind Results” — Extract server credentials from SASL bind results.
- “ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156 — Retrieve result of an asynchronous LDAP operation.

API introduced: V4R5

Syntax

```
#include <ldap.h>

int ldap_parse_sasl_bind_result(
    LDAP *ld,
    LDAPMessage *res,
    struct berval **servercred,
    int freeit)
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threading: Yes

The ldap_parse_sasl_bind_result() function is used to obtain server credentials, as a result of an attempt to perform mutual authentication.
Authorities and Locks
No i5/OS authority is required.

Parameters

ld (Input) Specifies the LDAP pointer returned by a previous call to "ldap_init()—Perform an LDAP Initialization Operation” on page 105, "ldap_ssl_init—Initialize an SSL Connection,” on page 204, or “ldap_open()—Perform an LDAP Open Operation” on page 135.

res (Input) Specifies the result of an LDAP operation as returned by "ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156 or one of the synchronous LDAP API operation calls.

servercredp (Output) Specifies a pointer to a result parameter. For SASL bind results, this result parameter will be filled in with the credentials returned by the server for mutual authentication (if returned). The credentials, if returned, are returned in a struct berval. NULL may be supplied to ignore this field.

freeit (Input) Specifies a boolean value that determines if the LDAP result chain (as specified by ref) is to be freed. Any non-zero value will result in the LDAP result chain being freed after the requested information is extracted. Alternatively, the "ldap_msgfree()—Free LDAP Result Message” on page 125 API can be used to free the LDAP result chain at a later time.

Return Value

LDAP_SUCCESS
if the result was successfully located and parsed.

another LDAP error code
if not successfully parsed.

Error Conditions

The ldap_parse_sasl_bind_result() function will return an LDAP error code if not successful. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values.

Error Messages

The following message may be sent from this function.

Message ID   Error Message Text
CPF3CF2 E     Error(s) occurred during running of ldap_parse_sasl_bind_result API.

Related Information

• "ldap_first_message()—Retrieve First LDAP Message” on page 82 — Retrieve first LDAP message.
• "ldap_next_message()—Retrieve Next LDAP Message” on page 132 — Retrieve next LDAP message.
• "ldap_parse_result()—Extract Information from Results” on page 144 — Extract information from results.
• "ldap_sasl_bind()—Perform an LDAP SASL Bind Request” on page 159 — Perform an LDAP SASL bind request.
• "ldap_sasl_bind_s()—Perform an LDAP SASL Bind Request (Synchronous)” on page 161 — Perform an LDAP SASL bind request (synchronous).
## ldap_parse_sort_control()—Retrieve Values in a Sorted Results Control

### Syntax

```c
#include <ldap.h>

int ldap_parse_sort_control(
    LDAP *ld,
    LDAPControl **serverControls,
    unsigned long *sortRC,
    char **attribute)
```

### Library Name/Service Program

QSYS/QGLDCLNT

### Default Public Authority

*USE

### Threadsafe

Yes

The `ldap_parse_sort_control()` function is used to retrieve the values in a sorted results control used when sorting search results.

See [LDAP Sort](#) for usage information about the functions used to perform sorting of entries returned from the server following an LDAP search operation.

### Authorities and Locks

No i5/OS authority is required. All authority checking is done by the LDAP server.

### Parameters

- **ld** (Input) Specifies the LDAP pointer returned by previous call to `ldap_init()`, `ldap_ssl_init()` or `ldap_open()`. Must not be NULL.
- **serverControls** (Input) Specifies a list of LDAP server controls. See [LDAP Controls](#) for more information about server controls. These controls are returned to the client when calling the `ldap_parse_result()` function on the set of results returned by the server.
- **sortRC** (Output) Specifies the LDAP return code retrieved from the sort results control returned by the server.
- **attribute** (Output) Specifies the name of the attributed in error returned by the server.

### Return Value

- **LDAP_SUCCESS** if the request was successful.
- **another LDAP error code** if the control could not be parsed.

### Example

See [Code disclaimer information](#) for information pertaining to code examples.
See the example that uses `ldap_create_sort_keylist()`, `ldap_create_sort_control()`, `ldap_free_sort_keylist()`, and `ldap_parse_sort_control()` to perform an LDAP sorted search in “ldap_create_sort_keylist()—Create a Structure with Sort Key Values used when Sorting Search Results” on page 51. Create a Structure with Sort Key Values used when Sorting Search Results.

**Related Information**

- “ldap_create_sort_keylist()—Create a Structure with Sort Key Values used when Sorting Search Results” on page 51 — Create a structure with sort key values.
- “ldap_free_sort_keylist()—Free all Memory used by the Sort Key List” on page 85 — Free all memory used by the sort key list.
- “ldap_create_sort_control()—Create a Sorted Results Control used when Sorting Search Results” on page 49 — Create a sorted results control.
- “ldap_create_page_control()—Create a Paged Results Control used when paging search results” on page 42 — Create a paged results control.
- “ldap_parse_page_control()—Retrieve Values in a Paged Results Control” on page 139 — Retrieve values in a paged results control.
- “ldap_search()—Perform an LDAP Search Operation” on page 164 — Asynchronously search the directory.
- “ldap_parse_result()—Extract Information from Results” on page 144 — Extract information from results.

API introduced: V5R3

### ldap_perror()—Print LDAP Error Information

**Syntax**

```c
#include <ldap.h>

void ldap_perror(
    LDAP *ld,
    const char *s)
```

**Default Public Authority:** *USE  
**Library Name/Service Program:** QSYS/QGLDCLNT  
**Threadsafe:** Yes

The `ldap_perror()` function prints an indication of the error on standard error. The error string printed out will be in English only.

**Authorities and Locks**

No i5/OS authority is required.

**Parameters**

`ld`  
*(Input)* Specifies the LDAP pointer returned by a previous call to “ldap_init()—Perform an LDAP Initialization Operation” on page 105, “ldap_ssl_init —Initialize an SSL Connection.” on page 204, or “ldap_open()—Perform an LDAP Open Operation” on page 135.
s (Input) Specifies the message prefix, which is prepended to the string form of the error code stored in the LDAP structure. The string form of the error is the same string that would be returned by a call to "ldap_err2string()—Retrieve LDAP Error Message String" on page 68.

Return Value
None

Error Conditions
The ldap_perror() API does not return an error code.

Error Messages
The following message may be sent from this function.

Message ID             Error Message Text
CPF3CF2 E               Error(s) occurred during running of ldap_perror API.

Related Information
• "ldap_get_errno()—Retrieve Error Information" on page 91 — Retrieve error code set.
• "ldap_get_lderrno()—Retrieve Error Information" on page 93 — Retrieve error information.
• "ldap_err2string()—Retrieve LDAP Error Message String" on page 68 — Convert LDAP error indication to a string.
• "ldap_result2error()—Retrieve LDAP Error Information" on page 158 — Extract LDAP error indication from LDAP result.

API introduced: V4R3

Idap_pwdpolicy_err2string()—Convert the Numeric LDAP Password Policy Error or Warning Code into a String

Syntax
#include <ldap.h>

char *ldap_pwdpolicy_err2string(
    int err)

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

The ldap_pwdpolicy_err2string() function is used to convert a numeric LDAP password policy error or warning code, as returned by "ldap_parse_pwdpolicy_response()—Obtain the LDAP Password Policy Error or Warning Codes" on page 141, into a NULL-terminated character string that describes the error or warning. The character string is returned as static data and must not be freed by the caller.

The text description returned will be provided in English only.

Authorities and Locks
No i5/OS authority is required.
Parameters

`err` (Input) Specifies an integer value returned from `ldap_parse_pwdpolicy_response()`—Obtain the LDAP Password Policy Error or Warning Codes on page 141 containing the password policy warning or error code.

Return Value

LDAP error description String

a textual description of the LDAP error code.

Error Conditions

The `ldap_pwdpolicy_err2string()` API will return “Unknown Error” if the LDAP error code is unknown. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error codes and their description.

Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of <code>ldap_pwdpolicy_err2string</code> API</td>
</tr>
</tbody>
</table>

Related Information

- “ldap_parse_pwdpolicy_response()—Obtain the LDAP Password Policy Error or Warning Codes” on page 141 — Obtain the LDAP Password Policy error or warning codes.
- “ldap_err2string()—Retrieve LDAP Error Message String” on page 68 — Convert LDAP error indication to a string.
- “ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156 — Retrieve result of an asynchronous LDAP operation.
- “ldap_bind()—Perform an LDAP Bind Request” on page 23 — Asynchronously bind to the directory (deprecated).
- “ldap_sasl_bind()—Perform an LDAP SASL Bind Request” on page 159 — Asynchronously bind to the directory using SASL.
- `ldap_err2string`, `ldap_result` API introduced: V5R3

Idap_remove_control()—Remove a Control from the List of LDAP Server Controls

Syntax

```c
#include <ldap.h>

int ldap_remove_control(
    LDAPControl *delControl,
    LDAPControl ***ctrlList,
    int freeit)
```

Lightweight Directory Access Protocol (LDAP) APIs 151
The `ldap_remove_control()` function is used to remove a control from the list of LDAP server controls.

**Note:** If `freeit` is not 0, the control will be freed. If `freeit` is set to 0, the control will not be freed.

**Authorities and Locks**

No i5/OS authority is required. All authority checking is done by the LDAP server.

**Parameters**

- **`delControl`** (Input) Specifies the control to be deleted. See [LDAP Controls](#) for more information about server controls.

- **`ctrlList`** (Input) Specifies a list of LDAP server controls. See [LDAP Controls](#) for more information about server controls.

- **`freeit`** (Input) Specifies whether or not to free the control. If set to TRUE, the control will be freed. If set to FALSE, the control will not be freed.

**Return Value**

- **LDAP_SUCCESS** if the request was successful.

- **LDAP_NO_MEMORY** if the control could not be removed.

**Related Information**

- [“ldap_insert_control()—Insert a Control in the List of LDAP Server Controls” on page 108](#) — Insert a control in the list of LDAP server controls.

- [“ldap_remove_control()—Remove a Control from the List of LDAP Server Controls” on page 151](#) — Add a new LDAP server control.

- [“ldap_copy_controls()—Make a Copy of the List of LDAP Server Controls” on page 34](#) — Synchronously add an entry with controls.

API introduced: V5R3
The `ldap_rename()` routine initiates an asynchronous modify DN operation.

In LDAP version 2, the "ldap_modrdn()—Perform an LDAP Modify RDN Request" on page 120 API was used to change the name of an LDAP entry. It could only be used to change the least significant component of a name (the RDN or relative distinguished name). The LDAP version 3 protocol provides the Modify DN protocol operation that allows more general name change access. The `ldap_rename()` routine is used to change the name of an entry. The `ldap_modrdn()` routine is deprecated.

### Authorities and Locks

No i5/OS authority is required. All authority checking is done by the LDAP server.

### Parameters

- **id** (Input): Specifies the LDAP pointer returned by a previous call to "ldap_init()—Perform an LDAP Initialization Operation" on page 105, "ldap_ssl_init—Initialize an SSL Connection." on page 204, or "ldap_open()—Perform an LDAP Open Operation" on page 135.
- **dn** (Input): Specifies the DN of the entry whose DN is to be changed.
- **newrdn** (Input): Specifies the new RDN to be given to the entry.
- **newparent** (Input): Specifies the new parent, or superior entry. If this parameter is NULL, only the RDN of the entry is changed. The root DN may be specified by passing a zero length string, "". The newparent parameter should always be NULL when using version 2 of the LDAP protocol; otherwise the server’s behavior is undefined.
- **deleteoldrdn** (Input): Specifies a boolean value. When set to 1, the old RDN value is to be deleted from the entry. When set to 0, the old RDN value should be retained as a non-distinguished value. This parameter only has meaning if newrdn is different from the old RDN.
- **serverctrls** (Input): Specifies a list of LDAP server controls. This parameter may be set to null. See LDAP Controls for more information about server controls.
- **clientctrls** (Input): Specifies a list of LDAP client controls. This parameter may be set to null. See LDAP Controls for more information about client controls.
- **msgidp** (Output): This result parameter is set to the message id of the request if the `ldap_rename()` call succeeds.

### Return Value

**LDAP_SUCCESS**

if the request was successfully sent, `ldap_rename()` places the message id of the request in *msgidp. A subsequent call to "ldap_result()—Retrieve Result of an Asynchronous LDAP Operation" on page 156 can be used to obtain the result of the operation. Once the operation has completed, `ldap_result()` returns a result that contains the status of the operation (in the form of...
an error code. The error code indicates if the operation completed successfully. The "ldap_parse_result()—Extract Information from Results" on page 144 API is used to check the error code in the result.

another LDAP error code
in case of an error.

Error Conditions
If ldap_rename() is not successful, an error code will be returned. See "LDAP Client API Error Conditions" on page 300 for possible LDAP error code values.

Error Messages
The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_rename API.</td>
</tr>
</tbody>
</table>

Related Information
- "ldap_rename_s()—Synchronously Rename an Entry" — Synchronously rename an entry.
- "ldap_result()—Retrieve Result of an Asynchronous LDAP Operation" on page 156 — Retrieve result of an asynchronous LDAP operation.
- "ldap_modrdn()—Perform an LDAP Modify RDN Request" on page 120 — Asynchronously modify the RDN of an entry (deprecated).
- "ldap_modrdn_s()—Perform an LDAP Modify RDN Request (Synchronous)" on page 122 — Synchronously modify the RDN of an entry (deprecated).

The ldap_rename() API supports LDAP V3 server controls and client controls.

API introduced: V4R5

ldap_rename_s()—Synchronously Rename an Entry

Syntax
```
#include <ldap.h>

int ldap_rename_s(
  LDAP *ld,
  const char *dn,
  const char *newrdn,
  const char *newparent,
  int deletooldrdn,
  LDAPControl **serverctrls,
  LDAPControl **clientctrls)
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGGLDCLNT
Threadsafe: Yes

The ldap_rename_s() routine performs a synchronous modify DN operation.
In LDAP version 2, the `ldap_modrdn_s()`—Perform an LDAP Modify RDN Request (Synchronous)” on page 122 API was used to change the name of an LDAP entry synchronously. It could only be used to change the least significant component of a name (the RDN or relative distinguished name). The LDAP V3 protocol provides the Modify DN protocol operation that allows more general name change access. The `ldap_rename_s()` routine is used to change the name of an entry, and the use of the `ldap_modrdn_s()` routine is deprecated.

**Authorities and Locks**

No i5/OS authority is required. All authority checking is done by the LDAP server.

**Parameters**

- **ld** *(Input)* Specifies the LDAP pointer returned by a previous call to “ldap_init()—Perform an LDAP Initialization Operation” on page 105, “ldap_ssl_init—Initialize an SSL Connection.” on page 204, or “ldap_open()—Perform an LDAP Open Operation” on page 135.

- **dn** *(Input)* Specifies the DN of the entry whose DN is to be changed.

- **newrdn** *(Input)* Specifies the new RDN to be given to the entry.

- **newparent** *(Input)* Specifies the new parent, or superior entry. If this parameter is NULL, only the RDN of the entry is changed. The root DN may be specified by passing a zero length string, "". The `newparent` parameter should always be NULL when using version 2 of the LDAP protocol; otherwise the server’s behavior is undefined.

- **deleteoldrdn** *(Input)* Specifies a boolean value. When set to non-zero, the old RDN value is removed from the entry. When set to 0, the old RDN value will be retained as a non-distinguished value. This parameter only has meaning if `newrdn` is different from the old RDN.

- **serverctrls** *(Input)* Specifies a list of LDAP server controls. This parameter may be set to null. See LDAP Controls for more information about server controls.

- **clientctrls** *(Input)* Specifies a list of LDAP client controls. This parameter may be set to null. See LDAP Controls for more information about client controls.

**Return Value**

- **LDAP_SUCCESS** if the request was successfully.

- **another LDAP error code** in case of an error.

**Error Conditions**

If `ldap_rename_s()` is not successful, an error code will be returned. See LDAP Client API Error Conditions” on page 300 for possible LDAP error code values.

**Error Messages**

The following message may be sent from this function.

**Message ID** | **Error Message Text**
--- | ---
CPF3CF2 E | Error(s) occurred during running of ldap_rename_s API.
Related Information

- "ldap_rename()—Asynchronously Rename an Entry” on page 152 — Asynchronously rename an entry.
- "ldap_modrdn()—Perform an LDAP Modify RDN Request” on page 120 — Asynchronously modify the RDN of an entry (deprecated).
- "ldap_modrdn_s()—Perform an LDAP Modify RDN Request (Synchronous)” on page 122 — Synchronously modify the RDN of an entry (deprecated).

The ldap_rename_s() API supports LDAP V3 server controls and client controls.

API introduced: V4R5

The ldap_result() function is used to wait for and return the result of an operation previously initiated by one of the LDAP asynchronous operation functions (such as “ldap_search()—Perform an LDAP Search Operation” on page 164 and “ldap_modify()—Perform an LDAP Modify Entry Request” on page 111).

Authorities and Locks

No i5/OS authority is required.

Parameters

ld (Input) Specifies the LDAP pointer returned by a previous call to "ldap_init()—Perform an LDAP Initialization Operation” on page 105, "ldap_ssl_init —Initialize an SSL Connection.” on page 204, or “ldap_open()—Perform an LDAP Open Operation” on page 135.

msgid (Input) Specifies the message ID of the operation whose results are to be returned. The parameter can be set to LDAP_RES_ANY if any result is desired.

all (Input) This parameter only has meaning for search results. For search results, all is used to specify how many search result messages will be returned on this call to ldap_result(). Specify LDAP_MSG_ONE to retrieve one search result message (for example, a single entry). Specify LDAP_MSG_ALL to request that all results of a search be received. ldap_result() will wait until all results are received before returning the results in a single chain. Specify LDAP_MSG_RECEIVED to indicate that all results retrieved so far should be returned in the result chain.

timeout (Input) Specifies how long in seconds to wait for results (as identified by the supplied msgid) to
be returned from ldap_result. A NULL value causes ldap_result() to wait until results for the
operation identified by msgid are available. To poll, the timeout parameter should be non-NULL,
pointing to a zero-valued timeval structure.

result (Output) Contains the result of the asynchronous operation identified by msgid. This value should
be freed by "ldap_msgfree()—Free LDAP Result Message" on page 125 when the result is no
longer needed.

Return Value

-1 If ldap_result() was unsuccessful, sets the appropriate LDAP error, and
"ldap_get_errno()—Retrieve Error Information" on page 91 API can be used to obtain the error
code.
0 If ldap_result() times out or there is no message available.
If successful, it returns one of the following result types:

```
LDAP_RES_BIND 0x61L
LDAP_RES_SEARCH_ENTRY 0x64L
LDAP_RES_SEARCH_RESULT 0x65L
LDAP_RES_MODIFY 0x67L
LDAP_RES_ADD 0x69L
LDAP_RES_DELETE 0x6bL
LDAP_RES_MODRDN 0x6dL
LDAP_RES_COMPARE 0x6fL
LDAP_RES_SEARCH_REFERENCE 0x73L
LDAP_RES_EXTENDED 0x78L
LDAP_EXTENDED_RES_NAME 0x8aL
LDAP_EXTENDED_RES_VALUE 0x8bL
LDAP_RES_REFERRAL 0xa3L
LDAP_RES_ANY (-1L)
```

Error Conditions

If ldap_result() is not successful, ld_errno will be set to indicate the error. See "LDAP Client API Error
Conditions" on page 300 for possible values of ld_errno field. Use "ldap_get_errno()—Retrieve Error
Information" on page 91 function to retrieve the error information.

Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2E</td>
<td>Error(s) occurred during running of ldap_result API.</td>
</tr>
</tbody>
</table>

Related Information

- "ldap_count_messages()—Count messages in a result chain” on page 38 — Count messages in a result
chain.
- "ldap_first_entry()—Retrieve First LDAP Entry” on page 81 — Retrieve first LDAP entry.
- "ldap_first_message()—Retrieve First LDAP Message” on page 82 — Retrieve first LDAP message.
- "ldap_first_reference()—Retrieve First Continuation Reference in a Chain of Search Results” on page 84
— Retrieve first continuation reference in a chain of search results.
**ldap_result2error()—Retrieve LDAP Error Information**

**Syntax**

```c
#include <ldap.h>

int ldap_result2error(
    LDAP *ld,
    LDAPMessage *res,
    int freeit)
```

**Default Public Authority:** *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The `ldap_result2error()` API takes a result as produced by `ldap_result()`—Retrieve Result of an Asynchronous LDAP Operation” on page 156 or `ldap_search_s()—Perform an LDAP Search Operation (Synchronous)” on page 171, and returns the corresponding error code.

**Authorities and Locks**

No i5/OS authority is required.

**Parameters**

- **Id** (Input) Specifies the LDAP pointer returned by a previous call to `ldap_init()—Perform an LDAP Initialization Operation” on page 105, `ldap_ssl_init —Initialize an SSL Connection.” on page 204, or `ldap_open()—Perform an LDAP Open Operation” on page 135.

- **res** (Input/Output) Specifies the result, as produced by `ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156, to be converted to the error code with which it is associated.

- **freeit** (Input) Specifies whether or not the result, `res`, should be freed as a result of calling `ldap_result2error()`. If non-zero, the result, `res`, will be freed by the call. If zero, `res` will not be freed by the call.

**Return Value**

**LDAP error code**

The result of the `ldap` request in `res`. 

---

API introduced: V4R3

---
Error Conditions

The `ldap_result2error()` function will return an LDAP error code. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error codes.

Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2</td>
<td>Error(s) occurred during running of ldap_result2error API.</td>
</tr>
</tbody>
</table>

Related Information

- “ldap_err2string()—Retrieve LDAP Error Message String” on page 68 — Convert LDAP error indication to a string.
- “ldap_get_errno()—Retrieve Error Information” on page 91 — Retrieve error information.
- “ldap_err2string()—Print LDAP Error Information” on page 149 — Print an LDAP error indication to standard error.
- “ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156 — Retrieve result of an asynchronous LDAP operation.
- “ldap_search_s()—Perform an LDAP Search Operation (Synchronous)” on page 171 — Perform an LDAP search operation (synchronous).

API introduced: V4R3

Syntax

```c
#include <ldap.h>

int ldap_sasl_bind(
    LDAP *ld,
    const char *dn,
    const char *mechanism,
    const struct berval *cred,
    LDAPControl **serverctrls,
    LDAPControl **clientctrls,
    int *msgidp)
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The `ldap_sasl_bind()` function is used to authenticate a distinguished name (DN) to a directory server using Simple Authentication Security Layer (SASL).

After a connection is made to an LDAP V2 server an LDAP bind API must be called before any other LDAP APIs can be called for that connection. For LDAP V3 servers, binding is optional.

`ldap_sasl_bind()` is an asynchronous request. The result of the operation can be obtained by a subsequent call to “`ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156.”
The asynchronous version of this API only supports the LDAP_SASL_SIMPLE mechanism. You must use "ldap_sasl_bind_s()—Perform an LDAP SASL Bind Request (Synchronous)" on page 161 for other mechanisms.

**Authorities and Locks**
No i5/OS authority is required. All authority checking is done by the LDAP server.

**Parameters**

* ld (Input) Specifies the LDAP pointer returned by a previous call to "ldap_init()—Perform an LDAP Initialization Operation" on page 105, "ldap_ssl_init—Initialize an SSL Connection." on page 204, or "ldap_open()—Perform an LDAP Open Operation" on page 135.

* dn (Input) Specifies the Distinguished Name of the entry to bind as.

* mechanism (Input) This value can be set to NULL to perform a simple bind. Other mechanisms (EXTERNAL, CRAM-MD5, and GSSAPI) are implemented, but do not support the asynchronous SASL bind. You must use "ldap_sasl_bind_s()—Perform an LDAP SASL Bind Request (Synchronous)" on page 161 for other mechanisms.

* cred (Input) Specifies the credentials with which to authenticate. Arbitrary credentials can be passed using this parameter. In most cases, this is the user’s password.

* serverctrls (Input) Specifies a list of LDAP server controls. This parameter may be set to null. See LDAP Controls for more information about server controls.

* clientctrls (Input) Specifies a list of LDAP client controls. This parameter may be set to null. See LDAP Controls for more information about client controls.

* msgidp (Output) This result parameter is set to the message id of the request if the ldap_sasl_bind() call succeeds.

**Return Value**

Message ID of the operation initiated

-1 if the request was not successful.

**Error Conditions**

If ldap_sasl_bind() is not successful, an error code will be returned. See "LDAP Client API Error Conditions" on page 300 for possible LDAP error code values.

**Error Messages**

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_sasl_bind API.</td>
</tr>
</tbody>
</table>

**Related Information**

- "ldap_sasl_bind_s()—Perform an LDAP SASL Bind Request (Synchronous)" on page 161 — Synchronously bind to the directory using the Simple Authentication Security Layer (SASL).
• "ldap_simple_bind()—Perform a Simple LDAP Bind Request” on page 196 — Asynchronously bind to the directory using simple authentication.
• "ldap_simple_bind_s()—Perform a Simple LDAP Bind Request (Synchronous)” on page 198 — Synchronously bind to the directory using simple authentication.
• "ldap_unbind()—Perform an LDAP Unbind Request” on page 209 — Asynchronously unbind from the LDAP server and close the connection.
• "ldap_unbind_s()—Perform an LDAP Unbind Request (Synchronous)” on page 212 — Synchronously unbind from the LDAP server and close the connection.

API introduced: V4R5

### ldap_sasl_bind_s()—Perform an LDAP SASL Bind Request (Synchronous)

**Syntax**

```c
#include <ldap.h>

int ldap_sasl_bind_s(
    LDAP *ld,
    const char *dn,
    const char *mechanism,
    const struct berval *cred,
    LDAPControl **serverctrls,
    LDAPControl **clientctrls,
    struct berval **servercred)
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The `ldap_sasl_bind_s()` function can be used to do general authentication over LDAP through the use of the Simple Authentication Security Layer (SASL).

After a connection is made to an LDAP V2 server an LDAP bind API must be called before any other LDAP APIs can be called for that connection. For LDAP V3 servers, binding is optional.

`ldap_sasl_bind_s()` performs a synchronous request.

### Authorities and Locks

For the EXTERNAL mechanism, *R authority is needed to the selected Certificate Store and *X authority is needed to each directory of its path.

### Parameters

- **ld** *(Input)* The LDAP pointer returned by a previous call to "ldap_init()—Perform an LDAP Initialization Operation” on page 105, "ldap_ssl_init —Initialize an SSL Connection.” on page 204, or "ldap_open()—Perform an LDAP Open Operation” on page 135.
- **dn** *(Input)* The Distinguished Name of the entry to bind as, may be NULL.
- **mechanism** *(Input)* Although a variety of mechanisms have been IANA registered, the mechanisms supported by the library at this time are:
LDAP_MECHANISMEXTERNAL mechanism, represented by the string "EXTERNAL".

LDAP_MECHANISM_CRAMMD5 mechanism, represented by the string "CRAM-MD5".

LDAP_MECHANISM_GSSAPI mechanism, represented by the string "GSSAPI".

LDAP_SASL_SIMPLE, represented by the empty string.

LDAP_MECHANISM_DIGEST mechanism, represented by the string "DIGEST-MD5".

By setting mechanism to a NULL pointer, the SASL bind request will be interpreted as a request for simple authentication (equivalent to using "ldap_simple_bind_s()—Perform a Simple LDAP Bind Request (Synchronous)" on page 198).

The LDAP_MECHANISM_EXTERNAL mechanism indicates to the server that information external to SASL should be used to determine whether the client is authorized to authenticate. For this implementation, the system providing the external information must be SSL. The server will use the identity from the client’s X.509 certificate that was chosen using the "ldap_ssl_client_init —Initialize the SSL Library," on page 199 and "ldap_ssl_init —Initialize an SSL Connection," on page 204 or "ldap_app_ssl_client_init_np()—Initialize the LDAP Client for a Secure Connection using DCM" on page 15 API. The dn and cred parameters must be NULL.

The LDAP_MECHANISM_CRAMMD5 mechanism is used to authenticate with the server using a challenge/response protocol that protects the "clear-text" password over the wire. This mechanism is useful only when the LDAP server can retrieve the user’s password. The contents of the cred berval must be a UTF8 representation of the password. See "ldap_xlate_local_to_utf8()—Convert String From the Local Code Page to UTF-8 Encoding" on page 228 for converting local data to UTF8.

The LDAP_MECHANISM_GSSAPI mechanism is used to enable Kerberos authentication. The dn parameter must be NULL. If the cred parameter is NULL, then it is assumed that the user has already authenticated to a Kerberos security server and has obtained a Ticket Granting Ticket (TGT) using a program such as kinit. The GSSAPI credential handle used to initiate a security context on the LDAP client side is obtained from the current login context. The cred parameter can also point to a berval containing a GSSAPI credential handle that will be used to initiate a security context with the LDAP server. For example, a server application can call ldap_sasl_bind_s with a credential handle that the server received from a client as a delegated credential handle.

The LDAP_MECHANISM_DIGEST mechanism is used to authenticate with the server using a challenge/response protocol that protects the "clear-text" password over the wire. Unlike the CRAM-MD5 mechanism, the protocol also includes elements to prevent chosen plaintext attacks. When using this mechanism, the contents of the cred berval must be a UTF-8 representation of the password. See ldap_xlate_local_to_utf8() for converting local data to UTF-8. The LDAP_MECHANISM_DIGEST mechanism requires a username parameter. This is supplied to the API using a client control with an OID of IBM_CLIENT_MD5_USER_NAME_OID and value containing the user name string in UTF-8. The client can specify the DIGEST realm using a client control with an OID of IBM_CLIENT_MD5_REALM_NAME_OID and value containing the realm string in UTF-8. If no realm is supplied the client API will use the first one supplied by the server. The client can specify the DIGEST authorization ID by passing the value as the dn parameter. The format allowed for the dn parameter when using the LDAP_MECHANISM_DIGEST mechanism is either a DN, a DN prefixed with "u:", or a username prefixed with "u:".

cred (Input) Specifies the credentials with which to authenticate. Arbitrary credentials can be passed using this parameter. In most cases, this is the user’s password.

serverctrls (Input) Specifies a list of LDAP server controls. This parameter may be set to null. See LDAP Controls for more information about server controls.
clientctrls
(Input) Specifies a list of LDAP client controls. This parameter may be set to null. See LDAP Controls for more information about client controls.

servercredp
(Output) This result parameter will be set to the credentials returned by the server. If no credentials are returned, it will be set to NULL.

Return Value
LDAP_SUCCESS
if the request was successful.

another LDAP error code
if the request was not successful.

Error Conditions
If ldap_sasl_bind_s() is not successful, an error code is returned. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values.

Error Messages
The following message may be sent from this function.

Message ID   Error Message Text
CPF3CF2 E    Error(s) occurred during running of ldap_sasl_bind_s API.

Related Information
• “ldap_init()—Perform an LDAP Initialization Operation” on page 105 — Perform an LDAP initialization operation.
• “ldap_ssl_client_init —Initialize the SSL Library.” on page 199 — Initializes the SSL library.
• “ldap_ssl_init —Initialize an SSL Connection.” on page 204 — Initializes an SSL connection.
• “ldap_app_ssl_client_init_np()—Initialize the LDAP Client for a Secure Connection using DCM” on page 15 — Initialize the LDAP client for a secure connection using DCM.
• “ldap_app_ssl_init_np —Initialize an SSL Connection” on page 18 — Initializes an SSL connection.
• “ldap_get_bind_controls()—Get LDAP Bind Controls” on page 88 — Get LDAP bind controls.
• “ldap_sasl_bind()—Perform an LDAP SASL Bind Request” on page 159 — Asynchronously bind to the directory using the Simple Authentication Security Layer (SASL).
• “ldap_simple_bind()—Perform a Simple LDAP Bind Request” on page 196 — Asynchronously bind to the directory using simple authentication.
• “ldap_simple_bind_s()—Perform a Simple LDAP Bind Request (Synchronous)” on page 198 — Synchronously bind to the directory using simple authentication.
• “ldap_unbind()—Perform an LDAP Unbind Request” on page 209 — Asynchronously unbind from the LDAP server and close the connection.
• “ldap_unbind_s()—Perform an LDAP Unbind Request (Synchronous)” on page 212 — Synchronously unbind from the LDAP server and close the connection.

Examples
See Code disclaimer information for information pertaining to code examples.
This example shows how to use the bind controls to specify parameters to the DIGEST-MD5 mechanism and pass them to the ldap_sasl_bind_s API:

```
LDAPControl digest_realm, digest_user_name;
LDAPControl *ctrls[] = { &digest_realm, &digest_user_name, NULL };

digest_realm.ldctl_oid = IBM_CLIENT_MD5_REALM_NAME_OID;
digest_realm.ldctl_value = { strlen(realm), realm }; /* realm is UTF-8 */
digest_realm.ldctl_iscritical = 0;

Digest_user_name.ldctl_oid = IBM_CLIENT_MD5_USER_NAME_OID;
Digest_user_name.ldctl_value = { strlen(username), username }; /* username is UTF-8 */
Digest_user_name.ldctl_iscritical = 0;

rc = ldap_sasl_bind_s( ld,
    dn, /* this is the authzId */
    LDAP_MECHANISM_DIGEST,
    NULL, /* server controls */
    ctrls, /* client controls - realm & username */
    &servercred );
```

API introduced: V4R5

### ldap_search()—Perform an LDAP Search Operation

**Syntax**

```
#include <ldap.h>

int ldap_search(
    LDAP *ld,
    const char *base,
    int scope,
    const char *filter,
    char **attrs,
    int attrsonly)
```

**Default Public Authority:** *USE*

**Library Name/Service Program:** QSYS/QGLDCLNT

**Threading:** Yes

The `ldap_search()` function is used to perform an LDAP search operation.

`ldap_search()` is an asynchronous request. A subsequent call to ["ldap_result()—Retrieve Result of an Asynchronous LDAP Operation" on page 156](#) can be used to obtain the results from the search.

### Authorities and Locks

No i5/OS authority is required. All authority checking is done by the LDAP server.

### Parameters

- **ld** *(Input)* Specifies the LDAP pointer returned by a previous call to ["ldap_init()—Perform an LDAP Initialization Operation" on page 105](#), ["ldap_ssl_init —Initialize an SSL Connection.” on page 204](#), or ["ldap_open()—Perform an LDAP Open Operation” on page 135](#).

- **base** *(Input)* Specifies the DN of the entry at which to start the search.
scope  (Input) Specifies the scope of the search. It can be LDAP_SCOPE_BASE (to search the base object itself), or LDAP_SCOPE_ONELEVEL (to search the base object’s immediate children), or LDAP_SCOPE_SUBTREE (to search the base object and all its descendents).

filter  (Input) Specifies a string representation of the filter to apply in the search. Simple filters can be specified as attribute=attributevalue. More complex filters are specified using a prefix notation according to the following BNF:

<filter> ::= '(' <filtercomp> ')'  
<filtercomp> ::= <and> | <or> | <not> | <simple>  
<and> ::= '&' <filterlist>  
<or> ::= '|' <filterlist>  
<not> ::= '!' <filter>  
<filterlist> ::= <filter> | <filter> <filterlist>  
<simple> ::= <attributetype> <filtertype> <attributevalue>  
<filtertype> ::= '=' | '~=' | '<=' | '>' | '>='

The ‘~’=’ construct is used to specify approximate matching. The representation for <attributetype> and <attributevalue> are as described in RFC 2252, "Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions." In addition, <attributevalue> can be a single * to achieve an attribute existence test, or can contain text and *’s interspersed to achieve substring matching.

For example, the filter "(mail=*)" will find any entries that have a mail attribute. The filter "(mail="@student.of.life.edu")" will find any entries that have a mail attribute ending in the specified string.

More complex filters are created using the & and | operators. For example, the filter "(&(objectclass=person)(mail=*))" will find any entries that have an objectclass of person and a mail attribute. To put parentheses or asterisks in a filter, escape them with a backslash ‘\’ character. See RFC 2254, "A String Representation of LDAP Search Filters," for a more complete description of allowable filters.

attrs  (Input) Specifies a null-terminated array of character string attribute types to return from entries that match filter. If NULL is specified, all attributes will be returned.

attrsonly  (Input) Specifies attribute information. Attrsonly should be set to 1 to request attribute types only. Set to 0 to request both attributes types and attribute values.

Return Value

Message ID of the Operation Initiated
if the request was successful. A subsequent call to "ldap_result()—Retrieve Result of an Asynchronous LDAP Operation" on page 156, can be used to obtain the result.

-1 if the request was not successful.

Error Conditions
If ldap_search() is not successful, -1 will be returned setting the session error(ld_errno) parameters in the LDAP structure appropriately. See "LDAP Client API Error Conditions" on page 300 for possible values for the error codes. Use "ldap_get_errno()—Retrieve Error Information" on page 91 to obtain the error code ld_errno.

Error Messages
The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_search API.</td>
</tr>
</tbody>
</table>
Related Information

- “ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156 — Retrieve result of an asynchronous LDAP operation.
- “ldap_search_s()—Perform an LDAP Search Operation (Synchronous)” on page 171 — Synchronously search the directory.
- “ldap_search_ext — Asynchronously Search the Directory Using Controls” — Asynchronously search the directory with controls.
- “ldap_search_ext_s — Synchronously Search the Directory Using Controls” on page 168 — Synchronously search the directory with controls.
- “ldap_search_st()—Perform an LDAP Search Operation (Timed Synchronous)” on page 173 — Synchronously search the directory with timeout.

API introduced: V4R3

### ldap_search_ext — Asynchronously Search the Directory Using Controls

Syntax

```c
#include <ldap.h>

int ldap_search_ext(
    LDAP *ld,
    const char *base,
    int scope,
    const char *filter,
    char **attrs,
    int attrsonly,
    LDAPControl **serverctrls,
    LDAPControl **clientctrls,
    struct timeval *timeout,
    int sizelimit,
    int *msgidp)
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The `ldap_search_ext()` routine initiates an asynchronous search operation.

**Authorities and Locks**

No i5/OS authority is required. All authority checking is done by the LDAP server.

**Parameters**

- `ld` (Input) Specifies the LDAP pointer returned by a previous call to “ldap_init()—Perform an LDAP Initialization Operation” on page 105, “ldap_ssl_init — Initialize an SSL Connection.” on page 204, or “ldap_open()—Perform an LDAP Open Operation” on page 135.
- `base` (Input) Specifies the DN of the entry at which to start the search.
- `scope` (Input) Specifies the scope of the search. It can be `LDAP_SCOPE_BASE` (to search the base object
timeout, sizelimit, clientctrls, serverctrls, msgidp}

**filter**  
(Input) Specifies a string representation of the filter to apply in the search. Simple filters can be specified as attributetype=attributevalue. More complex filters are specified using a prefix notation according to the following BNF:

```
<filter> ::= '(' <filtercomp> ')'  
<filtercomp> ::= <and> | <or> | <not> | <simple>  
<and> ::= '&' <filterlist>  
<or> ::= '|' <filterlist>  
<not> ::= '!' <filter>  
<filterlist> ::= <filter> | <filter> <filterlist>  
<simple> ::= <attributetype> <filtertype> <attributevalue>  
<filtertype> ::= '=' | '~=' | '<=' | '>='
```

The ‘~=' construct is used to specify approximate matching. The representation for `<attributetype>` and `<attributevalue>` are as described in RFC 2252, "Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions." In addition, `<attributevalue>` can be a single * to achieve an attribute existence test, or can contain text and *’s interspersed to achieve substring matching.

For example, the filter "(mail=*)" will find any entries that have a mail attribute. The filter "(mail=@student.of.life.edu)" will find any entries that have a mail attribute ending in the specified string.

More complex filters are created using the & and | operators. For example, the filter "(&(objectclass=person)(mail=*))" will find any entries that have an objectclass of person and a mail attribute. To put parentheses or asterisks in a filter, escape them with a backslash \ character. See RFC 2254, "A String Representation of LDAP Search Filters," for a more complete description of allowable filters.

**attrs**  
(Input) Specifies a null-terminated array of character string attribute types to return from entries that match filter. If NULL is specified, all attributes will be returned.

**attrsonly**  
(Input) Specifies attribute information. Attrsonly should be set to 1 to request attribute types only. Set to 0 to request both attributes types and attribute values.

**serverctrls**  
(Input) Specifies a list of LDAP server controls. This parameter may be set to null. See [LDAP Controls](#) for more information about server controls.

**clientctrls**  
(Input) Specifies a list of LDAP client controls. This parameter may be set to null. See [LDAP Controls](#) for more information about client controls.

**sizelimit**  
(Input) Specifies the maximum number of entries to return. Note that the server may set a lower limit which is enforced at the server.

**timeout**  
(Input) The local search timeout value and the operation time limit that is sent to the server within the search request.

**msgid**  
(Output) This result parameter is set to the message id of the request if the ldap_search_ext() call succeeds.
**Return Value**

Message ID of the Operation Initiated
if the request was successful. A subsequent call to “ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156 can be used to obtain the result.

another LDAP error code
if the request was not successful.

**Error Conditions**

If ldap_search_ext() is not successful, an error code will be returned. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error codes.

**Error Messages**
The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_search_ext API.</td>
</tr>
</tbody>
</table>

**Related Information**

- “ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156 — Retrieve result of an asynchronous LDAP operation.
- “ldap_search_s()—Perform an LDAP Search Operation (Synchronous)” on page 171 — Synchronously search the directory.
- “ldap_search()—Perform an LDAP Search Operation” on page 164 — Asynchronously search the directory.
- “ldap_search_ext_s — Synchronously Search the Directory Using Controls” — Synchronously search the directory with controls.
- “ldap_search_st()—Perform an LDAP Search Operation (Timed Synchronous)” on page 173 — Synchronously search the directory with timeout.

The ldap_search_ext() API supports LDAP V3 server controls, client controls, and allow varying size and time limits to be easily specified for each search operation.

API introduced: V4R5

**ldap_search_ext_s — Synchronously Search the Directory Using Controls**

Syntax
```
#include <ldap.h>

int ldap_search_ext_s(
    LDAP *ld,
    const char *base,
    int scope,
    const char *filter,
    char **attrs,
    int attrsonly,
    LDAPControl **serverctrls,
```
The `ldap_search_ext_s()` routine initiates a synchronous search operation, allowing LDAP controls to be sent to the server and client.

**Authorities and Locks**

No i5/OS authority is required. All authority checking is done by the LDAP server.

**Parameters**

`ld` *(Input)* Specifies the LDAP pointer returned by a previous call to `ldap_init()`—Perform an LDAP Initialization Operation” on page 105, `ldap_ssl_init —Initialize an SSL Connection.” on page 204, or “ldap_open()—Perform an LDAP Open Operation” on page 135.*

`base` *(Input)* Specifies the DN of the entry at which to start the search.

`scope` *(Input)* Specifies the scope of the search. It can be **LDAP_SCOPE_BASE** (to search the object itself), or **LDAP_SCOPE_ONELEVEL** (to search the object’s immediate children), or **LDAP_SCOPE_SUBTREE** (to search the object and all its descendants).

`filter` *(Input)* Specifies a string representation of the filter to apply in the search. Simple filters can be specified as `attributetype=attributevalue`. More complex filters are specified using a prefix notation according to the following BNF:

```
<filter> ::= '(' <filtercomp> ')'
<filtercomp> ::= <and> | <or> | <not> | <simple>
<and>    ::= '&' <filterlist>
<or>     ::= '|' <filterlist>
<not>    ::= '!' <filter>
<filterlist> ::= <filter> | <filter> <filterlist>
<simple> ::= <attributetype> <attributevalue>
<filtertype> ::= '=' | '~=' | '<=' | '>'
```

The ‘~’ construct is used to specify approximate matching. The representation for `<attributetype>` and `<attributevalue>` are as described in RFC 2252, “Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions.” In addition, `<attributevalue>` can be a single * to achieve an attribute existence test, or can contain text and *’s interspersed to achieve substring matching.

For example, the filter 
"(mail=*)" will find any entries that have a mail attribute. The filter "(mail=*@student.of.life.edu)" will find any entries that have a mail attribute ending in the specified string.

More complex filters are created using the & and | operators. For example, the filter "(&(objectclass=person)(mail=*))" will find any entries that have an objectclass of person and a mail attribute. To put parentheses or asterisks in a filter, escape them with a backslash ‘\’ character. See RFC 2254, "A String Representation of LDAP Search Filters,” for a more complete description of allowable filters.

`attrs` *(Input)* Specifies a null-terminated array of character string attribute types to return from entries that match `filter`. If NULL is specified, all attributes will be returned.
attrsonly
(Input) Specifies attribute information. Attrsonly should be set to 1 to request attribute types only. Set to 0 to request both attributes types and attribute values.

serverctrls
(Input) Specifies a list of LDAP server controls. This parameter may be set to null. See [LDAP Controls](#) for more information about server controls.

clientctrls
(Input) Specifies a list of LDAP client controls. This parameter may be set to null. See [LDAP Controls](#) for more information about client controls.

sizelimit
(Input) Specifies the maximum number of entries to return. Note that the server may set a lower limit which is enforced at the server.

timeout
(Input) The local search timeout value and the operation time limit that is sent to the server within the search request.

res
(Output) Contains the result of the synchronous search operation. This result should be passed to the LDAP parsing routines (see ["ldap_first_entry()—Retrieve First LDAP Entry” on page 81](#), "ldap_next_entry()—Retrieve Next LDAP Entry” on page 131) and so on). The caller is responsible for freeing res with ["ldap_msgfree()—Free LDAP Result Message” on page 125](#).

Return Value

LDAP_SUCCESS
if the request was successful.

another LDAP error
if the request was not successful. The code can be interpreted by ["ldap_perror()—Print LDAP Error Information” on page 149](#) or ["ldap_err2string()—Retrieve LDAP Error Message String” on page 68](#).

Error Conditions
If [ldap_search_ext_s()](#) is not successful, an error code will be returned. See [“LDAP Client API Error Conditions” on page 300](#) for possible values for the error codes.

Error Messages
The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_search_ext_s API.</td>
</tr>
</tbody>
</table>

Related Information

- ["ldap_first_entry()—Retrieve First LDAP Entry” on page 81](#) — Retrieve first LDAP entry.
- ["ldap_first_reference()—Retrieve First Continuation Reference in a Chain of Search Results” on page 84](#) — Return first continuation reference in a chain of search results.
- ["ldap_count_entries()—Retrieve Count of LDAP Entries” on page 36](#) — Return number of entries in a chain of search results.
- ["ldap_msgfree()—Free LDAP Result Message” on page 125](#) — Free LDAP result message.
- ["ldap_search_s()—Perform an LDAP Search Operation (Synchronous)” on page 171](#) — Synchronously search the directory.
The `ldap_search_ext()` API supports LDAP V3 server controls, client controls, and allows varying size and time limits to be easily specified for each search operation.

API introduced: V4R5

Syntax

```c
#include <ldap.h>

int ldap_search_s(
   LDAP* ld,
    const char* base,
    int scope,
    const char* filter,
    char** attrs,
    int attrsonly,
    LDAPMessage** res)
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The `ldap_search_s()` function is used to perform a synchronous LDAP search operation.

**Authorities and Locks**

No i5/OS authority is required. All authority checking is done by the LDAP server.

**Parameters**

- `ld` (Input) Specifies the LDAP pointer returned by a previous call to `ldap_init()`—Perform an LDAP Initialization Operation on page 105, `ldap_ssl_init — Initialize an SSL Connection` on page 204, or `ldap_open() — Perform an LDAP Open Operation` on page 135.
- `base` (Input) Specifies the DN of the entry at which to start the search.
- `scope` (Input) Specifies the scope of the search. It can be LDAP_SCOPE_BASE (to search the object itself), or LDAP_SCOPE_ONELEVEL (to search the object’s immediate children), or LDAP_SCOPE_SUBTREE (to search the object and all its descendants).
- `filter` (Input) Specifies a string representation of the filter to apply in the search. Simple filters can be specified as attributetype=attributevalue. More complex filters are specified using a prefix notation according to the following BNF:

```plaintext
<filter> ::= '(' <filtercomp> ')'  
<filtercomp> ::= <and> | <or> | <not> | <simple> 
<and>   ::= '&<filterlist> 
<or>    ::= '|<filterlist>
```

The `ldap_search_ext_s()` function supports LDAP V3 server controls, client controls, and allows varying size and time limits to be easily specified for each search operation.
The ‘~’ construct is used to specify approximate matching. The representation for
<attributetype> and <attributevalue> are as described in RFC 2252, “Lightweight Directory Access
Protocol (v3): Attribute Syntax Definitions.” In addition, <attributevalue> can be a single * to
achieve an attribute existence test, or can contain text and *'s interspersed to achieve substring
matching.

For example, the filter "(mail=*)" will find any entries that have a mail attribute. The filter
"(mail=*@student.of.life.edu)" will find any entries that have a mail attribute ending in the
specified string.

More complex filters are created using the & and | operators. For example, the filter
"(&(objectclass=person)(mail=*))" will find any entries that have an objectclass of person and a
mail attribute. To put parentheses or asterisks in a filter, escape them with a backslash \ character. See RFC 2254, "A String Representation of LDAP Search Filters,” for a more complete
description of allowable filters.

.attrs (Input) Specifies a null-terminated array of character string attribute types to return from entries
that match filter. If NULL is specified, all attributes will be returned.

.attrsonly (Input) Specifies attribute information. Attrsonly should be set to 1 to request attribute types only.
Set to 0 to request both attributes types and attribute values.

.res (Output) Contains the result of the synchronous search operation. This result should be passed to
the LDAP parsing routines (see "ldap_first_entry()—Retrieve First LDAP Entry” on page 81,
"ldap_next_entry()—Retrieve Next LDAP Entry” on page 131, and so on). The caller is
responsible for freeing res with "ldap_msgfree()—Free LDAP Result Message” on page 125.

Return Value

LDAP_SUCCESS
if the request was successful.

another LDAP error
if the request was not successful. The code can be interpreted by "ldap_err2string()—Retrieve LDAP
Error Information” on page 149 or "ldap_err2string()—Retrieve LDAP Error Message String” on
page 68.

Error Conditions

If ldap_search_s() is not successful, an error code will be returned. See “LDAP Client API Error
Conditions” on page 300 for possible LDAP error codes values.

Error Messages

The following message may be sent from this function.

Message ID    Error Message Text
CPF3CF2 E   Error(s) occurred during running of ldap_search_s API.

Related Information

• "ldap_first_entry()—Retrieve First LDAP Entry” on page 81 — Retrieve first LDAP entry.
**ldap_search_st()**—Perform an LDAP Search Operation (Timed Synchronous)

**Syntax**

```c
#include <sys/time.h>
#include <ldap.h>

int ldap_search_st(
    LDAP *ld,
    const char *base,
    int scope,
    const char *filter,
    char **attrs,
    int attrsonly,
    struct timeval *timeout,
    LDAPMessage **res)
```

**Default Public Authority:** *USE
**Library Name/Service Program:** QSYS/QGLDCLNT
**Threadsafe:** Yes

The **ldap_search_st()** function is used to perform an LDAP search operation.

**Authorities and Locks**

No i5/OS authority is required. All authority checking is done by the LDAP server.

**Parameters**

- **ld** (Input) Specifies the LDAP pointer returned by a previous call to **ldap_init()**—Perform an LDAP Initialization Operation on page 105, **ldap_ssl_init**—Initialize an SSL Connection” on page 204, or **ldap_open()**—Perform an LDAP Open Operation” on page 135.
- **base** (Input) Specifies the DN of the entry at which to start the search.
- **scope** (Input) Specifies the scope of the search. It can be **LDAP_SCOPE_BASE** (to search the object itself), or **LDAP_SCOPE_ONELEVEL** (to search the object’s immediate children), or **LDAP_SCOPE_SUBTREE** (to search the object and all its descendants).
filter  (Input) Specifies a string representation of the filter to apply in the search. Simple filters can be specified as attributetype=attributevalue. More complex filters are specified using a prefix notation according to the following BNF:

\[
\begin{align*}
<\text{filter}> & ::= '(' <\text{filtercomp}> ')' \\
<\text{filtercomp}> & ::= <\text{and}> | <\text{or}> | <\text{not}> | <\text{simple}> \\
<\text{and}> & ::= '&' <\text{filterlist}> \\
<\text{or}> & ::= '|' <\text{filterlist}> \\
<\text{not}> & ::= '!'<\text{filter}> \\
<\text{filterlist}> & ::= <\text{filter}> | <\text{filter}> <\text{filterlist}> \\
<\text{simple}> & ::= <\text{attributetype}> <\text{filtertype}> <\text{attributevalue}> \\
<\text{filtertype}> & ::= '=' | '!=' | '<=' | '>=' | '<>' | '~=' \\
\end{align*}
\]

The ‘~’ construct is used to specify approximate matching. The representation for <attributetype> and <attributevalue> are as described in RFC 2252, “Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions.” In addition, <attributevalue> can be a single * to achieve an attribute existence test, or can contain text and ‘s interspersed to achieve substring matching.

For example, the filter "(mail=*\"@student.of.life.edu\")" will find any entries that have a mail attribute. The filter "(mail=@student.of.life.edu)" will find any entries that have a mail attribute ending in the specified string.

More complex filters are created using the & and | operators. For example, the filter "(&(objectclass=person)(mail=*))" will find any entries that have an objectclass of person and a mail attribute. To put parentheses or asterisks in a filter, escape them with a backslash ‘\’ character. See RFC 2254, “A String Representation of LDAP Search Filters,” for a more complete description of allowable filters.

attrs  (Input) Specifies a null-terminated array of character string attribute types to return from entries that match filter. If NULL is specified, all attributes will be returned.

attrsonly  (Input) Specifies attribute information. Attrsonly should be set to 1 to request attribute types only. Set to 0 to request both attributes types and attribute values.

timeout  (Input) The local search timeout value.

res  (Output) Contains the result of the synchronous search operation. This result should be passed to the LDAP parsing routines (see “ldap_first_entry()—Retrieve First LDAP Entry” on page 81, “ldap_next_entry()—Retrieve Next LDAP Entry” on page 131, and so on). The caller is responsible for freeing res with “ldap_msgfree()—Free LDAP Result Message” on page 125.

Return Value

LDAP_SUCCESS
if the request was successful.

another LDAP error code
if the request was not successful.

Error Conditions

If ldap_search_st() is not successful, an error code will be returned. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error codes.

Error Messages

The following message may be sent from this function.
CPF3CF2 E  Error(s) occurred during running of ldap_search_st API.

Related Information

- "ldap_first_entry()—Retrieve First LDAP Entry” on page 81 — Retrieve first LDAP entry.
- "ldap_first_reference()—Retrieve First Continuation Reference in a Chain of Search Results” on page 84 — Return first continuation reference in a chain of search results.
- "ldap_count_entries()—Retrieve Count of LDAP Entries” on page 36 — Return number of entries in a chain of search results.
- "ldap_msfree()—Free LDAP Result Message” on page 125 — Free LDAP result message.
- "ldap_search_s()—Perform an LDAP Search Operation (Synchronous)” on page 171 — Synchronously search the directory.
- "ldap_search()—Perform an LDAP Search Operation” on page 164 — Asynchronously search the directory.
- "ldap_search_ext —Asynchronously Search the Directory Using Controls” on page 166 — Asynchronously search the directory with controls.
- "ldap_search_ext_s — Synchronously Search the Directory Using Controls” on page 168 — Synchronously search the directory with controls.

API introduced: V4R3

Idap_server_conf_save() — Store Server Information into Local Configuration

Syntax

```
#include <ldap.h>

typedef struct LDAP_Server_Info {
    char *lsi_host; /* LDAP server's hostname */
    unsigned short lsi_port; /* LDAP port */
    char *lsi_suffix; /* Server's LDAP suffix */
    char *lsi_query_key; /* service_key[.edomain]*/
    char *lsi_dns_domain; /* Publishing DNS domain */
    int lsi_replica_type; /* master or replica */
#define LDAP_LSI_MASTER 1 /* LDAP Master */
#define LDAP_LSI_REPLICA 2 /* LDAP Replica */
    int lsi_sec_type; /* SSL or non-SSL */
#define LDAP_LSI_NOSSL 1 /* Non-SSL */
#define LDAP_LSI_SSL 2 /* Secure Server */
    unsigned short lsi_priority; /* Server priority */
    unsigned short lsi_weight; /* load balancing weight */
    char *lsi_vendor_info; /* vendor information */
    char *lsi_info; /* LDAP Info string */
    struct LDAP_Server_Info *prev; /* linked list previous ptr */
    struct LDAP_Server_Info *next; /* linked list next ptr */
} LDAPServerInfo;

int ldap_server_conf_save(
    char *filename,
    unsigned long ttl,
    LDAPServerInfo *server_info_listp );
```
The `ldap_server_conf_save()` API is used to store server information for the local configuration.

## Authorities and Locks

### Object Authorities

The caller must have Execute (*X) authority to each directory in the path name preceding the name of the configuration file. The caller must have Write (*W) authority to the configuration file.

## Parameters

- **server_info_list**
  - (input) A linked list of LDAPServerInfo structures. Each LDAPServerInfo structure defined in the list contains information on an LDAP server. This information will be stored in a file and can be retrieved using ["ldap_server_locate()— Locate Suitable LDAP Servers" on page 178]. The LDAPServerInfo structure contains the following fields:

  - **lsi_host**
    - Fully-qualified hostname of the target server (NULL-terminated string).
  - **lsi_port**
    - Integer representation of the LDAP server’s port.
  - **lsi_suffix**
    - String that specifies a supported suffix for the LDAP server (NULL-terminated string).
  - **lsi_query_key**
    - Specifies the The eNetwork domain to which the LDAP server belongs, prefixed by the service key. For example, if service key is ldap and eNetwork domain is sales, then lsi_query_key would be set to ldap.sales. If the server is not associated with an eNetwork domain (as published in DNS), then lsi_query_key consists solely of the service key value.
  - **lsi_dns_domain**
    - DNS domain in which the LDAP server was published. For example, the DNS search may have been for ldap.sales.tcp.austin.ibm.com, but the resulting server(s) has a fully-qualified DNS host name of ldap2.raleigh.ibm.com. In this example, lsi_host would be set to ldap2.raleigh.ibm.com whilst lsi_dns_domain would be set to austin.ibm.com. The actual domain in which the server was "published" may be of interest, particularly when multiple DNS domains are configured (or supplied as input).
  - **lsi_replica_type**
    - Specifies the type of server, LDAP_LSI_MASTER or LDAP_LSI_REPLICA. If set to zero, the type is unknown.
  - **lsi_sec_type**
    - Specifies the port’s security type, LDAP_LSI_NOSSL or LDAP_LSI_SSL. This value is derived from the "ldap" or "ldaps" prefix on the LDAP URL. If the LDAP URL is not defined, the security type is unknown and lsi_sec_type is set to zero.
  - **lsi_priority**
    - The priority value obtained from the SRV RR (or the "pseudo-SRV" TXT RR). Set to zero if unknown or not available.
  - **lsi_weight**
    - The weight value obtained from the SRV RR (or the "pseudo-SRV" TXT RR). Set to zero if unknown or not available.
  - **lsi_vendor_info**
    - NULL-terminated string obtained from the ldapvendor TXT RR (if defined). May be used to identify the LDAP server vendor/version information.
  - **lsi_info**
    - NULL-terminated information string obtained from the ldapinfo TXT RR (if defined). If not defined, lsi_info is set to NULL. This information string can be used by the LDAP or network administrator to publish additional information about the target LDAP server.

- **filename**
  - (input) The configuration filename. Specify NULL to get the default filename, /QIBM/UserData/OS400/DirSrv/ldap_server_info.conf.

- **ttl**
  - (input) Specifies the time-to-live (in minutes) for server information saved in the configuration file. Set ttl to zero if it is intended to be a permanent repository of information.
Return Value
LDAP_SUCCESS
    if the request was successful.

another LDAP error code
    if the request was not successful.

Error Conditions
If ldap_server_conf_save() is not successful, an LDAP error code will be returned. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values.

Error Messages
The following message may be sent from this function.

Message ID     Error Message Text
CPF3CF2 E       Error(s) occurred during running of ldap_server_conf_save API.

Related Information
• “ldap_init()—Perform an LDAP Initialization Operation” on page 105 — Perform an LDAP initialization operation.
• "ldap_server_locate()— Locate Suitable LDAP Servers" on page 178 — Locate suitable LDAP servers.
• "ldap_server_free_list()— Free the List of LDAP Servers" — Free the list of LDAP servers.

API introduced: V4R5

ldap_server_free_list()— Free the List of LDAP Servers

Syntax
#include <ldap.h>

int ldap_server_free_list(
    LDAPServerInfo *server_info_listp);

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The ldap_server_free_list() API is used to free the linked list of LDAPServerInfo structures (and all associated storage) as returned from the “ldap_server_locate()— Locate Suitable LDAP Servers” on page 178 API.

Authorities and Locks
No i5/OS authority is required.

Parameters

server_info_listp
    (Input) The address of a linked list of LDAPServerInfo structures to be freed.
Return Value
LDAP_SUCCESS
  if the request was successful.

another LDAP error code
  if the request was not successful.

Error Conditions
If ldap_server_free_list() is not successful, an error code will be returned. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values.

Error Messages
The following message may be sent from this function.

Message ID   Error Message Text
CPF3CF2 E   Error(s) occurred during running of ldap_server_free_list API.

Related Information
• “ldap_server_conf_save()— Store Server Information into Local Configuration” on page 175 — Store server information into local configuration.
• “ldap_server_locate()— Locate Suitable LDAP Servers”— Locate suitable LDAP servers.

API introduced: V4R5

ldap_server_locate()— Locate Suitable LDAP Servers

Syntax
#include <ldap.h>

typedef struct LDAP_Server_Request {
    int search_source;     /* Source for server info */
#define LDAP_LSI_CONF_DNS 0 /* Config first, then DNS (def)*/
#define LDAP_LSI_CONF_ONLY 1 /* Local Config file only */
#define LDAP_LSI_DNS_ONLY 2 /* DNS only */
    char *conf_filename;   /* pathname of config file */
    int reserved;         /* Reserved, set to zero */
    char *service_key;    /* Service string */
    char *enetwork_domain; /* eNetwork domain (eDomain) */
    char **name_servers;  /* Array of name server addrs */
    char **dns_domains;   /* Array of DNS domains */
    int connection_type;  /* Connection type */
#define LDAP_LSI_UDP_TCP 0 /* Use UDP, then TCP (default)*/
#define LDAP_LSI_UDP 1 /* Use UDP only */
#define LDAP_LSI_TCP 2 /* Use TCP only */
    int connection_timeout; /* connect timeout (seconds) */
    char *DN_filter;      /* DN suffix filter */
    unsigned char reserved2[64]; /* reserved fields, set to 0 */
} LDAPServerRequest;

typedef struct LDAP_Server_Info {
    char *lsi_host;        /* LDAP server's hostname */
    unsigned short lsi_port; /* LDAP port */
} LDAPServerInfo;
The `ldap_server_locate()` API is used to locate one or more suitable LDAP servers. In general, an application will use the `ldap_server_locate()` API as follows:

- Prior to connecting to an LDAP server in the enterprise, use `ldap_server_locate()` to obtain a list of one or more LDAP servers that have been published in DNS (or in the local configuration file). Typically an application can simply use the default request settings (by passing a NULL for the LDAPServerRequest parameter). By default, the API will look for server information in the local configuration file first (`/QIBM/UserData/OS400/DirSrv/ldap_server_info.conf`), then move on to DNS if the local configuration file doesn’t exist (or has expired).

- Once the application has obtained the list of servers, it should walk the list, using the first server that meets its needs. This will maximize the advantage that can be derived from using the priority and weighting scheme implemented by the administrator. The application may not want to use the first server in the list for several reasons:
  - The client needs to specifically connect using SSL (or non-SSL). In this case, the server needs to walk the list until it finds a server entry with the appropriate type of security type. Note that an LDAP server may be listening on both an SSL and non-SSL port. In this case, the server will have two entries in the server list.
  - The client specifically needs to connect to a Master (or Replica).
  - The client needs to connect to a server that supports a particular suffix. NOTE that the list of server's returned in the list can be filtered by specifying DN_filter, which filters out servers that do not have a suffix under which the DN resides.
  - There is some other characteristic associated with the desired server (perhaps defined in the ldapinfo string).

- Once the client has selected a server, it then issues the "ldap_init()—Perform an LDAP Initialization Operation" on page 105 or "ldap_ssl_init —Initialize an SSL Connection." on page 204 API. If the selected server is unavailable, the application is free to move down the list of servers until it either finds a suitable server it can connect to, or the list is exhausted.

**Authorities and Locks**

**Object Authorities**

The caller must have Execute (*X) authority to each directory in the path name preceding the
name of the configuration file (/QIBM/UserData/OS400/DirSrv). The caller must have Read (*R) authority to the configuration file (ldap_server_info.conf).

Parameters

server_request  
(Input) Specifies a pointer to an LDAPServerRequest structure. If the default behavior is desired for all possible input parameters, simply set server_request to NULL. Otherwise, supply the address of the LDAPServerRequest structure, which contains the following fields:

search_source  
Specifies where to find the server information.

The options are:
• First access the local LDAP DNS configuration file. If the file is not found, or the file does not contain information for a combination of the service_key, eDomain and any of the DNS domains (as specified by the application), then access DNS.
• Search the local LDAP DNS configuration file only.
• Search DNS only.

conf_filename  
Specifies an alternative configuration filename. Specify NULL to use the default filename and location.

service_key  
Specifies the search key (that is, the service name string) to be used when obtaining a list of SRV, "pseudo-SRV TXT" or CNAME alias records from DNS. If not specified, the default is "ldap". Administrators are encouraged to use the ldap default when setting up information in DNS servers, to maximize a client application's ability to find LDAP servers that have been published in DNS.

tenetwork_domain  
Indicates that LDAP servers belonging to the specified eNetwork domain are to be located. The criteria for searching DNS to locate the appropriate LDAP server(s) is constructed by concatenating the following information:
• search_key (defaults to ldap)
• enetwork_domain
• DNS domain
• "tcp"

For example, if:
• The default search_key of ldap is used
• The eNetwork domain is sales5
• The client’s default DNS domain is midwest.acme.com

Then the DNS "value" used to search DNS for the set of LDAP servers belonging to the sales5 domain is ldap.sales5.midwest.acme.com.tcp.

If enetwork_domain is set to zero, the following steps are taken to determine the enetwork_domain:
• The locally configured default, if set, will be used (as set with the "ldap_enetwork_domain_set()— Store the User’s Default eNetwork Domain Name” on page 66 API).
• If a locally configured default is not set, then a platform-specific value is used. On Windows NT, the user’s logon domain is used.
• If a platform-specific eNetwork domain is not defined, then the eNetwork domain component in the DNS "value" is omitted. In the above example, this would result in the following string being used: ldap.midwest.acme.com.tcp.

If enetwork_domain is set to a NULL string, then the eNetwork domain component in the DNS "value" is omitted. This might be useful for finding a default eNetwork domain (when a specific edomain name is not known).
**name_servers**

Specifies an array of one or more string representations of DNS name server IP address (in dotted decimal format; for example, "122.122.33.49"). If not specified, the locally configured DNS name server(s) will be used.

**dns_domains**

Specifies an array of one or more DNS domain names. If not specified, the local DNS domain configuration is used.

Note that domain names supplied here can take the following forms:

- austin.ibm.com (standard DNS format)
- cn=fred, ou=accounting, dc=austin, dc=ibm, dc=com

With respect to providing a domain name, these are equivalent. Both result in a domain name of "austin.ibm.com". This approach makes it easier for an application to locate LDAP servers to which it needs to bind (based on a user name space mapped into the DNS name space).

**DNS DOMAINS and CONFIGURATION FILE**

The local configuration file may contain server information for combinations of the following:

- Service key (typically set to ldap)
- eNetwork domain
- DNS domains

When the application sets search_source to LDAP_LSI_CONFIG_DNS, the ldap_server_locate() API will attempt to find server information in the configuration file for the designated service key, eNetwork domain and DNS domain(s).

If the configuration file does not contain information that matches this criteria, the locator API will search DNS, using the specified service key, eNetwork domain and DNS domain(s). For example:

- The application supplies the following three DNS domains:
  - austin.ibm.com
  - raleigh.ibm.com
  - miami.ibm.com
- plus, the application uses the default service key (that is, ldap and specifies sales for the eNetwork domain).
- The configuration file contains server information for austin.ibm.com and miami.ibm.com (with the default service key and eNetwork domain of sales).
- The search_source parameter is set to LDAP_LSI_CONFIG_DNS, which indicates that both the configuration file and DNS are to be used if necessary.
- The locator API will build a single ordered list of server entries, with the following:
  - Server entries for the austin.ibm.com DNS domain, as extracted from the configuration file.
  - Server entries for the raleigh.ibm.com DNS domain, as obtained from DNS over the network.
  - Server entries from the miami.ibm.com DNS domain, as extracted from the configuration file.

In other words, the resulting list of servers will contain all the austin.ibm.com servers first, followed by the raleigh.ibm.com servers, followed by the miami.ibm.com servers. Within each grouping of servers (by DNS domain), the entries are sorted by priority and weight.
connection_type

Specifies the type of connection to use when communicating with the DNS name server. The following options are supported:

- Use UDP first. It no response is received, or data truncation occurs, then use TCP.
- Only use UDP.
- Only use TCP.

If set to zero, the default is to use UDP first (then TCP).

UDP is the preferred connection type, and typically performs well. You might want to consider using TCP/IP:

- The amount of data being returned will not fit in the 512-byte UDP packet.
- The transmission and receipt of UDP packets turns out to be unreliable. This may depend on network characteristics.

connection_timeout

Specifies a timeout value when querying DNS (for both TCP and UDP). If LDAP_LSI_UDP_TCP is specified for connection_type and a response is not received in the specified time period for UDP, TCP will be attempted. A value of zero results in an infinite timeout. When the LDAPServerRequest parameter is set to NULL, the default is ten seconds. When passing the LDAPServerRequest parameter, this parameter should be set to a non-zero value if an indefinite timeout is not desired.

DN_filter

Specifies a Distinguished Name to be used as a filter, for selecting candidate LDAP servers based on the server’s suffix (or suffixes). If the most significant portion of the DN is an exact match with a server’s suffix (after normalizing for case), an LDAPServerInfo structure is returned for the server/suffix combination. If it doesn’t match, an LDAPServerInfo structure is not returned for the server/suffix combination.

reserved2

Represents a reserved area for future function, which should be initialized to zero.

server_info_listpp

(output) Specifies the address that will be set to point to a linked list of LDAPServerInfo structures. Each LDAPServerInfo structure defined in the list contains server information obtained from either:

- DNS
- Local configuration

Upon successful return from ldap_server_locate(), server_info_listpp points to a linked list of LDAPServerInfo structures. The LDAPServerInfo structure (as defined above), contains the following fields:

lsi_host

Fully-qualified hostname of the target server (NULL-terminated string).

lsi_port

Integer representation of the LDAP server’s port.

lsi_suffix

String that specifies a supported suffix for the LDAP server (NULL-terminated string).

lsi_query_key

Specifies the The eNetwork domain to which the LDAP server belongs, prefixed by the service key. For example, if service key is ldap and eNetwork domain is sales, then lsi_query_key would be set to ldap.sales. If the server is not associated with an eNetwork domain (as published in DNS), then lsi_query_key consists solely of the service key value.

lsi_dns_domain

DNS domain in which the LDAP server was published. For example, the DNS search may have been for ldap.sales.tcp.austin.ibm.com, but the resulting server(s) has a fully-qualified DNS host name of ldap2.raleigh.ibm.com. In this example, lsi_host would be set to ldap2.raleigh.ibm.com whilst lsi_dns_domain would be set to austin.ibm.com. The actual domain in which the server was "published" may be of interest, particularly when multiple DNS domains are configured (or supplied as input).

lsi_replica_type

Specifies the type of server, LDAP_LSI_MASTER or LDAP_LSI_REPLICA. If set to zero, the type is unknown.

lsi_sec_type

Specifies the port’s security type, LDAP_LSI_NOSSL or LDAP_LSI_SSL. This value is derived from the "ldap" or "ldaps" prefix on the LDAP URL. If the LDAP URL is not defined, the security type is unknown and lsi_sectype is set to zero.

lsi_priority

The priority value obtained from the SRV RR (or the "pseudo-SRV" TXT RR). Set to zero if unknown or not available.
The weight value obtained from the SRV RR (or the "pseudo-SRV" TXT RR). Set to zero if unknown or not available.

**lsi_vendor_info**
NULL-terminated string obtained from the ldapvendor TXT RR (if defined). May be used to identify the LDAP server vendor/version information.

**lsi_info**
NULL-terminated information string obtained from the ldapinfo TXT RR (if defined). If not defined, lsi_info is set to NULL. This information string can be used by the LDAP or network administrator to publish additional information about the target LDAP server.

**Return Value**

**LDAP_SUCCESS**
if the request was successful.

another LDAP error code
if the request was not successful.

**Error Conditions**

If `ldap_server_locate()` is not successful, an error code will be returned. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values.

**Error Messages**
The following message may be sent from this function.

**Message ID** | **Error Message Text**
--- | ---
CPF3CF2 E | Error(s) occurred during running of ldap_server_locate API.

**Related Information**

- “ldap_init()—Perform an LDAP Initialization Operation” on page 105 — Perform an LDAP initialization operation.
- “ldap_server_conf_save()—Store Server Information into Local Configuration” on page 175 — Store server information into local configuration.
- “ldap_server_free_list()—Free the List of LDAP Servers” on page 177 — Free the list of LDAP servers.

API introduced: V4R5

**ldap_set_iconv_local_charset()— Set the Active LDAP Character Set**

**Syntax**

```c
#include <ldap.h>

int ldap_set_iconv_local_charset ( char *charset )
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes
The `ldap_set_iconv_local_charset()` API checks if the character set is supported. If supported, the API calls  

`ldap_set_iconv_local_codepage() — Set the Active LDAP Code Page” on page 185 to set the global variable `ldap_global_codepage` to a corresponding codepage value.

A limited set of the IANA character sets will be supported. Character sets supported include:

<table>
<thead>
<tr>
<th>Character Set Name</th>
<th>Locale</th>
<th>Codepage</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO-8859-1</td>
<td>EN_US</td>
<td>819</td>
</tr>
<tr>
<td>ISO-8859-2</td>
<td>HU_HU</td>
<td>912</td>
</tr>
<tr>
<td>ISO-8859-5</td>
<td>RU_RU</td>
<td>915</td>
</tr>
<tr>
<td>ISO-8859-6</td>
<td>AR_AA</td>
<td>1089</td>
</tr>
<tr>
<td>ISO-8859-7</td>
<td>EL_GR</td>
<td>813</td>
</tr>
<tr>
<td>ISO-8859-8</td>
<td>IW_IL</td>
<td>916</td>
</tr>
<tr>
<td>ISO-8859-9</td>
<td>TR_TR</td>
<td>920</td>
</tr>
<tr>
<td>IBM437</td>
<td>n/a</td>
<td>437</td>
</tr>
<tr>
<td>IBM850</td>
<td>EN_US</td>
<td>850</td>
</tr>
<tr>
<td>IBM852</td>
<td>n/a</td>
<td>852</td>
</tr>
<tr>
<td>IBM857</td>
<td>n/a</td>
<td>857</td>
</tr>
<tr>
<td>IBM862</td>
<td>n/a</td>
<td>862</td>
</tr>
<tr>
<td>IBM864</td>
<td>n/a</td>
<td>864</td>
</tr>
<tr>
<td>IBM866</td>
<td>n/a</td>
<td>866</td>
</tr>
<tr>
<td>IBM869</td>
<td>n/a</td>
<td>869</td>
</tr>
<tr>
<td>TIS-620</td>
<td>TH_TH</td>
<td>874</td>
</tr>
<tr>
<td>EUC-JP</td>
<td>JA_JP</td>
<td>954</td>
</tr>
<tr>
<td>EUC-KR</td>
<td>KO_KR</td>
<td>970</td>
</tr>
<tr>
<td>EUC-CN</td>
<td>ZN_CN</td>
<td>1383</td>
</tr>
<tr>
<td>EUC-TW</td>
<td>ZH_TW</td>
<td>964</td>
</tr>
<tr>
<td>Shift-JIS</td>
<td>JA_JP</td>
<td>932</td>
</tr>
<tr>
<td>GBK</td>
<td>ZH_CN</td>
<td>1386</td>
</tr>
<tr>
<td>Big5</td>
<td>ZH_TW</td>
<td>950</td>
</tr>
</tbody>
</table>

### Authorities and Locks

No i5/OS authority is required.

### Parameters

`charset` (input) specifies character set value.

### Return Value

**LDAP_SUCCESS**  
if the request was successful.

**Other LDAP error code**  
if the request was not successful.

### Error Conditions

The `ldap_set_iconv_local_charset()` API returns an LDAP error code if not successful. See **“LDAP Client API Error Conditions”** on page 300 for possible values for LDAP error codes.

### Error Messages

The following message may be sent from this function.
Message ID: CPF3CF2E
Error Message Text: Error(s) occurred during running of ldap_set_iconv_local_charset API.

Related Information

- "ldap_xlate_local_to_utf8()— Convert String From the Local Code Page to UTF-8 Encoding” on page 228 — Convert String From the Local to UTF-8 Code Page.
- "ldap_xlate_utf8_to_local() — Convert String From the UTF-8 Encoding to Local Code Page” on page 231 — Convert String From UTF-8 to Local Code Page.
- "ldap_xlate_local_to_unicode()— Convert String From the Local Code Page to UCS-2 (or UNICODE) Encoding” on page 226 — Convert String From the Local to UCS-2 Code Page.
- "ldap_xlate_unicode_to_local() — Convert String From the UCS-2 (or UNICODE) Encoding to Local Code Page” on page 229 — Convert String From UCS-2 to Local Code Page.
- "ldap_get_iconv_local_codepage() — Get the Active LDAP Code Page” on page 92 — Get the Active LDAP Code Page.
- "ldap_set_iconv_local_codepage() — Set the Active LDAP Code Page” — Set the Active LDAP Code Page.
- "ldap_set_locale() — Change the Locale Used by LDAP” on page 188 — Change the Locale Used by LDAP.
- "ldap_get_locale()— Get Active LDAP Locale” on page 95 — Get the Locale Used by LDAP.

API introduced: V4R5

## ldap_set_iconv_local_codepage() — Set the Active LDAP Code Page

Syntax

```c
#include <ldap.h>

int ldap_set_iconv_local_codepage ( char *codepage )
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The `ldap_set_iconv_local_codepage()` API is used to set a global variable, `ldap_global_codepage`, to a value passed by `codepage` or to a value associated with a locale if `codepage` is NULL.

NOTE that the word local in the API refers to the value of the global variable `ldap_global_codepage` if it is set or a codepage value associated with the current locale.

### Authorities and Locks

No i5/OS authority is required.

### Parameters

`codepage`

(input) specifies local code page value.
Return Value

LDAP_SUCCESS
  if the request was successful.

Other LDAP error code
  if the request was not successful.

Error Conditions

The `ldap_set_iconv_local_codepage()` API will return an LDAP error code if not successful. See “LDAP Client API Error Conditions” on page 300 for possible values for LDAP error codes.

Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of <code>ldap_set_iconv_local_codepage</code> API</td>
</tr>
</tbody>
</table>

Related Information

- “[`ldap_xlate_local_to_utf8()`— Convert String From the Local Code Page to UTF-8 Encoding” on page 228 — Convert string from the local to UTF-8 code page.](#)
- “[`ldap_xlate_utf8_to_local()` — Convert String From the UTF-8 Encoding to Local Code Page” on page 231 — Convert string from UTF-8 to local code page.](#)
- “[`ldap_xlate_local_to_unicode()`— Convert String From the Local Code Page to UCS-2 (or UNICODE) Encoding” on page 226 — Convert string from the local to UCS-2 code page.](#)
- “[`ldap_xlate_unicode_to_local()` — Convert String From the UCS-2 (or UNICODE) Encoding to Local Code Page” on page 229 — Convert string from UCS-2 to local code page.](#)
- “[`ldap_get_iconv_local_codepage()`— Get the Active LDAP Code Page” on page 92 — Get the active LDAP code page.](#)
- “[`ldap_set_iconv_local_charset()`— Set the Active LDAP Character Set” on page 183 — Set the active LDAP character set.](#)
- “[`ldap_set_locale()` — Change the Locale Used by LDAP” on page 188 — Change the locale used by LDAP.](#)
- “[`ldap_get_locale()`— Get Active LDAP Locale” on page 95 — Get the locale used by LDAP.](#)

API introduced: V4R5

Syntax

```
#include <ldap.h>

int ldap_set_lderrno(
    LDAP *ld,
    int error,
    const char *dn,
    const char *errmsg )
```

ldap_set_lderrno() — Set Error Information

IBM Systems - iSeries: Lightweight Directory Access Protocol (LDAP) APIs
The `ldap_set_lderrno()` function sets an error code and other information about an error in the specified LDAP structure.

**Authorities and Locks**

No i5/OS authority is required.

**Parameters**

- `ld` (Input) Specifies the LDAP pointer returned by a previous call to `ldap_init()`—Perform an LDAP Initialization Operation” on page 105, `ldap_ssl_init —Initialize an SSL Connection,” on page 204, or “ldap_open()—Perform an LDAP Open Operation” on page 135.
- `error` (Input) The LDAP error code to be set in the `ld`.
- `dn` (Input) The distinguished name (DN) that identifies an existing entry. Normally, it is used to indicate how much of the name in the request is recognized by a server on an `LDAP_NO_SUCH_OBJECT` error. However, in this case since it is an input to this API it should be a DN consistent with the `error` and `errmsg` parameters input on this API.
- `errmsg` (Input) The text of the error message, as if returned from a server.

**Return Value**

LDAP error code

See “LDAP Client API Error Conditions” on page 300 for possible LDAP error codes values.

**Error Messages**

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of <code>ldap_set_lderrno</code> API</td>
</tr>
</tbody>
</table>

**Related Information**

- “ldap_err2string()—Retrieve LDAP Error Message String” on page 68 — Convert LDAP error indication to a string.
- “ldap_error()—Print LDAP Error Information” on page 149 — Print an LDAP error indication to standard error.
- “ldap_get_errno()—Retrieve Error Information” on page 91 — Obtain information from most recent error.
- “ldap_get_lderrno()—Retrieve Error Information” on page 93 — Retrieve Error Information
- “ldap_result2error()—Retrieve LDAP Error Information” on page 158 — Extract LDAP error indication from LDAP result.

API introduced: V5R1
Idap_set_locale() — Change the Locale Used by LDAP

Syntax
#include <ldap.h>

int ldap_set_locale(
    char *locale)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: No

The ldap_set_locale() API is used to change the locale used by LDAP for conversions between the local code page and UTF-8 or Unicode. Unless explicitly set with the ldap_set_locale() API, LDAP will use the application’s default locale. To force the LDAP library to use another locale, specify the appropriate locale string.

Note that the specified locale is applicable to all conversions by the LDAP library within the applications address space. The LDAP locale should be set or changed only when there is no other LDAP activity occurring within the application on other threads.

Authorities and Locks
*R authority is needed to the selected locale file and *X to the associated directories.

Parameters
locale (Input) The locale to be used by LDAP when using conversion apis to convert local text to/from UTF-8 or Unicode. If the locale is not explicitly set, the LDAP library will use the application’s default locale. To force the LDAP library to use another locale, specify the appropriate locale string.

You can set the value of locale to C, ””, LC_C or the IFS pathname of a *LOCALE object. A locale value of C indicates the default C environment. A locale value of ”” tells ldap_set_locale() to use the default locale for the implementation.

Examples:
rc = ldap_set_locale(LC_C);
rc = ldap_set_locale("/qsys.lib/en_us.locale");

Return Value
LDAP_SUCCESS
    if the request was successful.

another LDAP error code
    if the request was not successful.

Error Conditions
The ldap_set_locale() API will return LDAP error code if not successful. See “LDAP Client API Error Conditions” on page 300 for possible values for LDAP error codes.

Error Messages
The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_set_locale API.</td>
</tr>
</tbody>
</table>
Related Information

- "ldap_xlate_local_to_utf8()— Convert String From the Local Code Page to UTF-8 Encoding” on page 228 — Convert string from the local to UTF-8 code page.
- "ldap_xlate_utf8_to_local() — Convert String From the UTF-8 Encoding to Local Code Page” on page 231 — Convert string from UTF-8 to local code page.
- "ldap_xlate_local_to_unicode()— Convert String From the Local Code Page to UCS-2 (or UNICODE) Encoding” on page 226 — Convert string from the local to UCS-2 code page.
- "ldap_xlate_unicode_to_local() — Convert String From the UTF-8 Encoding to Local Code Page” on page 229 — Convert string from UCS-2 to local code page.
- "ldap_get_infoconv_local_codepage()— Get the Active LDAP Code Page” on page 92 — Get the active LDAP code page.
- "ldap_set_infoconv_local_codepage() — Set the Active LDAP Code Page” on page 185 — Set the active LDAP code page.
- "ldap_set_infoconv_local_charset()— Set the Active LDAP Character Set” on page 183 — Set the active LDAP character set.
- "ldap_get_locale()— Get Active LDAP Locale” on page 95 — Get the locale used by LDAP.

API introduced: V4R5

Top | “Lightweight Directory Access Protocol (LDAP) APIs,” on page 1 | APIs by category

**ldap_set_option() — Set LDAP Options**

Syntax

```c
#include <ldap.h>

int ldap_set_option(
    LDAP *ld,
    int optionToSet,
    const void *optionValue )
```

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

The ldap_set_option() function is used to set options for the specified LDAP connection.

**Authorities and Locks**

No i5/OS authority is required.

**Parameters**

*ld* (Input) An LDAP pointer returned by a previous call to "ldap_init()—Perform an LDAP Initialization Operation” on page 105, "ldap_ssl_init —Initialize an SSL Connection.” on page 204, or "ldap_open()—Perform an LDAP Open Operation” on page 135. If a NULL *ld* is passed in, the default option value is set. Later calls to "ldap_init()—Perform an LDAP Initialization Operation” on page 105, "ldap_ssl_init —Initialize an SSL Connection.” on page 204, or "ldap_open()—Perform an LDAP Open Operation” on page 135 will use the set value as the default for the option.
**optionToSet**  
(Input) The option value to be set. See below for the list of supported options.

**optionValue**  
(Input) The address of the value. For LDAP V3 client options, `optionValue` is the actual value to be set.

The following session settings can be set using the `ldap_set_option()` API:

- **LDAP_OPT_SIZELIMIT**: Maximum number of entries that can be returned on a search operation
- **LDAP_OPT_TIMELIMIT**: Maximum number of seconds to wait for search results
- **LDAP_OPT_REFHOPLIMIT**: Maximum number of referrals in a sequence that the client can follow
- **LDAP_OPT_DEREF**: Rules for following aliases at the server
- **LDAP_OPT_REFERRALS**: Whether referrals should be followed by the client
- **LDAP_OPT_DEBUG**: Client debug options
- **LDAP_OPT_SSL_CIPHER**: SSL ciphers to use
- **LDAP_OPT_SSL_TIMEOUT**: SSL timeout for refreshing session keys
- **LDAP_OPT_REBIND_FN**: Address of application’s setrebindproc procedure
- **LDAP_OPT_PROTOCOL_VERSION**: LDAP protocol version to use (V2 or V3)
- **LDAP_OPT_SERVER_CONTROLS**: Default server controls.
- **LDAP_OPT_CLIENT_CONTROLS**: Default client library controls
- **LDAP_OPT_UTF8_IO**: String Data type UTF-8 option

The value returned by `ldap_set_option()`—Retrieve LDAP Options” on page 96 when **LDAP_OPT_PROTOCOL_VERSION** is specified can be used to determine how parameters should be passed to the `ldap_set_option()` call. The easiest way to work with this compatibility feature is to guarantee that calls to `ldap_set_option()` are all performed while the **LDAP_OPT_PROTOCOL_VERSION** is set to the same value. If this cannot be guaranteed by the application, then follow the format of the example below when coding the call to `ldap_set_option()`:

```c
int sizeLimit=100;

int protocolVersion;

ldap_set_option( ld, LDAP_OPT_PROTOCOL_VERSION, &protocolVersion );

if ( protocolVersion == LDAP_VERSION2 ) {
    ldap_set_option( ld, LDAP_OPT_SIZELIMIT, (void *)&sizeLimit );
} else {
/* the protocol version is LDAP_VERSION3 */
    ldap_set_option( ld, LDAP_OPT_SIZELIMIT, &sizeLimit );
}
```

Additional details on specific options for `ldap_set_option()` are provided in the following sections.

**LDAP_OPT_SIZELIMIT**

Specifies the maximum number of entries that can be returned on a search operation. Note: the actual size limit for operations is also bounded by the maximum number of entries that the server is configured to return. Thus, the actual size limit will be the lesser of the value specified on this option and the value configured in the LDAP server. The default sizelimit is unlimited, specified with a value of zero (thus deferring to the sizelimit setting of the LDAP server).

**Examples:**

```c
sizevalue=50;
ldap_set_option( ld, LDAP_OPT_SIZELIMIT, &sizevalue);
ldap_get_option( ld, LDAP_OPT_SIZELIMIT, &sizevalue);
```
**LDAP_OPT_TIMELIMIT**

Specifies the number of seconds to wait for search results. Note: the actual time limit for operations is also bounded by the maximum time that the server is configured to allow. Thus, the actual time limit will be the lesser of the value specified on this option and the value configured in the LDAP server. The default is unlimited (specified with a value of zero).

**Examples:**

```c
    timeval=50;
    ldap_set_option( ld, LDAP_OPT_TIMELIMIT, &timevalue);
    ldap_get_option( ld, LDAP_OPT_TIMELIMIT, &timevalue);
```

**LDAP_OPT_REFHOPLIMIT**

Specifies the maximum number of hops that the client library will take when chasing referrals. The default is 5.

**Examples:**

```c
    hoplimit=7;
    ldap_set_option( ld, LDAP_OPT_REFHOPLIMIT, &hoplimit);
    ldap_get_option( ld, LDAP_OPT_REFHOPLIMIT, &hoplimit);
```

**LDAP_OPT_DEREF**

Specifies alternative rules for following aliases at the server. The default is LDAP_DEREF_NEVER.

Supported values:

0      LDAP_DEREF_NEVER
1      LDAP_DEREF_SEARCHING
2      LDAP_DEREF_FINDING
3      LDAP_DEREF_ALWAYS

**Examples:**

```c
    int deref = LDAP_DEREF_NEVER;
    ldap_set_option( ld, LDAP_OPT_DEREF, &deref);
    ldap_get_option( ld, LDAP_OPT_DEREF, &deref);
```

**LDAP_OPT_REFFERALS**

Specifies whether the LDAP library will automatically follow referrals returned by LDAP servers or not. It can be set to one of the constants LDAP_OPT_ON or LDAP_OPT_OFF. By default, the LDAP client will follow referrals.

**Examples:**

```c
    int value;
    ldap_set_option( ld, LDAP_OPT_REFFERALS, (void *)LDAP_OPT_ON);
    ldap_get_option( ld, LDAP_OPT_REFFERALS, &value);
```

**LDAP_OPT_DEBUG**

Specifies a bit-map that indicates the level of debug trace for the LDAP library.

Supported values:

```c
/* Debug levels */
LDAP_DEBUG_OFF    0x000
LDAP_DEBUG_TRACE  0x001
```
/* Debug levels */
LDAP_DEBUG_PACKETS 0x002
LDAP_DEBUG_ARGS 0x004
LDAP_DEBUG_CONNS 0x008
LDAP_DEBUG_BER 0x010
LDAP_DEBUG_FILTER 0x020
LDAP_DEBUG_CONFIG 0x040
LDAP_DEBUG_ACL 0x080
LDAP_DEBUG_STATS 0x100
LDAP_DEBUG_STATS2 0x200
LDAP_DEBUG_SHELL 0x400
LDAP_DEBUGPARSE 0x800
LDAP_DEBUG_ANY 0xffff

Examples:
int value;
int debugvalue = LDAP_DEBUG_TRACE | LDAP_DEBUG_PACKETS;
ladp_set_option( ld, LDAP_OPT_DEBUG, &debugvalue);
ladp_get_option( ld, LDAP_OPT_DEBUG, &value);

An alternative way to set the debug level is to set the LDAP_DEBUG environment variable in the job that the client application will run in. The environment variable is set to the same numerical value that the value variable would be set to if ldap_set_option() was used. An example of enabling client trace for an application using the LDAP_DEBUG environment variable:

ADDENVVAR ENVVAR(LDAP_DEBUG) VALUE(0X0003)

After the client application has run, use

DMPUSRTRC jobnumber-of-the-client-job

Then, to display the trace information interactively, use

DSPPFM QAP0ZDMP QP0Znnnnnn -- where nnnnnn is the job number.

**LDAP_OPT_SSL_CIPHER**

Specifies a set of one or more ciphers to be used when negotiating the cipher algorithm with the LDAP server. The first cipher in the list that is common with the list of ciphers supported by the server is chosen. For the export version of the library, the value used is "0306". For the domestic version of the library, the default value is "05040A090306". Note that the cipher string supported by the export version of the LDAP client library is fixed and cannot be modified.

Supported ciphers:

- **LDAP_SSL_RC4_MD5_EX** 03
- **LDAP_SSL_RC2_MD5_EX** 05 (Non-export only)
- **LDAP_SSL_RC4_SHA_US** 04 (Non-export only)
- **LDAP_SSL_RC4_MD5_US** 06
- **LDAP_SSL_DES_SHA_US** 09 (Non-export only)
- **LDAP_SSL_3DES_SHA_US** 0A (Non-export only)
- **LDAP_SSL_AES_SHA_US** 2F (Non-export only)

Examples:

```c
char *setcipher = "2F090A";
char *getcipher;
ladp_set_option( ld, LDAP_OPT_SSL_CIPHER, setcipher);
ladp_get_option( ld, LDAP_OPT_SSL_CIPHER, &getcipher );
```
**LDAP_OPT_SSL_TIMEOUT**

Specifies in seconds the SSL inactivity timer. After the specified seconds, in which no SSL activity has occurred, the SSL connection will be refreshed with new session keys. A smaller value may help increase security, but will have a small impact on performance. The default SSL timeout value is 43200 seconds.

Examples:

```c
value = 100;
ldap_set_option( ld, LDAP_OPT_SSL_TIMEOUT, &value );
ldap_get_option( ld, LDAP_OPT_SSL_TIMEOUT, &value );
```

**LDAP_OPT_REBIND_FN**

Specifies the address of a routine to be called by the LDAP library when the need arises to authenticate a connection with another LDAP server. This can occur, for example, when the LDAP library is chasing a referral. If a routine is not defined, referrals will always be chased using the anonymous identity. A default routine is not defined.

Examples:

```c
eextern LDAPRebindProc proc_address;
LDAPRebindProc value;
ldap_set_option( ld, LDAP_OPT_REBIND_FN, &proc_address);
ldap_get_option( ld, LDAP_OPT_REBIND_FN, &value );
```

**LDAP_OPT_PROTOCOL_VERSION**

Specifies the LDAP protocol to be used by the LDAP client library when connecting to an LDAP server. Also used to determine which LDAP protocol is being used for the connection. For an application that uses the following code:

```c
/*"ldap_init()—Perform an LDAP Initialization Operation" on page 105*/
ldap_set_option( ld, LDAP_OPT_PROTOCOL_VERSION, &version2 );
/*"ldap_open()—Perform an LDAP Open Operation" on page 135*/
ldap_set_option( ld, LDAP_OPT_PROTOCOL_VERSION, &version3 );
```

The default value of this option will be LDAP_VERSION3 for communicating with the LDAP server. For an application that uses the following code:

```c
ldap_set_option( ld, LDAP_OPT_PROTOCOL_VERSION, &version2 );
```

The default value of this option will be LDAP_VERSION2 if the application uses the deprecated "ldap_open()—Perform an LDAP Open Operation" on page 135 API. In either case, the `LDAP_OPT_PROTOCOL_VERSION` option can be used with `ldap_set_option()` to change the default. The LDAP protocol version should be reset prior to issuing the bind (or any operation that causes an implicit bind).

Examples:

```c
version2 = LDAP_VERSION2;
version3 = LDAP_VERSION3;
/* Example for Version 3 application setting version to version 2 */
ldap_set_option( ld, LDAP_OPT_PROTOCOL_VERSION, &version2 );
/* Example of Version 2 application setting version to version 3 */
ldap_set_option( ld, LDAP_OPT_PROTOCOL_VERSION, &version3 );
ldap_get_option( ld, LDAP_OPT_PROTOCOL_VERSION, &value );
```

**LDAP_OPT_SERVER_CONTROLS**

Specifies a default list of server controls to be sent with each request. The default list can be overridden by specifying a server control, or list of server controls, on specific APIs. By default, no server controls will be sent.

Example:

```c
ldap_set_option( ld, LDAP_OPT_SERVER_CONTROLS, &ctrlp );
```

**LDAP_OPT_CLIENT_CONTROLS**

Specifies a default list of client controls to be processed by the client library with each request. Since client controls are not defined for this version of the library, the `ldap_set_option()` API can be used to define a set of default, non-critical client controls. If one or more client controls in the set is critical, the entire list is rejected with a return code of `LDAP_UNAVAILABLE_CRITICAL_EXTENSION`.
**LDAP_OPT_UTF8_IO**

Specifies whether the LDAP library will automatically convert string data to and from the local code page. It can be set to one of the constants `LDAP_UTF8_XLATE_ON` or `LDAP_UTF8_XLATE_OFF`. By default, the LDAP library will convert string data.

When conversion is disabled, the LDAP library assumes that data received from the application by LDAP APIs is already represented in UTF-8. Similarly, the LDAP library assumes that the application is prepared to receive string data from the LDAP library represented in UTF-8 (or as binary).

When `LDAP_UTF8_XLATE_ON` is set (the default), the LDAP library assumes that string data received from the application by LDAP APIs is in the default (or explicitly designated) code page. Similarly, all string data returned from the LDAP library (back to the application) is converted to the designated local code page.

*It is important to note* that only string data supplied on connection-based APIs will be translated (that is, only those APIs that include an `ld` will be subject to translation). For example, string values passed in to "ldap_search()—Perform an LDAP Search Operation” on page 164 will be converted, but string values passed in to "ldap_init()—Perform an LDAP Initialization Operation” on page 105 will not.

*It is also important to note* that translation of strings from a UTF-8 encoding to local code page may result in loss of data when one or more characters in the UTF-8 encoding cannot be represented in the local code page. When this occurs, a substitution character replaces any UTF-8 characters that cannot be converted to the local code page.

For more information on explicitly setting the locale for conversions, see "ldap_set_locale() — Change the Locale Used by LDAP” on page 188.

**Examples:**

```c
int value;
ldap_set_option( ld, LDAP_OPT_UTF8_IO, (void *)LDAP UTF8_XLATE_ON);
ldap_get_option( ld, LDAP_OPT_UTF8_IO, &value);
```

**Return Value**

*LDAP_SUCCESS*

if the request was successful.

*another LDAP error code*

if the request was not successful.

**Error Conditions**

The `ldap_set_option()` function will return an LDAP error code if not successful. See "LDAP Client API Error Conditions” on page 300 for possible LDAP error codes values.

**Error Messages**

The following message may be sent from this function.

**Message ID** CPF3CF2 E **Error Message Text**

Error(s) occurred during running of ldap_set_option API.

**Related Information**

- "ldap_get_option()—Retrieve LDAP Options” on page 96 — Retrieve an option associated with an LDAP descriptor.
Idap_set_rebind_proc()—Set Rebind Procedure

Syntax

```c
#include <ldap.h>

void ldap_set_rebind_proc(
  LDAP *ld,
  LDAPRebindProc rebindproc )
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The `ldap_set_rebind_proc()` function is used to set the entry-point of a routine that will be called back to obtain bind credentials for use when a new server is contacted during the following of an LDAP referral. Note that this function is only useful when the `LDAP_OPT_REFERRALS` option is set (this is the default). If `ldap_set_rebind_proc()` is never called, or if it is called with a NULL `rebindproc` parameter, an unauthenticated simple LDAP bind will always be done when chasing referrals.

`rebindproc` should be a function that is declared like this:

```c
int rebindproc( LDAP *ld, char **whop, char **credp,
                int *methodp, int freeit );
```

The LDAP library will first call the rebindproc to obtain the referral bind credentials, and the `freeit` parameter will be zero. The function must set `whop`, `credp`, and `methodp` as appropriate. If the rebindproc returns `LDAP_SUCCESS`, referral processing continues, and the rebindproc will be called a second time with `freeit` non-zero to give your application a chance to free any memory allocated in the previous call.

If anything but `LDAP_SUCCESS` is returned by the first call to the rebindproc, referral processing is stopped and that error code is returned for the original LDAP operation.

Authorities and Locks
No i5/OS authority is required.

Parameters

`ld`  
(Input) Specifies the LDAP pointer returned by a previous call to "ldap_init()—Perform an LDAP Initialization Operation” on page 105, "ldap_ssl_init—Initialize an SSL Connection.” on page 204, or "ldap_open()—Perform an LDAP Open Operation” on page 135.

`rebindproc`  
(Input) Specifies the entry-point of a routine that will be called to obtain bind credentials used when a new server is contacted during the following of an LDAP referral.
Return Value
None

Error Conditions
The ldap_set_rebind_proc() API does not return an error code.

Error Messages
The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_set_rebind_proc API.</td>
</tr>
</tbody>
</table>

Related Information
- "ldap_bind()—Perform an LDAP Bind Request" on page 23 — Asynchronously bind to the directory (deprecated).
- "ldap_bind_s()—Perform an LDAP Bind Request (Synchronous)" on page 25 — Synchronously bind to the directory (deprecated).
- "ldap_sasl_bind()—Perform an LDAP SASL Bind Request" on page 159 — Asynchronously bind to the directory using SASL.
- "ldap_sasl_bind_s()—Perform an LDAP SASL Bind Request (Synchronous)" on page 161 — Synchronously bind to the directory using SASL.
- "ldap_simple_bind()—Perform a Simple LDAP Bind Request" — Asynchronously bind to the directory using simple authentication.
- "ldap_simple_bind_s()—Perform a Simple LDAP Bind Request (Synchronous)" on page 198 — Synchronously bind to the directory using simple authentication.
- "ldap_unbind()—Perform an LDAP Unbind Request" on page 209 — Asynchronously unbind from the LDAP server and close the connection.
- "ldap_unbind_s()—Perform an LDAP Unbind Request (Synchronous)" on page 212 — Synchronously unbind from the LDAP server and close the connection.

API introduced: V4R3

ldap_simple_bind()—Perform a Simple LDAP Bind Request

Syntax
#include <ldap.h>

int ldap_simple_bind(
    LDAP *ld,
    const char *dn,
    const char *passwd
)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The ldap_simple_bind() function is used to authenticate a distinguished name (DN) to a directory server.
For LDAP V2 servers, after a connection is made to an LDAP server by using the
"ldap_open()—Perform an LDAP Open Operation” on page 135, "ldap_init()—Perform an LDAP Initialization Operation” on
page 105, or "ldap_ssl_init —Initialize an SSL Connection.” on page 204 APIs, an LDAP bind API must be
called before any other LDAP APIs can be called for that connection. For LDAP V3 servers, the bind is
optional.

ldap_simple_bind() is an asynchronous request. The result of the operation can be obtained by a
subsequent call to "ldap_result()—Retrieve Result of an Asynchronous LDAP Operation” on page 156.

Authorities and Locks
No i5/OS authority is required. All authority checking is done by the LDAP server.

Parameters

\( ld \) (Input) Specifies the LDAP pointer returned by a previous call to "ldap_init()—Perform an LDAP
Initialization Operation” on page 105, "ldap_ssl_init —Initialize an SSL Connection.” on page 204,
or "ldap_open()—Perform an LDAP Open Operation” on page 135.

\( dn \) (Input) Specifies the Distinguished Name of the entry to bind as.

\( passwd \) (Input) Specifies the password used in association with DN of the entry in which to bind.

Return Value

Message ID of the Operation Initiated

if the request was successful. A subsequent call to "ldap_result()—Retrieve Result of an
Asynchronous LDAP Operation” on page 156, can be used to obtain the result.

-1 if the request was not successful, setting the session error parameters in the LDAP structure
appropriately, which can be obtained by using "ldap_get_lderrno()—Retrieve Error Information”
on page 93.

Error Conditions

If ldap_simple_bind() is not successful, -1 will be returned setting the session error (ld_errno) parameters
in the LDAP structure appropriately. See ”LDAP Client API Error Conditions” on page 300 for possible
LDAP error code values. Use "ldap_get_lderrno()—Retrieve Error Information” on page 93 to obtain the
error code ld_errno.

Error Messages

The following message may be sent from this function.

Message ID Error Message Text
CPF3CF2 E Error(s) occurred during running of ldap_simple_bind API.

Related Information

- "ldap_bind()—Perform an LDAP Bind Request” on page 23 — Asynchronously bind to the directory
  (deprecated).
- "ldap_bind_s()—Perform an LDAP Bind Request (Synchronous)” on page 25 — Synchronously bind to
  the directory (deprecated).
- "ldap_sasl_bind()—Perform an LDAP SASL Bind Request” on page 159 — Asynchronously bind to the
directory using SASL.
- "ldap_sasl_bind_s()—Perform an LDAP SASL Bind Request (Synchronous)” on page 161 —
  Synchronously bind to the directory using SASL.
• “ldap_simple_bind_s()—Perform a Simple LDAP Bind Request (Synchronous)” — Synchronously bind to the directory using simple authentication.
• “ldap_unbind()—Perform an LDAP Unbind Request” on page 209 — Asynchronously unbind from the LDAP server and close the connection.
• “ldap_unbind_s()—Perform an LDAP Unbind Request (Synchronous)” on page 212 — Synchronously unbind from the LDAP server and close the connection.
• “ldap_set_rebind_proc()—Set Rebind Procedure” on page 195 — Sets the entry-point of a routine during the chasing of referrals.

API introduced: V4R3

Top | “Lightweight Directory Access Protocol (LDAP) APIs,” on page 1 | APIs by category

**ldap_simple_bind_s()—Perform a Simple LDAP Bind Request (Synchronous)**

**Syntax**

```c
#include <ldap.h>

int ldap_simple_bind_s(LDAP *ld, const char *dn, const char *passwd)
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The `ldap_simple_bind_s()` function is used to authenticate a distinguished name (DN) to a directory server.

For LDAP V2 servers, after a connection is made to an LDAP server by using the “ldap_open()—Perform an LDAP Open Operation” on page 135, “ldap_init()—Perform an LDAP Initialization Operation” on page 105, or “ldap_ssl_init —Initialize an SSL Connection.” on page 204 APIs, an LDAP bind API must be called before any other LDAP APIs can be called for that connection. For LDAP V3 servers, the bind is optional.

`ldap_simple_bind_s()` performs a synchronous request.

**Authorities and Locks**

No i5/OS authority is required. All authority checking is done by the LDAP server.

**Parameters**

*ld* (Input) Specifies the LDAP pointer returned by a previous call to “ldap_init()—Perform an LDAP Initialization Operation” on page 105, “ldap_ssl_init —Initialize an SSL Connection,” on page 204, or “ldap_open()—Perform an LDAP Open Operation” on page 135.

*dn* (Input) Specifies the Distinguished Name of the entry to bind as.

*passwd* (Input) Specifies the password used in association with DN of the entry in which to bind.

**Return Value**

`LDAP_SUCCESS`

if the request was successful.
another LDAP error
    if the request was not successful.

**Error Conditions**

If `ldap_simple_bind_s()` is not successful, it returns an LDAP error code. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values.

**Error Messages**

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_simple_bind_s API.</td>
</tr>
</tbody>
</table>

**Related Information**

- “ldap_bind()—Perform an LDAP Bind Request” on page 23 — Asynchronously bind to the directory (deprecated).
- “ldap_bind_s()—Perform an LDAP Bind Request (Synchronous)” on page 25 — Synchronously bind to the directory (deprecated).
- “ldap_sasl_bind()—Perform an LDAP SASL Bind Request” on page 159 — Asynchronously bind to the directory using SASL.
- “ldap_sasl_bind_s()—Perform an LDAP SASL Bind Request (Synchronous)” on page 161 — Synchronously bind to the directory using SASL.
- “ldap_simple_bind()—Perform a Simple LDAP Bind Request” on page 196 — Asynchronously bind to the directory using simple authentication.
- “ldap_unbind()—Perform an LDAP Unbind Request” on page 209 — Asynchronously unbind from the LDAP server and close the connection.
- “ldap_unbind_s()—Perform an LDAP Unbind Request (Synchronous)” on page 212 — Synchronously unbind from the LDAP server and close the connection.
- “ldap_set_rebind_proc()—Set Rebind Procedure” on page 195 — Sets the entry-point of a routine during the chasing of referrals.

API introduced: V4R3

---

**ldap_ssl_client_init —Initialize the SSL Library.**

**Syntax**

```c
#include <ldap.h>
#include <ldapssl.h>

int ldap_ssl_client_init(
    char *keyring,
    char *keyring_pw,
    int ssl_timeout,
    int *pSSLReasonCode)
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes
The `ldap_ssl_client_init()` routine is used to initialize the SSL protocol stack for an application process. It should be called once, prior to making any other LDAP calls. Once `ldap_ssl_client_init()` has been successfully called, any subsequent invocations will return a return code of LDAP_SSL_ALREADY_INITIALIZED.

A related API, `ldap_app_ssl_client_init_np()`—Initialize the LDAP Client for a Secure Connection using Digital Certificate Manager (DCM)” on page 15 is available for using Digital Certificate Manager (DCM) Application IDs when authenticating the client to the server. Either `ldap_ssl_client_init()` or `ldap_app_ssl_client_init_np()` (but not both) can be called in an application process.

Although still supported, the use of the “`ldap_ssl_start()—Start a Secure LDAP Connection” on page 207 API is now deprecated. The `ldap_ssl_client_init()` and “`ldap_ssl —Initialize an SSL Connection,” on page 204 or “`ldap_app_ssl_client_init_np()—Initialize the LDAP Client for a Secure Connection using Digital Certificate Manager (DCM)” on page 15 and “`ldap_app_ssl_init_np —Initialize an SSL Connection” on page 18 APIs should be used instead.

**Authorities and Locks**

Read, *R, authority is needed to the selected Certificate Store and Execute, *X, to the associated directories.

**Parameters**

- **keyring**
  
  (Input) Specifies the name of a key database file (with "kdb" extension). The key database file typically contains one or more certificates of certification authorities (CAs) that are trusted by the client. These types of X.509 certificates are also known as trusted roots. A key database can also be used to store the client’s private key(s) and associated client certificate(s). A private key and associated client certificate are required only if the LDAP server is configured to require client and server authentication. If the LDAP server is configured to provide only server authentication, a private key and client certificate are not required.

  A fully-qualified path and filename is recommended. If a filename without a fully-qualified path is specified, the LDAP library will look in the current directory for the file. The key database file specified here must have been created using the Digital Certificate Manager (DCM). If a key database is not supplied, `keyring` is null, the "SYSTEM Certificate Store is used.

- **keyring_pw**
  
  (Input) Specifies the password that is used to protect the contents of the key database. This password is important since it protects the private key stored in the key database. The password was specified when the key database was initially created. A NULL pointer to the password is accepted.

- **ssl_timeout**
  
  (Input) Specifies the SSL timeout value in seconds. The timeout value controls the frequency with which the SSL protocol stack regenerates session keys. If `ssl_timeout` is set to 0, the default value SSLV3_CLIENT_TIMEOUT will be used. Otherwise, the value supplied will be used, provided it is less than or equal to 86,400. If `ssl_timeout` is greater than 86,400, LDAP_PARAM_ERROR is returned.

- **pSSLReasonCode**
  
  (Input) Specifies a pointer to the SSL Reason Code, which provides additional information in the event that an error occurs during initialization of the SSL stack (when `ldap_ssl_client_init()` is called). See QSYSINC/H.LDAPSSL for reason codes that can be returned.

**Example**

See Code disclaimer information for information pertaining to code examples.
The following scenario depicts the recommended calling sequence where the entire set of LDAP transactions are "protected" by using a secure SSL connection, including the dn and password that flow on the "ldap_simple_bind()—Perform a Simple LDAP Bind Request" on page 196.

```c
rc = ldap_ssl_client_init(keyfile, keyfile_pw, timeout, &sslrc);
ld = ldap_ssl_init(ldaphost, ldapport, label);
rc = ldap_set_option( ld, LDAP_OPT_SSL_CIPHER, &ciphers);
rc = ldap_simple_bind_s(ld, binddn, passwd);

...additional LDAP API calls
rc = ldap_unbind( ld);
```

The following scenario depicts using the SASL EXTERNAL mechanism for authenticating the client to the server using the credentials in the SSL certificate:

```c
rc = ldap_ssl_client_init(keyfile, keyfile_pw, timeout, &sslrc);
ld = ldap_ssl_init(ldaphost, ldapport, label);
rc = ldap_set_option( ld, LDAP_OPT_SSL_CIPHER, &ciphers);
rc = ldap_sasl_bind_s(ld, NULL, LDAP_MECHANISM_EXTERNAL, NULL, NULL, NULL);

...additional LDAP API calls
rc = ldap_unbind( ld);
```

Note that the sequence of calls for the deprecated APIs is `ldap_open/init()`, `ldap_ssl_start()`, followed by `ldap_bind()`.

The following ciphers are attempted for the SSL handshake by default, in the order shown.

(Export Version)

- RC4_MD5_EXPORT
- RC2_MD5_EXPORT

(Non-export Version)

- RC4_SHA_US
- RC4_MD5_US
- DES_SHA_US
- 3DES_SHA_US
- RC4_MD5_EXPORT
- RC2_MD5_EXPORT

See "ldap_get_option()—Retrieve LDAP Options" on page 96 and "ldap_set_option() — Set LDAP Options" on page 189 for more information on setting the ciphers to be used.

The `ldap_ssl_client_init()` API includes RSA software. RSA is a trademark of RSA Data Security, Inc.

**Return Value**

- **LDAP_SUCCESS**
  - if the request was successful.

- **another LDAP error**
  - if the request was not successful.

**Error Conditions**

If `ldap_ssl_client_init()` is not successful, it returns an LDAP error code. See "LDAP Client API Error Conditions" on page 300 for possible values for the error codes.
Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_ssl_client_init API.</td>
</tr>
</tbody>
</table>

Related Information

- **ldap_app_ssl_client_init_np()**—Initialize the LDAP Client for a Secure Connection using DCM
- **ldap_ssl_init**—Initialize an SSL Connection.
- **ldap_ssl_environment_init()**—Initialize SSL for a secure connection
- **ldap_start_tls_app_np()**—Start a TLS session with an Application ID
- **ldap_start_tls_s_np()**—Start a TLS session with a Certificate
- **ldap_stop_tls_s_np()**—End a TLS Session
- **ldap_ssl_start()**—Start a Secure LDAP Connection

API introduced: V4R5

### ldap_ssl_environment_init()—Initialize SSL for a secure connection

**Syntax**

```c
#include <ldap.h>
#include <ldapssl.h>

int ldap_ssl_environment_init(
    char *keyfile,  
    char *keyfile_pw, 
    int ssrTimeout, 
    int *pSSLReasonCode)
```

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

The **ldap_ssl_environment_init()** function is used to initialize SSL for a secure connection between a client and server. It has the same parameters as **ldap_ssl_client_init()** but can be called more than once. It returns LDAP_SUCCESS or the appropriate LDAP error code. It does not return LDAP_SSL_ALREADY_INITIALIZED. An application that requires SSL connections to different servers can initialize environments in separate calls to this function, with different keyring files. The environment created is used by all SSL connections established by calling **ldap_ssl_init()** until the next call is made to **ldap_ssl_environment_init()**. Subsequent calls to **ldap_ssl_environment_init()** do not affect existing SSL connections.

Note: Do not call this function when other threads in this process are starting secure connections.
Authorities and Locks
Read, *R, authority is needed to the selected Certificate Store and Execute, *X, to the associated directories.

Parameters

$keyfile$ (Input) Specifies the name of a key database file (with "kdb" extension). The key database file typically contains one or more certificates of certification authorities (CAs) that are trusted by the client. These types of X.509 certificates are also known as trusted roots. A key database can also be used to store the client’s private key(s) and associated client certificate(s). A private key and associated client certificate are required only if the LDAP server is configured to require client and server authentication. If the LDAP server is configured to provide only server authentication, a private key and client certificate are not required.

A fully-qualified path and filename is recommended. If a filename without a fully-qualified path is specified, the LDAP library will look in the current directory for the file. The key database file specified here must have been created using the Digital Certificate Manager (DCM). If a key database is not supplied, $keyfile$ is NULL, the *SYSTEM Certificate Store is used.

$keyfile_psw$ (Input) Specifies the password that is used to protect the contents of the key database. This password is important since it protects the private key stored in the key database. The password was specified when the key database was initially created. NULL may be specified if the keyfile does not have a password.

$sslTimeout$ (Input) Specifies the SSL timeout value in seconds. The timeout value controls the frequency with which the SSL protocol stack regenerates session keys. If $sslTimeout$ is set to 0, the default value SSLV3_CLIENT_TIMEOUT will be used. Otherwise, the value supplied will be used, provided it is less than or equal to 86,400. If $sslTimeout$ is greater than 86,400, LDAP_PARAM_ERROR is returned.

$pSSLReasonCode$ (Output) Specifies a pointer to the SSL Reason Code, which provides additional information in the event that an error occurs during initialization of the SSL stack (when ldap_ssl_client_init() is called). See QSYSINC/H.LDAPSSL for reason codes that can be returned.

Return Value

LDAP_SUCCESS
if the request was successful.

LDAP_NO_MEMORY
if the request was not successful.

LDAP_SSL_INITIALIZE_FAILED
if the request was not successful.

LDAP_PARAM_ERROR
if the timeout value specified is greater than 86400.

LDAP_SSL_NOT_AVAILABLE
if the request was not successful.

Error Conditions
If ldap_ssl_environment_init() is not successful, it returns an LDAP error code. See “LDAP Client API Error Conditions” on page 300 for possible values for the error codes.
Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_ssl_environment_init API.</td>
</tr>
</tbody>
</table>

Related Information

- "ldap_ssl_client_init —Initialize the SSL Library." on page 199 — Initializes SSL library.
- ldap_start_tls_s_np() — Starts a TLS session.
- "ldap_stop_tls_s_np()—End a TLS Session” on page 209 — Ends a TLS session.

API Introduced: V5R4

Idap_ssl_init —Initialize an SSL Connection.

Syntax

```c
#include <ldap.h>
#include <ldapssl.h>

LDAP *ldap_ssl_init(
    char *host,
    int port,
    char *name)
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The `ldap_ssl_init()` routine is used to initialize a secure SSL session with a server. The server is not actually contacted until an operation is performed that requires it, allowing various options to be set after initialization. Once the secure connection is established for the `ld`, all subsequent LDAP messages that flow over the secure connection are encrypted, including the `ldap_simple_bind()`—Perform a Simple LDAP Bind Request” on page 196 parameters, until “ldap_unbind()—Perform an LDAP Unbind Request” on page 209 is called.

Although still supported, the use of the “ldap_ssl_start()—Start a Secure LDAP Connection” on page 207 API is now deprecated. The “ldap_ssl_client_init —Initialize the SSL Library.” on page 199 and `ldap_ssl_init()` or “ldap_app_ssl_client_init_np()—Initialize the LDAP Client for a Secure Connection using DCM” on page 18 APIs should be used instead.

Authorities and Locks

Read, *R, authority is needed to the selected Certificate Store and Execute, *X, to the associated directories.

Parameters

`host`

(Input) Several methods are supported for specifying one or more target LDAP servers, including the following:
Explicit Host List
Specifies the name of the host on which the LDAP server is running. The host parameter may contain a blank-separated list of hosts to try to connect to, and each host may optionally be of the form host:port. If present, the port overrides the parameter port. The following are typical examples:

1=ldap_ssl_init ("server1", ldaps_port, cert_label);
1=ldap_ssl_init ("server2:1200", ldaps_port, cert_label);
1=ldap_ssl_init ("server1:800 server2:2000 server3", ldaps_port, cert_label);

Localhost
If the host parameter is null, the LDAP server will be assumed to be running on the local host.

Default Hosts
LDAP servers, with SSL ports, using the SecureWay "ldap_server_locate()—Locate Suitable LDAP Servers" on page 178 function. The port specified on the call is ignored, since ldap_server_locate() returns the port. For example, the following two are equivalent:

1=ldap_ssl_init (*ldaps://", ldaps_port, cert_label);
1=ldap_ssl_init (LDAPS_URL_PREFIX, LDAPS_PORT, cert_label);

If more than one default server is located, the list is processed in sequence, until an active server is found.

The LDAP URL can include a distinguished name, used as a filter for selecting candidate LDAP servers based on the server’s suffix (or suffixes). If the most significant portion of the DN is an exact match with a server’s suffix (after normalizing for case), the server is added to the list of candidate servers. For example, the following will only return default LDAP servers that have a suffix that supports the specified DN:

1=ldap_ssl_init (*ldaps://cn=fred, dc=austin, dc=ibm, dc=com", LDAPS_PORT, cert_label);

In this case, a server that has a suffix of "dc=austin, dc=ibm, dc=com" would match. If more than one default server is located, the list is processed in sequence, until an active server is found.

If the LDAP URL contains a host name and optional port, the host is used to create the connection. No attempt is made to locate the default server(s), and the DN, if present, is ignored. For example, the following two are equivalent:

1=ldap_ssl_init (*ldaps://myserver", LDAPS_PORT, cert_label);
1=ldap_ssl_init (myserver", LDAPS_PORT, cert_label);

Local Socket
If the host parameter is prefixed with "/", the host parameter is assumed to be the name of a UNIX socket (that is, socket family is AF_UNIX) and port is ignored. This will fail for ldap_ssl_init() because UNIX sockets do not support SSL, nor is it necessary since data will not be flowing over the network.

Host with Privileged Port
If a specified host is prefixed with "privport://", then the LDAP library will use the resvport() function to attempt to obtain one of the reserved ports (512 through 1023), instead of an "ephemeral" port. The search for a reserved port starts at 1023 and stops at 512. If a reserved port cannot be obtained, the function call will fail. For example:

1=ldap_ssl_init ("privport://server1", ldaps_port, cert_label);
1=ldap_ssl_init ("privport://server2:1200", ldaps_port, cert_label);
1=ldap_ssl_init ("privport://server1:800 server2:2000 privport://server3", ldaps_port, cert_label);

port (Input) The port number to which to connect. If the default IANA-assigned SSL port of 636 is desired, LDAPS_PORT should be specified.

name (Input) The name, or label, associated with the client private key/certificate pair in the key database. It is used to uniquely identify a private key/certificate pair, as stored in the key database, and may be something like: Digital ID for Fred Smith.

If the LDAP server is configured to perform Server Authentication, a client certificate is not required (and name can be set to null). If the LDAP server is configured to perform Client and Server Authentication, a client certificate is required. name can be set to null if a default certificate/private key pair has been designated as the default (using Using Ikmgui). Similarly, name can be set to null if there is a single certificate/private key pair in the designated key database.

Example
See Code disclaimer information for information pertaining to code examples.
The following scenario depicts the recommended calling sequence where the entire set of LDAP transactions are “protected” by using a secure SSL connection, including the dn and password that flow on the \texttt{ldap_simple_bind()}:

\begin{verbatim}
rc = ldap_ssl_client_init(keyfile, keyfile_pw, timeout, reasoncode);
ld = ldap_ssl_init(ldaphost, ldapport, label);
rc = ldap_set_option( ld , LDAP_OPT_SSL_CIPHER, &ciphers);
rc = ldap_simple_bind_s(ld, binddn, passwd);
...additional LDAP API calls
rc = ldap_unbind( ld );
\end{verbatim}

The sequence of calls for the deprecated APIs is \texttt{ldap_open/init()}, \texttt{ldap_ssl_start()}, followed by \texttt{ldap_bind()}. See “ldap_get_option()—Retrieve LDAP Options” on page 96 or “ldap_set_option() — Set LDAP Options” on page 189 for more information on setting the ciphers to be used.

### Return Value

**Session Handle**

if the request was successful. If successful, the Session Handle returned by \texttt{ldap_ssl_init()} is a pointer to an opaque data type representing an LDAP session. The “\texttt{ldap_get_option()}—Retrieve LDAP Options” on page 96 and “\texttt{ldap_set_option()} — Set LDAP Options” on page 189 APIs are used to access and set a variety of session-wide parameters. See “\texttt{ldap_get_option()}—Retrieve LDAP Options” on page 96 and “\texttt{ldap_set_option()} — Set LDAP Options” on page 189 for more information.

NULL if the request was not successful.

### Error Conditions

\texttt{ldap_ssl_init()} will return NULL if not successful.

### Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_ssl_init API.</td>
</tr>
</tbody>
</table>

### Related Information

- “\texttt{ldap_init()}—Perform an LDAP Initialization Operation” on page 105 — Perform an LDAP initialization operation.
- “\texttt{ldap_ssl_client_init —Initialize the SSL Library}.” on page 199 — Initializes the SSL library.
- “\texttt{ldap_app_ssl_init_np —Initialize an SSL Connection}” on page 18 — Initializes an SSL Connection.
- “\texttt{ldap_ssl_start()}—Start a Secure LDAP Connection” on page 207 — Creates a secure SSL connection (deprecated).
- “\texttt{ldap_ssl_environment_init()}—Initialize SSL for a secure connection” on page 202 — Initialize SSL for a secure connection between client and server.
- “\texttt{ldap_start_tls_app_np()}—Start a TLS session with an Application ID.”
- “\texttt{ldap_start_tls_s_np()}—Start a TLS session with a Certificate.”
- “\texttt{ldap_stop_tls_s_np()}—End a TLS Session” on page 209 — End a TLS session.
The keyring ld directories. Read, Authorities Connection." Client deprecated. Although "ldap_unbind()—Perform messages and on LDAP ldap_ssl_start()—Start API

```c
#include <ldapssl.h>
#include <ldap.h>

int ldap_ssl_start(  
    LDAP *ld,  
    char *keyring,  
    char *keyring_pw,  
    char *name )
```

Default Public Authority: *USE  
Library Name/Service Program: QSYS/QGLDCLNT  
Threadsafe: Yes

The `ldap_ssl_start()` function is used to start a secure connection (using Secure Sockets Layer (SSL)) to an LDAP server. `ldap_ssl_start()` accepts the `ld` from an "ldap_open()—Perform an LDAP Open Operation" on page 135 and performs an SSL handshake to a server. `ldap_ssl_start()` must be called after `ldap_open()` and prior to `ldap_bind()`. Once the secure connection is established for the `ld`, all subsequent LDAP messages that flow over the secure connection are encrypted, including the `ldap_bind()` parameters, until "ldap_unbind()—Perform an LDAP Unbind Request" on page 209 is called.

Although still supported, the use of the "ldap_ssl_start()—Start a Secure LDAP Connection" API is now deprecated. The "ldap_ssl_client_init —Initialize an SSL Connection." on page 204 or "ldap_app_ssl_client_init_np()—Initialize the LDAP Client for a Secure Connection using DCM” on page 15 and "ldap_app_ssl_init_np —Initialize an SSL Connection” on page 18 APIs should be used instead.

## Authorities and Locks
Read, *R, authority is needed to the selected Certificate Store and Execute, *X, to the associated directories.

### Parameters

- **ld** (Input) Specifies the LDAP pointer returned by a previous call to "ldap_init()—Perform an LDAP Initialization Operation” on page 105, "ldap_ssl_init —Initialize an SSL Connection.” on page 204, or "ldap_open()—Perform an LDAP Open Operation” on page 135

- **keyring** (Input) Specifies the name of a key database file (with "kdb" extension). The key database file typically contains one or more certificates of certification authorities (CAs) that are trusted by the client. These types of X.509 certificates are also known as trusted roots. A key database can also be used to store the client’s private key(s) and associated client certificate(s). A private key and associated client certificate are required only if the LDAP server is configured to require client and server authentication. If the LDAP server is configured to provide only server authentication, a private key and client certificate are not required.

Note: Although still supported, use of the `ldap_ssl_start()` is discouraged (its use has been deprecated). Any application using the `ldap_ssl_start()` API should only use a single key database (per application process).

A fully-qualified path and filename is recommended. If a filename without a fully-qualified path
is specified, the LDAP library will look in the current directory for the file. The key database file
specified here must have been created using Digital Certificate Manager, DCM. If a key database
is not supplied, the default roots are used for trusted Certification Authorities (CAs).

**keyring_pw**

(Input) Specifies the password that is used to protect the contents of the key database. This
password is important since it protects the private key stored in the key database. The password
was specified when the key database was initially created. A NULL pointer is accepted.

**name**

(Input) Specifies the name, or label, associated with the client private key/certificate pair in the
key database. It is used to uniquely identify a private key/certificate pair, as stored in the key
database.

If the LDAP server is configured to perform Server Authentication, a client certificate is not
required (and *name* can be set to null). If the LDAP server is configured to perform Client and
Server Authentication, a client certificate is required. *name* can be set to null if a default
certificate/private key pair has been designated as the default (using Using DCM). Similarly,
name can be set to null if there is a single certificate/private key pair in the designated key
database.

**Return Value**

Skit error code
- If the request was successful.
-1 if *ld* is not set (NULL).

**Error Conditions**

If *ld* is not NULL, *ldap_ssl_start()* returns Skit error code, otherwise it returns -1. See gskssl.h for possible
values of skit error codes.

**Error Messages**

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_ssl_start API.</td>
</tr>
</tbody>
</table>

**Related Information**

- “…ldap_ssl_init”—Initialize an SSL Connection.” on page 204—Initializes an SSL connection.
- “…ldap_ssl_client_init—Initialize the SSL Library.” on page 199—Initializes the SSL library.
- “…ldap_ssl_environment_init()—Initialize SSL for a secure connection” on page 202—Initialize SSL for
  a secure connection between client and server.
- *ldap_start_tls_app_np()*—Start a TLS session with an Application ID
- *ldap_start_tls_s_np()*—Start a TLS session with a Certificate
- *ldap_stop_tls_s_np()*—End a TLS Session” on page 209—End a TLS session.

The *ldap_ssl_start()* API includes RSA software. RSA is a trademark of RSA Data Security, Inc.

API introduced: V4R3

Top | “…Lightweight Directory Access Protocol (LDAP) APIs,” on page 1 | APIs by category
ldap_stop_tls_s_np()—End a TLS Session

Syntax

```c
#include <ldap.h>

int ldap_stop_tls_s_np(
    LDAP *ld)
```

Library Name/Service Program: QSYS/QGLDCLNT
Default Public Authority: *USE
Threadsafe: Yes

The `ldap_stop_tls_s_np()` function is used to end a previously established TLS session to a server. Depending on the server implementation, further communication using the LDAP handle uses the original unsecured connection, or the connection may be closed by the server.

Authorities and Locks
No i5/OS authority is required. All authority checking is done by the LDAP server.

Parameters

`ld` (Input) Specifies the LDAP pointer used in a previous call to `ldap_start_tls_s_np()` or `ldap_start_tls_app_np()`. Must not be NULL.

Return Value

- **LDAP_SUCCESS** if the request was successful.
- another LDAP error if the request was not successful.

Related Information

- `ldap_start_tls_s_np()` — Starts a TLS session with a Certificate.
- `ldap_start_tls_app_np()` — Starts a TLS session with an Application ID.
- "ldap_ssl_environment_init()—Initialize SSL for a secure connection" on page 202 — Initializes SSL for a secure connection between client and server.
- "ldap_ssl_client_init —Initialize the SSL Library." on page 199 — Initializes SSL library.

API Introduced: V5R4
The **ldap_unbind()** function is used to end the connection to the LDAP server and free the resources contained in the *ld* structure.

Once it is called, any open connection to the LDAP server is closed, and the *ld* structure is invalid. The **“ldap_unbind_s()—Perform an LDAP Unbind Request (Synchronous)” on page 212** and **ldap_unbind()** APIs are both synchronous, and can be used interchangeably.

### Authorities and Locks

No i5/OS authority is required.

### Parameters

*ld*  
(Input) Specifies the LDAP pointer returned by a previous call to **“ldap_init()—Perform an LDAP Initialization Operation” on page 105**, **“ldap_ssl_init —Initialize an SSL Connection.” on page 204**, or **“ldap_open()—Perform an LDAP Open Operation” on page 135**.

### Return Value

LDAP_SUCCESS  
if the request was successful.

LDAP error  
if the request was not successful.

### Error Conditions

If **ldap_unbind()** is not successful, it returns an LDAP error code other than LDAP_SUCCESS. See **“LDAP Client API Error Conditions” on page 300** for possible LDAP error code values.

### Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_unbind API.</td>
</tr>
</tbody>
</table>

### Related Information

- **“ldap_bind()—Perform an LDAP Bind Request” on page 23** — Asynchronously bind to the directory (deprecated).
- **“ldap_bind_s()—Perform an LDAP Bind Request (Synchronous)” on page 25** — Synchronously bind to the directory (deprecated).
- **“ldap_sasl_bind()—Perform an LDAP SASL Bind Request” on page 159** — Asynchronously bind to the directory using SASL.
- **“ldap_sasl_bind_s()—Perform an LDAP SASL Bind Request (Synchronous)” on page 161** — Synchronously bind to the directory using SASL.
• “ldap_simple_bind()—Perform a Simple LDAP Bind Request” on page 196 — Asynchronously bind to
the directory using simple authentication.
• “ldap_simple_bind_s()—Perform a Simple LDAP Bind Request (Synchronous)” on page 198 —
Synchronously bind to the directory using simple authentication.
• “ldap_unbind_ext()—Perform an LDAP Unbind Request” — Perform an LDAP Unbind Request
• “ldap_unbind_s()—Perform an LDAP Unbind Request (Synchronous)” on page 212 — Synchronously
unbind from the LDAP server and close the connection.
• “ldap_set_rebind_proc()—Set Rebind Procedure” on page 195 — Sets the entry-point of a routine
during the chasing of referrals.

API introduced: V4R3

Top | “Lightweight Directory Access Protocol (LDAP) APIs,” on page 1 | APIs by category

### ldap_unbind_ext()—Perform an LDAP Unbind Request

Syntax
#include <ldap.h>

```c
int ldap_unbind_ext(
    LDAP *ld,
    LDAPControl **servctrls,
    LDAPControl **clientctrls )
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The `ldap_unbind_ext()` function is used to end the connection to the LDAP server and free the resources
contained in the `ld` structure.

Once it is called, any open connection associated with the LDAP session handle, `ld`, to the LDAP server is
closed, and any resources associated with the handle are disposed of before returning. The `ld` structure is
invalid and cannot be used for any further api calls. The `ldap_unbind_ext()` is synchronous and allows
server and client controls to be included. Note that since there is no server response to an unbind there is
no way to receive a response to a server control sent with an `ldap_unbind_ext()`.

### Authorities and Locks
No i5/OS authority is required.

### Parameters

- `ld` (Input) Specifies the LDAP pointer returned by a previous call to “ldap_init()—Perform an LDAP
  Initialization Operation” on page 105, “ldap_ssl_init—Initialize an SSL Connection.” on page 204,
or “ldap_open()—Perform an LDAP Open Operation” on page 135.

- `serverctrls` (Input) Specifies a list of LDAP server controls. This parameter may be set to null. See LDAP
  Controls for more information about server controls.

- `clientctrls` (Input) Specifies a list of LDAP client controls. This parameter may be set to null. See LDAP
  Controls for more information about client controls.
Return Value
LDAP_SUCCESS
   if the request was successful.

LDAP error
   if the request was not successful.

Error Conditions
If ldap_unbind_ext() is not successful, it returns an LDAP error code other than LDAP_SUCCESS. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values.

Error Messages
The following message may be sent from this function.

Message ID     Error Message Text
CPF3CF2 E       Error(s) occurred during running of ldap_unbind_ext API.

Related Information
• “ldap_bind()—Perform an LDAP Bind Request” on page 23 — Asynchronously bind to the directory (deprecated).
• “ldap_bind_s()—Perform an LDAP Bind Request (Synchronous)” on page 25 — Synchronously bind to the directory (deprecated).
• “ldap_sasl_bind()—Perform an LDAP SASL Bind Request” on page 159 — Asynchronously bind to the directory using SASL.
• “ldap_sasl_bind_s()—Perform an LDAP SASL Bind Request (Synchronous)” on page 161 — Synchronously bind to the directory using SASL.
• “ldap_simple_bind()—Perform a Simple LDAP Bind Request” on page 196 — Asynchronously bind to the directory using simple authentication.
• “ldap_simple_bind_s()—Perform a Simple LDAP Bind Request (Synchronous)” on page 198 — Synchronously bind to the directory using simple authentication.
• “ldap_unbind_s()—Perform an LDAP Unbind Request (Synchronous)” — Synchronously unbind from the LDAP server and close the connection.

API introduced: V5R1

Syntax
#include <ldap.h>

int ldap_unbind_s(
  LDAP *ld)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The ldap_unbind_s() function is used to end the connection to the LDAP server and free the resources contained in the ld structure.
Once it is called, any open connection to the LDAP server is closed, and the ld structure is invalid. The ldap_unbind_s() and “ldap_unbind()—Perform an LDAP Unbind Request” APIs are both synchronous and can be used interchangeably.

**Authorities and Locks**

No i5/OS authority is required.

**Parameters**

ld (Input) Specifies the LDAP pointer returned by a previous call to “ldap_init()—Perform an LDAP Initialization Operation” on page 105, “ldap_ssl_init — Initialize an SSL Connection.” on page 204, or “ldap_open()—Perform an LDAP Open Operation” on page 135.

**Return Value**

LDAP_SUCCESS if the request was successful.

another LDAP error if the request was not successful.

**Error Conditions**

If ldap_unbind_s() is not successful, it returns another LDAP error code. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values.

**Error Messages**

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_unbind_s API.</td>
</tr>
</tbody>
</table>

**Related Information**

- “ldap_bind()—Perform an LDAP Bind Request” — Asynchronously bind to the directory (deprecated).
- “ldap_bind_s()—Perform an LDAP Bind Request (Synchronous)” — Synchronously bind to the directory (deprecated).
- “ldap_sasl_bind()—Perform an LDAP SASL Bind Request” — Asynchronously bind to the directory using SASL.
- “ldap_sasl_bind_s()—Perform an LDAP SASL Bind Request (Synchronous)” — Synchronously bind to the directory using SASL.
- “ldap_simple_bind()—Perform a Simple LDAP Bind Request” — Asynchronously bind to the directory using simple authentication.
- “ldap_simple_bind_s()—Perform a Simple LDAP Bind Request (Synchronous)” — Synchronously bind to the directory using simple authentication.
- “ldap_unbind()—Perform an LDAP Unbind Request” — Asynchronously unbind from the LDAP server and close the connection.

API introduced: V4R3
**ldap_url_parse()—Parse an LDAP URL**

Syntax

```c
#include <ldap.h>

typedef struct ldap_url_desc {
    char *lud_host; /* LDAP host to contact */
    int  lud_port; /* port on host */
    char *lud_dn; /* base for search */
    char **lud_attrs; /* NULL-terminate list of attributes */
    int  lud_scope; /* a valid LDAP_SCOPE... value */
    char *lud_filter; /* LDAP search filter */
    char *lud_string; /* for internal use only */
} LDAPURLDesc;

int ldap_url_parse(
    char   *url,
    LDAPURLDesc **ludpp)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes
```

The `ldap_url_parse()` function breaks down the LDAP URL passed in `url` into its component pieces. The URLs passed in to `ldap_url_parse()` must be in the local codepage. Use “`ldap_url_parse_utf8()—Parse a UTF8 codepage LDAP URL string`” on page 215 for UTF-8 URLs.

The LDAPURLDesc structure returned by this API should be freed with “`ldap_free_urldesc()—Free an LDAP URL Description`” on page 86.

This routine supports the use of LDAP URLs (Uniform Resource Locators). Supported LDAP URLs look like this, where sections in brackets are optional:

```
ldap[s]://[hostport]/[dn]?attributes/?scope/?filter][]
```

where:
- **hostport** is a host name with an optional ":portnumber"
- **dn** is the base DN to be used for an LDAP search operation
- **attributes** is a comma separated list of attributes to be retrieved
- **scope** is one of these three strings: base one sub (default=base)
- **filter** is LDAP search filter as used in a call to `ldap_search`

For example:

```
```

URLs that are wrapped in angle-brackets and/or preceded by "URL:" are also tolerated, including the form `url:ldapurl`.

For example:

```
```

This form also is allowed: `<url:ldapurl>`

For example:

```
```
Authorities and Locks
No i5/OS authority is required.

Parameters
url (Input) Specifies a pointer to the URL string.
ludpp (Output) This result parameter will be set to a LDAPURLDesc structure containing the parsed URL.

Return Value
LDAP_SUCCESS
If successful, an LDAP URL description is allocated, filled in, and ludpp is set to point to it.

other LDAP Error code
If an error occurs, one of these values is returned:

LDAP_URL_ERR_NOTLDAP URL doesn’t begin with “ldap://”
LDAP_URL_ERR_BADSCOPE URL scope string is invalid
LDAP_URL_ERR_MEM can’t allocate memory space

Error Messages
The following message may be sent from this function.

Message ID Error Message Text
CPF3CF2 E Error(s) occurred during running of ldap_url_parse API.

Related Information
- “ldap_free_urldesc()—Free an LDAP URL Description” on page 86 — Frees an LDAP URL description.
- “ldap_is_ldap_url()—Verify LDAP URL” on page 108 — Check a URL string to see if it is an LDAP URL.
- “ldap_url_parse_utf8()—Parse a UTF8 codepage LDAP URL string” — Parse a UTF8 codepage LDAP URL string
- “ldap_url_search()—Perform an LDAP URL Search Operation” on page 217 — Asynchronously search using an LDAP URL.
- “ldap_url_search_s() — Perform an LDAP URL Search Operation (Synchronous)” on page 219 — Synchronously search using an LDAP URL.
- “ldap_url_search_st()—Perform an LDAP URL Search Operation (Timed Synchronous)” on page 221 — Synchronously search using an LDAP URL and a timeout.

API introduced: V4R3

ldap_url_parse_utf8()—Parse a UTF8 codepage LDAP URL string

Syntax
#include <ldap.h>

typedef struct ldap_url_desc {
    char *lud_host; /* LDAP host to contact */
    int lud_port; /* port on host */
    char *lud_dn; /* base for search */
    char **lud_attrs; /* NULL-terminate list of attributes */
    int lud_scope; /* a valid LDAP_SCOPE... value */
    char *lud_filter; /* LDAP search filter */
    char *lud_string; /* for internal use only */
} LDAPURLDesc;

int ldap_url_parse_utf8(
    char *url,
    LDAPURLDesc **ludpp
)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The `ldap_url_parse_utf8()` function breaks down the UTF8 codepage LDAP URL string passed in `url` into its component pieces. To parse URLs in the local codepage, use the `ldap_url_parse()`—Parse an LDAP URL on page 214.

The LDAPURLDesc structure returned by this API should be freed with the `ldap_free_urldesc()`—Free an LDAP URL Description” on page 86.

This routine supports the use of LDAP URLs (Uniform Resource Locators). Supported LDAP URLs look like this, where sections in brackets are optional:

```
ldap[s]://[hostport][/][dn][?][attributes][?][scope][?][filter]]]
```

where:

- **hostport** is a host name with an optional ":portnumber"
- **dn** is the base DN to be used for an LDAP search operation
- **attributes** is a comma separated list of attributes to be retrieved
- **scope** is one of these three strings: base one sub (default=base)
- **filter** is LDAP search filter as used in a call to `ldap_search`

For example:

```
```

URLs that are wrapped in angle-brackets and/or preceded by "URL:" are also tolerated, including the form `url:ldapurl`.

For example:

```
```

This form also is allowed: `<url:ldapurl>`.

For example:

```
```

**Authorities and Locks**

No i5/OS authority is required.
Parameters

`url`  (Input) A pointer to the UTF8 codepage URL string.

`ludpp` (Output) This result parameter will be set to a LDAPURLDesc structure containing the parsed URL.

Return Value

LDAP_SUCCESS

If successful, an LDAP URL description is allocated, filled in, and `ludpp` is set to point to it.

other LDAP Error code

If an error occurs, one of these values is returned:

- LDAP_URL_ERR_NOTLDAP  URL doesn’t begin with “ldap://”
- LDAP_URL_ERR_BADSCOPE URL scope string is invalid
- LDAP_URL_ERR_MEM  can’t allocate memory space

Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_url_parse_utf8 API.</td>
</tr>
</tbody>
</table>

Related Information

- “ldap_url_parse()—Parse an LDAP URL” on page 214 — Parse an LDAP URL.
- “ldap_free_urldesc()—Free an LDAP URL Description” on page 86 — Frees an LDAP URL description.
- “ldap_is_ldap_url()—Verify LDAP URL” on page 108 — Check a URL string to see if it is an LDAP URL.
- “ldap_url_search()—Perform an LDAP URL Search Operation” — Asynchronously search using an LDAP URL.
- “ldap_url_search_s() — Perform an LDAP URL Search Operation (Synchronous)” on page 219 — Synchronously search using an LDAP URL.
- “ldap_url_search_st()—Perform an LDAP URL Search Operation (Timed Synchronous)” on page 221 — Synchronously search using an LDAP URL and a timeout.

API introduced: V5R1

Syntax

```
#include <ldap.h>

int ldap_url_search(
    LDAP *ld,
    char *url,
    int attrsonly)
```

Lightweight Directory Access Protocol (LDAP) APIs  217
The ldap_url_search() function is used to perform an asynchronous LDAP search based on the contents of the url parameter. This function acts like "ldap_search()—Perform an LDAP Search Operation" on page 164 except that the search parameters are specified by the URL.

This routine supports the use of LDAP URLs (Uniform Resource Locators).

LDAP URLs look like this:

```
ldap[s]://[hostport][/[dn][?[attributes][?[scope][?[filter]]]]]
```

where:
- **hostport** is a host name with an optional ":portnumber"
- **dn** is the base DN to be used for an LDAP search operation
- **attributes** is a comma separated list of attributes to be retrieved
- **scope** is one of these three strings: base one sub (default=base)
- **filter** is LDAP search filter as used in a call to ldap_search

For example:
```
```

URLs that are wrapped in angle-brackets and/or preceded by "URL:" are also tolerated, including the form url:ldapurl.

For example:
```
```

This form also is allowed: <url:ldapurl>.

For example:
```
```

Notes:
1. For search operations, if hostport is omitted, host and port for the current connection are used. If hostport is specified, and is different from the host and port combination used for the current connection, the search is directed to that host and port, instead of using the current connection. In this case, the underlying referral mechanism is used to bind to hostport.
2. If the LDAP URL does not contain a search filter, the filter defaults to "(objectClass=*)".

**Authorities and Locks**

None.

**Return Value**

**Message ID of the Operation Initiated**

-1 if the request was not successful. A subsequent call to "ldap_result()—Retrieve Result of an Asynchronous LDAP Operation" on page 156 can be used to obtain the result.
Error Conditions

If `ldap_url_search()` is not successful, -1 will be returned setting the session error parameters (`ld_error`) in the LDAP structure appropriately, which can be obtained by using `ldap_get_lderrno()`—Retrieve Error Information” on page 93. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values.

Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_url_search API.</td>
</tr>
</tbody>
</table>

Related Information

- “ldap_free_urldesc()—Free an LDAP URL Description” on page 86 — Frees an LDAP URL description.
- “ldap_is_ldap_url()—Verify LDAP URL” on page 108 — Check a URL string to see if it is an LDAP URL.
- “ldap_url_parse()—Parse an LDAP URL” on page 214 — Break up an LDAP URL string into its components.
- “ldap_url_search_s() — Perform an LDAP URL Search Operation (Synchronous)” — Synchronously search using an LDAP URL.
- “ldap_url_search_st()—Perform an LDAP URL Search Operation (Timed Synchronous)” on page 221 — Synchronously search using an LDAP URL and a timeout.

API introduced: V4R3

**ldap_url_search_s() — Perform an LDAP URL Search Operation (Synchronous)**

Syntax

```
#include <ldap.h>

int ldap_url_search_s(
    LDAP *ld,
    char *url,
    int attrsonly,
    LDAPMessage **res)
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The `ldap_url_search_s()` function is used to perform a synchronous LDAP search based on the contents of `url` parameter.

This function acts like “ldap_search_s()—Perform an LDAP Search Operation (Synchronous)” on page 171 except that the search parameters are specified by the URL.

This routine support the use of LDAP URLs (Uniform Resource Locators).
LDAP URLs look like this:

```
ldap[s]://[hostport]/[/dn?attributes?[scope]?[filter]]
```

where:

- **hostport** is a host name with an optional ":portnumber"
- **dn** is the base DN to be used for an LDAP search operation
- **attributes** is a comma separated list of attributes to be retrieved
- **scope** is one of these three strings: base one sub (default=base)
- **filter** is LDAP search filter as used in a call to `ldap_search`

For example:

```
```

URLs that are wrapped in angle-brackets and/or preceded by "URL:" are also tolerated, including the form `url:ldapurl`.

For example:

```
```

This form also is allowed: `<url:ldapurl>`.

For example:

```
```

**Notes:**

1. For search operations, if hostport is omitted, host and port for the current connection are used. If hostport is specified, and is different from the host and port combination used for the current connection, the search is directed to that host and port, instead of using the current connection. In this case, the underlying referral mechanism is used to bind to hostport.

2. If the LDAP URL does not contain a search filter, the filter defaults to "(objectClass=*)".

**Authorities and Locks**

No i5/OS authority is required. All authority checking is done by the LDAP server.

**Parameters**

- **ld** (Input) Specifies the LDAP pointer returned by a previous call to "ldap_init()—Perform an LDAP initialization Operation" on page 105, "ldap_ssl_init—Initialize an SSL Connection." on page 204, or "ldap_open()—Perform an LDAP Open Operation” on page 135.
- **url** (Input) Specifies a pointer to the URL string.
- **attrsonly** (Input) Specifies attribute information. Set to 1 to request attribute types only. Set to 0 to request both attribute types and attribute values.
- **res** (Output) Contains the result of the synchronous search operation. This result should be passed to the LDAP parsing routines (see "ldap_first_entry()—Retrieve First LDAP Entry” on page 81, "ldap_next_entry()—Retrieve Next LDAP Entry” on page 131, and so on). The caller is responsible for freeing res with "ldap_msgfree()—Free LDAP Result Message” on page 125.

**Return Value**

`LDAP_SUCCESS` if the request was successful.
another LDAP error
  if the request was not successful.

Error Conditions
If `ldap_url_search_s` is not successful, it returns an LDAP error code other than LDAP_SUCCESS. See "LDAP Client API Error Conditions" on page 300 for possible LDAP error code values.

Error Messages
The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_url_search_s API.</td>
</tr>
</tbody>
</table>

Related Information
- "ldap_free_urldesc()—Free an LDAP URL Description" on page 86 — Frees an LDAP URL description.
- "ldap_url_parse()—Parse an LDAP URL" on page 214 — Extract information from results.
- "ldap_is_ldap_url()—Verify LDAP URL" on page 108 — Check a URL string to see if it is an LDAP URL.
- "ldap_url_search()—Perform an LDAP URL Search Operation” on page 217 — Asynchronously search using an LDAP URL.
- "ldap_url_search_st()—Perform an LDAP URL Search Operation (Timed Synchronous)” — Synchronously search using an LDAP URL and a timeout.

API introduced: V4R3

```
#include <sys/time.h>
#include <ldap.h>

int ldap_url_search_st( LDAP* ld,
                         char* url,
                         int attrsonly,
                         struct timeval* timeout,
                         LDAPMessage** res );
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The `ldap_url_search_st()` function is used to perform a synchronous LDAP search with a specified timeout based on the contents of the `url` parameter.

This function acts like "ldap_search_st()—Perform an LDAP Search Operation (Timed Synchronous)” on page 173 except that the search parameters are retrieved from the URL.

This routine supports the use of LDAP URLs (Uniform Resource Locators).
LDAP URLs look like this:

```
ldap[s]://[hostport]/[dn][attributes][scope][filter]]]]]]
```

where:
- **hostport** is a host name with an optional ":portnumber"
- **dn** is the base DN to be used for an LDAP search operation
- **attributes** is a comma separated list of attributes to be retrieved
- **scope** is one of these three strings: base one sub (default=base)
- **filter** is LDAP search filter as used in a call to **ldap_search**

For example:
```
ldap://example.ibm.com/c=US,o,description?one?o=ibm
```

URLs that are wrapped in angle-brackets and/or preceded by "URL:" are also tolerated, including the form **url:ldapurl**.

For example:
```
URL:ldap://example.ibm.com/c=US,o,description?one?o=ibm
```

This form also is allowed: `<url:ldapurl>`.

For example:
```
<URL:ldap://example.ibm.com/c=US,o,description?one?o=ibm>
```

**Notes:**
1. For search operations, if hostport is omitted, host and port for the current connection are used. If hostport is specified, and is different from the host and port combination used for the current connection, the search is directed to that host and port, instead of using the current connection. In this case, the underlying referral mechanism is used to bind to hostport.
2. If the LDAP URL does not contain a search filter, the filter defaults to "(objectClass=*)."

**Authorities and Locks**

No i5/OS authority is required. All authority checking is done by the LDAP server.

**Parameters**

- **Id** (Input) Specifies the LDAP pointer returned by a previous call to **"ldap_init()—Perform an LDAP initialization Operation"** on page 105, **"ldap_ssl_init—Initialize an SSL Connection."** on page 204, or **"ldap_open()—Perform an LDAP Open Operation"** on page 135.
- **url** (Input) Specifies a pointer to the URL string.
- **attrsonly** (Input) Specifies attribute information. Set to 1 to request attribute types only. Set to 0 to request both attribute types and attribute values.
- **timeout** (Input) Specifies a timeout value for a synchronous search issued by the **"ldap_url_search_st()"** routine.
- **res** (Output) Contains the result of the synchronous search operation. This result should be passed to the LDAP parsing routines (see **"ldap_first_entry()—Retrieve First LDAP Entry"** on page 81, **"ldap_next_entry()—Retrieve Next LDAP Entry"** on page 131, and so on). The caller is responsible for freeing **res** with **"ldap_msgfree()—Free LDAP Result Message"** on page 125.
Return Value

**LDAP_SUCCESS**
if the request was successful.

**another LDAP error**
if the request was not successful.

Error Conditions

If `ldap_url_search_st()` is not successful, it returns an LDAP error code other than LDAP_SUCCESS. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values.

Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_url_search_st API.</td>
</tr>
</tbody>
</table>

Related Information

- “ldap_free urldesc()—Free an LDAP URL Description” on page 86 — Frees an LDAP URL description.
- “ldap_url_parse()—Parse an LDAP URL” on page 214 — Extract information from results.
- “ldap_is ldap_url()—Verify LDAP URL” on page 108 — Check a URL string to see if it is an LDAP URL.
- “ldap_url_search()—Perform an LDAP URL Search Operation” on page 217 — Asynchronously search using an LDAP URL.
- “ldap_url_search_s() — Perform an LDAP URL Search Operation (Synchronous)” on page 219 — Synchronously search using an LDAP URL.

API introduced: V4R3

```
#include <ldap.h>

void ldap_value_free(
    char **vals
)
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The `ldap_value_free()` function frees the memory allocated by the “ldap_get_values()—Retrieve a Set of Attribute Values from an Entry” on page 102 function.

Authorities and Locks

No i5/OS authority is required.
Parameters

vals  (Input) Specifies a pointer to a null-terminated array of attribute values, as returned by "ldap_get_values()—Retrieve a Set of Attribute Values from an Entry" on page 102.

Return Value

None

Error Conditions

ldap_value_free() API does not return an error code.

Error Messages

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_value_free API.</td>
</tr>
</tbody>
</table>

Related Information

- "ldap_get_values()—Retrieve a Set of Attribute Values from an Entry" on page 102 — Return an attribute’s values.
- "ldap_get_values_len()—Retrieve a Set of Binary Attribute Values" on page 104 — Return an attribute’s binary values.
- "ldap_count_values()—Retrieve Count of Attribute Values” on page 40 — Return number of values.
- "ldap_count_values_len()—Retrieve Count of Binary Attribute Values” on page 41 — Return number of binary values.
- "ldap_value_free_len()—Free Memory Allocated by ldap_get_values_len()” — Free memory allocated by ldap_get_values_len().

API introduced: V4R3

Syntax

```c
#include <ldap.h>

struct berval {
    unsigned long bv_len;
    char *bv_val;
};

void ldap_value_free_len(
    struct berval **vals)
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes
The **ldap_value_free_len()** function frees the memory allocated by the "ldap_get_values_len()—Retrieve a Set of Binary Attribute Values” on page 104 function.

**Authorities and Locks**

No i5/OS authority is required.

**Parameters**

bvals  (Input) Specifies a pointer to a null-terminated array of pointers to berval structures, as returned by "ldap_get_values_len()—Retrieve a Set of Binary Attribute Values” on page 104.

**Return Value**

None

**Error Conditions**

ldap_value_free_len() API does not return an error code.

**Error Messages**

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_value_free_len API.</td>
</tr>
</tbody>
</table>

**Related Information**

- "ldap_get_values()—Retrieve a Set of Attribute Values from an Entry” on page 102 — Return an attribute’s values.
- "ldap_get_values_len()—Retrieve a Set of Binary Attribute Values” on page 104 — Return an attribute’s binary values.
- "ldap_count_values()—Retrieve Count of Attribute Values” on page 40 — Return number of values.
- "ldap_count_values_len()—Retrieve Count of Binary Attribute Values” on page 41 — Return number of binary values.
- "ldap_value_free_len()—Free Memory Allocated by ldap_get_values_len()” on page 224 — Free memory allocated by ldap_get_values().

API introduced: V4R3

---

**ldap_version — Obtain LDAP Version and SSL Cipher Information**

**Syntax**

```c
#include <ldap.h>
#include <ldapssl.h>

int ldap_version(
    LDAPVersion  *version
);
```

**Default Public Authority:** *USE

**Library Name/Service Program:** QSYS/QGLDCLNT

**Threadsafe:** Yes
The `ldap_version()` routine is used to return the toolkit version (multiplied by 100). It also sets information in the LDAPVersion structure.

**Authorities and Locks**
No i5/OS authority is required.

**Parameters**

`version`
(Input) Specifies the address of an LDAPVersion structure that contains the following returned values:

- `sdk_version`: Toolkit version, multiplied by 100.
- `protocol_version`: Highest LDAP protocol supported, multiplied by 100.
- `SSL_version`: SSL version supported, multiplied by 100.
- `security_level`: Level of encryption supported, in bits. Set to `LDAP_SECURITY_NONE` if SSL not enabled.
- `ssl_max_cipher`: A string containing the default ordered set of ciphers supported by this installation. See “`LDAP_OPT_SSL_CIPHER` on page 192” in `ldap_set_option()` for more information about changing the set of ciphers used to negotiate the secure connection with the server.
- `sdk_vendor`: A pointer to a static string that identifies the supplier of the LDAP library. This string should not be freed by the application.
- `sdk_build_level`: A pointer to a static string that identifies the build level, including the date when the library was built. This string should not be freed by the application.

**Return Value**

Software Developer Toolkit Version
Sets information in the LDAPVersion structure and return the SDK VERSION.

**Error Conditions**
The `ldap_version()` API does not return an error code.

**Error Messages**
The following message may be sent from this function.

**Message ID**  **Error Message Text**
CPF3CF2 E  Error(s) occurred during running of `ldap_version` API.

API introduced: V4R5

---

**ldap_xlate_local_to_unicode()— Convert String From the Local Code Page to UCS-2 (or UNICODE) Encoding**

**Syntax**

```c
#include <ldap.h>

int ldap_xlate_local_to_unicode(
    char *inbufp,
    unsigned long *inlenp,
    char *outbufp,
    unsigned long *outlenp )
```

---

226  IBM Systems - iSeries: Lightweight Directory Access Protocol (LDAP) APIs
The `ldap_xlate_local_to_unicode()` API is used to convert a string from the local code page to the UCS-2 encoding as defined by ISO/IEC 10646-1. This same set of characters is also defined in the UNICODE standard.

**Authorities and Locks**
No i5/OS authority is required.

**Parameters**
- `inbufp` (Input) A pointer to the address of the input buffer containing the data to be translated.
- `inlenp` (Input) Length in bytes of the `inbufp` buffer. This value is decremented when the conversion is done, such that on return it indicates the length of `inlenp` buffer that is left to be translated.
- `outbufp` (Output) A pointer to the address of the output buffer for translated data.
- `outlenp` (Output) Length in bytes of the `outbufp` buffer. This value is decremented when the conversion is done, such that on return it indicates the length of `outlenp` buffer space left available for translated data.

Note that in general, the output buffer should be three times as large as the input buffer if the intent is to translate the entire input buffer in a single call.

**Return Value**
- LDAP_SUCCESS if the request was successful.
- another LDAP error code if the request was not successful.

**Error Conditions**
The `ldap_xlate_local_to_unicode()` API will return an LDAP error code other than LDAP_SUCCESS if not successful. See [“LDAP Client API Error Conditions” on page 300](#) for possible LDAP error code values.

**Error Messages**
The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_xlate_local_to_unicode API.</td>
</tr>
</tbody>
</table>

**Related Information**
- [“ldap_xlate_utf8_to_local() — Convert String From the UTF-8 Encoding to Local Code Page” on page 231](#) — Convert string from UTF-8 to local code page.
- [“ldap_xlate_local_to_utf8() — Convert String From the Local Code Page to UTF-8 Encoding” on page 228](#) — Convert string from local to UTF-8 code page.
- [“ldap_xlate_unicode_to_local() — Convert String From the UCS-2 (or UNICODE) Encoding to Local Code Page” on page 229](#) — Convert string from UCS-2 to local code page.
ldap_xlate_local_to_utf8()— Convert String From the Local Code Page to UTF-8 Encoding

Syntax
#include <ldap.h>

int ldap_xlate_local_to_utf8(
    char    *inbufp,
    unsigned long *inlenp,
    char    *outbufp,
    unsigned long *outlenp
)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes

The ldap_xlate_local_to_utf8() API is used to convert a string from the local code page to a UTF-8 encoding (which is used by LDAP when communicating with an LDAP V3 compliant server).

Authorities and Locks
No i5/OS authority is required.

Parameters

inbufp (Input) A pointer to the address of the input buffer containing the data to be translated.

inlenp (Input) Length in bytes of the inbufp buffer. This value is decremented when the conversion is done, such that on return it indicates the length of inlenp buffer that is left to be translated.

outbufp (Output) A pointer to the address of the output buffer for translated data.

outlenp (Output) Length in bytes of the outbufp buffer. This value is decremented when the conversion is done, such that on return it indicates the length of outlenp buffer space left available for translated data.

Note that in general, the output buffer should be three times as large as the input buffer if the intent is to translate the entire input buffer in a single call.
Return Value
LDAP_SUCCESS
if the request was successful.

another LDAP error code
if the request was not successful.

Error Conditions
The ldap_xlate_local_to_utf8() will return an LDAP error code if not successful. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values.

Error Messages
The following message may be sent from this function.

Message ID   Error Message Text
CPF3CF2 E   Error(s) occurred during running of ldap_xlate_local_to_utf8 API.

Related Information
• “ldap_xlate_utf8_to_local() — Convert String From the UTF-8 Encoding to Local Code Page” on page 231 — Convert string from UTF-8 to local code page.
• “ldap_xlate_local_to_unicode()— Convert String From the Local Code Page to UCS-2 (or UNICODE) Encoding” on page 226 — Convert string from the local to UCS-2 code page.
• “ldap_xlate_unicode_to_local() — Convert String From the UCS-2 (or UNICODE) Encoding to Local Code Page” — Convert string from UCS-2 to local code page.
• “ldap_get_iconv_local_codepage()— Get the Active LDAP Code Page” on page 92 — Get the active LDAP code page.
• “ldap_set_iconv_local_codepage() — Set the Active LDAP Code Page” on page 185 — Set the active LDAP code page.
• “ldap_set_iconv_local_charset()— Set the Active LDAP Character Set” on page 183 — Set the active LDAP character set.
• “ldap_set_locale() — Change the Locale Used by LDAP” on page 188 — Change the locale used by LDAP.
• “ldap_get_locale()— Get Active LDAP Locale” on page 95 — Get the locale used by LDAP.

API introduced: V4R5

ldap_xlate_unicode_to_local() — Convert String From the UCS-2 (or UNICODE) Encoding to Local Code Page

Syntax
#include <ldap.h>

int ldap_xlate_unicode_to_local(
    char *inbufp,
    unsigned long *inlenp,
    char *outbufp,
    unsigned long *outlenp )
The `ldap_xlate_unicode_to_local()` API is used to convert a UCS-2 encoded string to the local code page encoding.

It is important to note that translation of strings from a UCS-2 (or UNICODE) encoding to local code page may result in loss of data when one or more characters in the UCS-2 encoding cannot be represented in the local code page. When this occurs, a substitution character replaces any UCS-2 characters that cannot be converted to the local code page.

**Authorities and Locks**
No i5/OS authority is required.

**Parameters**
- `inbufp` (Input) A pointer to the address of the input buffer containing the data to be translated.
- `inlenp` (Input) Length in bytes of the `inbufp` buffer. This value is decremented when the conversion is done, such that on return it indicates the length of `inlenp` buffer that is left to be translated.
- `outbufp` (Output) A pointer to the address of the output buffer for translated data.
- `outlenp` (Output) Length in bytes of the `outbufp` buffer. This value is decremented when the conversion is done, such that on return it indicates the length of `outlenp` buffer space left available for translated data.
  
  Note that in general, the output buffer should be three times as large as the input buffer if the intent is to translate the entire input buffer in a single call.

**Return Value**
- `LDAP_SUCCESS` if the request was successful.
- another LDAP error code if the request was not successful.

**Error Conditions**
The `ldap_xlate_unicode_to_local()` API will return an LDAP error code if not successful. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values.

**Error Messages**
The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of ldap_xlate_unicode_to_local API.</td>
</tr>
</tbody>
</table>

**Related Information**
- “ldap_xlate_utf8_to_local() — Convert String From the UTF-8 Encoding to Local Code Page” on page 231 — Convert string from UTF-8 to local code page.
“ldap_xlate_local_to_utf8()— Convert String From the Local Code Page to UTF-8 Encoding” on page 228 — Convert string from local to UTF-8 code page.

“ldap_xlate_local_to_unicode()— Convert String From the Local Code Page to UCS-2 (or UNICODE) Encoding” on page 226 — Convert string from local to UCS-2 code page.

“ldap_get_iconv_local_codepage()— Get the Active LDAP Code Page” on page 92 — Get the active LDAP code page.

“ldap_set_iconv_local_codepage()— Set the Active LDAP Code Page” on page 185 — Set the active LDAP code page.

“ldap_set_iconv_local_charset()— Set the Active LDAP Character Set” on page 183 — Set the active LDAP character set.

“ldap_set_locale()— Change the Locale Used by LDAP” on page 188 — Change the locale used by LDAP.

“ldap_get_locale()— Get Active LDAP Locale” on page 95 — Get the locale used by LDAP.

API introduced: V4R5

The ldap_xlate_utf8_to_local() API is used to convert a UTF-8 encoded string to the local code page encoding.

It is important to note that translation of strings from a UTF-8 encoding to local code page may result in loss of data when one or more characters in the UTF-8 encoding cannot be represented in the local code page. When this occurs, a substitution character replaces any UTF-8 characters that cannot be converted to the local code page.

**Authorities and Locks**
No i5/OS authority is required.

**Parameters**

- **inbufp** (Input) A pointer to the address of the input buffer containing the data to be translated.
- **inlenp** (Input) Length in bytes of the *inbufp* buffer. This value is decremented when the conversion is done, such that on return it indicates the length of the *inlenp* buffer that is left to be translated.
- **outbufp** (Output) A pointer to the address of the output buffer for translated data.

**Syntax**

```c
#include <ldap.h>

int ldap_xlate_utf8_to_local(
    char *inbufp,
    unsigned long *inlenp,
    char *outbufp,
    unsigned long *outlenp)
```

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDCLNT
Threadsafe: Yes
**Return Value**

**LDAP_SUCCESS**
if the request was successful.

another LDAP error code
if the request was not successful.

**Error Conditions**

The `ldap_xlate_utf8_to_local()` will return an LDAP error code if not successful. See “LDAP Client API Error Conditions” on page 300 for possible LDAP error code values.

**Error Messages**

The following message may be sent from this function.

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3CF2 E</td>
<td>Error(s) occurred during running of <code>ldap_xlate_utf8_to_local</code> API.</td>
</tr>
</tbody>
</table>

**Related Information**

- “`ldap_xlate_local_to_utf8()`— Convert String From the Local Code Page to UTF-8 Encoding” on page 228 — Convert string from the local to UTF-8 code page.
- “`ldap_xlate_local_to_unicode()`— Convert String From the Local Code Page to UCS-2 (or UNICODE) Encoding” on page 226 — Convert string from the local to UCS-2 code page.
- “`ldap_xlate_unicode_to_local()`— Convert String From the UCS-2 (or UNICODE) Encoding to Local Code Page” on page 229 — Convert string from UCS-2 to local code page.
- “`ldap_get_iconv_local_codepage()`— Get the Active LDAP Code Page” on page 92 — Get the active LDAP code page.
- “`ldap_set_iconv_local_codepage()` — Set the Active LDAP Code Page” on page 185 — Set the active LDAP code page.
- “`ldap_get_iconv_local_charset()`— Set the Active LDAP Character Set” on page 183 — Set the active LDAP character set.
- “`ldap_set_locale()` — Change the Locale Used by LDAP” on page 188 — Change the locale used by LDAP.
- “`ldap_get_locale()`— Get Active LDAP Locale” on page 95 — Get the locale used by LDAP.

API introduced: V4R5

---

**Configure Directory Server (QgdCfgDirSvr)**

Required Parameter Group:

<table>
<thead>
<tr>
<th>Required Parameter Group:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
</tbody>
</table>
The Configure Directory Server (QgldCfgDirSvr) API creates the initial directory server configuration. This includes identifying the library that will contain the underlying database objects, the administrator of the server, and the initial set of suffixes to be present on the server.

**Authorities and Locks**

*ALLOBJ and *IOSYSCFG special authority is required to use this API.

**Required Parameter Group**

**Input data**

**INPUT; CHAR(*)**

Data that describes the desired directory server configuration. The content and format of this structure are determined by the format name. See "Format of Input Data" for a description of these formats.

**Length of input data**

**INPUT; BINARY(4)**

The length of the input data structure.

**Format name**

**INPUT; CHAR(8)**

The content and format of the input configuration data. The possible format name follows:

CFGD0100 Configure Directory Server.

See "Format of Input Data" for a description of these formats.

**Error code**

**I/O; CHAR(*)**

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

**Format of Input Data**

For details about the format of the input data, see the following sections. For details about the fields in each format, see "Field Descriptions" on page 234.

**CFGD0100 Format**

This format is used to provide initial configuration data about the directory server.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td><strong>Field</strong></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>BINARY(4) Offset to database path</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>BINARY(4) Length of database path</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>BINARY(4) Offset to administrator distinguished name (DN)</td>
</tr>
<tr>
<td>Offset</td>
<td>Type</td>
<td>Field</td>
</tr>
<tr>
<td>--------</td>
<td>--------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>32</td>
<td>20</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td></td>
<td>CHAR(*)</td>
<td>Database path</td>
</tr>
<tr>
<td></td>
<td>CHAR(*)</td>
<td>Administrator DN</td>
</tr>
<tr>
<td></td>
<td>CHAR(*)</td>
<td>Administrator password</td>
</tr>
</tbody>
</table>

### Field Descriptions

**Administrator DN.** The distinguished name of a directory object that has access to all objects in the directory. This field is specified in UTF-16 (CCSID 13488).

**Administrator password.** The password used when you connect to the directory as the administrator. This field is specified in UTF-16 (CCSID 13488).

**Database path.** The path to an existing library containing the directory database objects. This is an integrated file system path name, for example, /QSYS.LIB/QDIRSRV.LIB. The library must exist in a system ASP or a basic user ASP (ASP value of 1 to 32). The library cannot exist in an independent ASP (ASP value greater than 32). This field is specified in UTF-16 (CCSID 13488).

**Displacement to next suffix.** The displacement, in bytes, from the start of the current suffix entry to the next suffix entry.

**Displacement to suffix name.** The displacement, in bytes, from the start of the current suffix entry to the suffix name field.

**Length of administrator DN.** The length, in UTF-16 (CCSID 13488) characters, of the administrator DN

**Length of administrator password.** The length, in UTF-16 (CCSID 13488) characters, of the administrator password field.

**Length of database path.** The length, in UTF-16 (CCSID 13488) characters, of the database path field.

**Length of suffix name.** The length, in UTF-16 (CCSID 13488) characters, of the suffix name field.

**Number of suffixes.** The number of suffixes present in the suffix list.

**Offset to administrator DN.** The offset, in bytes, from the start of the input data to the administrator DN field.
Offset to administrator password. The offset, in bytes, from the start of the input data to the administrator password field.

Offset to database path. The offset, in bytes, from the start of the input data to the database path field.

Offset to suffixes. The offset, in bytes, from the start of the input data to the list of suffixes.

Reserved. A reserved field. This field must be set to zero.

Suffixes. The list of suffixes to be present on the server. At least one must be present in the initial configuration.

Suffix name. The distinguished name of the root of a directory tree present on the server. This field is specified in UTF-16 (CCSID 13488).

Error Messages

<table>
<thead>
<tr>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Library &amp;1 not found.</td>
</tr>
<tr>
<td>Object name not a QSYS object.</td>
</tr>
<tr>
<td>Memory allocation error.</td>
</tr>
<tr>
<td>Administrator DN not valid.</td>
</tr>
<tr>
<td>Suffix not valid.</td>
</tr>
<tr>
<td>*ALLOBJ and *IOSYSCFG special authority required.</td>
</tr>
<tr>
<td>Database path not valid.</td>
</tr>
<tr>
<td>Validation list not created.</td>
</tr>
<tr>
<td>Server configuration cannot be modified while the server is active.</td>
</tr>
<tr>
<td>Database library must be in system ASP or basic user ASP.</td>
</tr>
</tbody>
</table>

Error Messages

API introduced: V4R3

List Authorized Users (QSYLAUTU) API

Required Parameter Group:

<table>
<thead>
<tr>
<th>1</th>
<th>Qualified user space name</th>
<th>Input</th>
<th>Char(20)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Format name</td>
<td>Input</td>
<td>Char(8)</td>
</tr>
<tr>
<td>3</td>
<td>Error code</td>
<td>I/O</td>
<td>Char(*)</td>
</tr>
</tbody>
</table>

Default Public Authority: *USE
Threadsafe: Yes

The List Authorized Users (QSYLAUTU) API puts a list of authorized system users into a user space.

This API provides information similar to the Display Authorized Users (DSPAUTUSR) command.

Authorities and Locks

User Space Authority

*CHANGE

Authority to Library Containing User Space

*EXECUTE
Authority to User Profiles in List of Authorized Users

*READ, only those profiles that you have *READ authority to are returned.

required parameter group

Qualified user space name

INPUT; CHAR(20)

The name of the existing user space that the list of authorized users is returned to. The first 10 characters specify the user space name, and the second 10 characters specify the library.

You can use these special values for the library name:

* CURLIB The current library is searched for the user space. If there is no current library, QGPL (general purpose library) is used.
* LBL The library list is searched for the user space.

Format name

INPUT; CHAR(8)

The name of the format used to list the authorized users.

You can specify these formats:

"AUTU0100 Format" Each entry contains the user name, group names, an indicator that specifies whether the user is a user profile or a group profile, and an indicator that specifies whether the user is a group that has members.

"AUTU0200 Format" on page 237 Each entry contains the same information as AUTU0100 plus the text description for the user.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter

User Space Variables

The following tables describe the order and format of the data returned in the user space. For detailed descriptions of the fields in the tables, see “Field Descriptions” on page 237.

Input Parameter Section

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>CHAR(10)</td>
<td>User space name</td>
</tr>
<tr>
<td>10</td>
<td>0A</td>
<td>CHAR(10)</td>
<td>User space library name</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>CHAR(8)</td>
<td>Format name</td>
<td></td>
</tr>
</tbody>
</table>

AUTU0100 Format

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>CHAR(10)</td>
<td>User profile name</td>
</tr>
<tr>
<td>10</td>
<td>0A</td>
<td>CHAR(10)</td>
<td>Group name</td>
<td></td>
</tr>
</tbody>
</table>
### AUTU0200 Format

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0A</td>
<td>CHAR(10)</td>
<td>User profile name</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0A</td>
<td>CHAR(10)</td>
<td>Group name</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>14</td>
<td>CHAR(50)</td>
<td>Text name</td>
</tr>
<tr>
<td>70</td>
<td>70</td>
<td>46</td>
<td>CHAR(2)</td>
<td>Reserved</td>
</tr>
<tr>
<td>72</td>
<td>72</td>
<td>48</td>
<td>BINARY(4)</td>
<td>Number of supplemental groups</td>
</tr>
<tr>
<td>76</td>
<td>76</td>
<td>4C</td>
<td>ARRAY(15) OF CHAR(10)</td>
<td>Supplemental groups</td>
</tr>
<tr>
<td>226</td>
<td>226</td>
<td>E2</td>
<td>CHAR(1)</td>
<td>User or group indicator</td>
</tr>
<tr>
<td>227</td>
<td>227</td>
<td>E3</td>
<td>CHAR(1)</td>
<td>Group member indicator</td>
</tr>
</tbody>
</table>

### Field Descriptions

**Format name.** The name of the format used to list authorized users.

**Group member indicator.** Whether this user is a group that has members.

Possible values follow:

0 The user is not a group, or is a group but does not have any members. This value is returned if the user or group indicator field is 0.
1 The user is a group that has members.

**Group name.** The name of the user’s group profile. If the user does not have a group profile, this field contains *NONE.*

**Number of supplemental groups.** The number of supplemental groups returned in the supplemental groups field. The number of supplemental groups will be zero if the user does not have any supplemental groups.

**Reserved.** An ignored field.

**Supplemental groups.** The array of supplemental groups for the user profile. The number of supplemental groups field will indicate how many entries there are in the array.

**Text name.** The text description for the authorized user.

**User profile name.** The name of the authorized user.
User space name. The name of the user space used to return the list of authorized users on the system.

User space library name. The name of the library containing the user space.

User or group indicator. Whether this user is a user profile or a group profile.

Possible values follow:

0 User profile (profile does not have a GID)
1 Group profile (profile has a GID)

Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF2225 E</td>
<td>Not able to allocate internal system object.</td>
</tr>
<tr>
<td>CPF3CAA E</td>
<td>List is too large for user space &amp;1.</td>
</tr>
<tr>
<td>CPF3CF1 E</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>CPF3C21 E</td>
<td>Format name &amp;1 is not valid.</td>
</tr>
<tr>
<td>CPF3C90 E</td>
<td>Literal value cannot be changed.</td>
</tr>
<tr>
<td>CPF9801 E</td>
<td>Object &amp;2 in library &amp;3 not found.</td>
</tr>
<tr>
<td>CPF9802 E</td>
<td>Not authorized to object &amp;2 in &amp;3.</td>
</tr>
<tr>
<td>CPF9803 E</td>
<td>Cannot allocate object &amp;2 in library &amp;3.</td>
</tr>
<tr>
<td>CPF9807 E</td>
<td>One or more libraries in library list deleted.</td>
</tr>
<tr>
<td>CPF9808 E</td>
<td>Cannot allocate one or more libraries on library list.</td>
</tr>
<tr>
<td>CPF9810 E</td>
<td>Library &amp;1 not found.</td>
</tr>
<tr>
<td>CPF9820 E</td>
<td>Not authorized to use library &amp;1.</td>
</tr>
<tr>
<td>CPF9830 E</td>
<td>Cannot assign library &amp;1.</td>
</tr>
<tr>
<td>CPF9838 E</td>
<td>User profile storage limit exceeded.</td>
</tr>
<tr>
<td>CPF9872 E</td>
<td>Program or service program &amp;1 in library &amp;2 ended. Reason code &amp;3.</td>
</tr>
</tbody>
</table>

API introduced: V4R4

Change Directory Server Attributes (QgldChgDirSvrA)

Required Parameter Group:

- 1 Input data Input Char(*)
- 2 Length of input data Input Binary(4)
- 3 Format name Input Char(8)
- 4 Error code I/O Char(*)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDUAPI
Threadsafe: No

The Change Directory Server Attributes (QgldChgDirSvrA) API changes the directory server configuration. It can be used to change the following server properties:

- General server properties
- Suffixes served by this server
- Encrypted connection configuration. The Secure Sockets Layer (SSL) is used for encrypted communication.
- Performance settings
Authorities and Locks

*ALLOBJ and *IOSYSCFG special authority is required to use this API with formats CSVR0100, CSVR0200, CSVR0300, CSVR0400, CSVR0500, CSVR0600, CSVR0800, or CSVR0900. *AUDIT special authority is required to use this API with format CSVR0700.

Required Parameter Group

Input data

```plaintext
INPUT; CHAR(*)
```

A variable that contains the input data. See "Format of Input Data" for a description of the data associated with a specific format name.

Length of input data

```plaintext
INPUT; BINARY(4)
```

The length of the input data area.

Format name

```plaintext
INPUT; CHAR(8)
```

The format name identifying the type of information to be changed. The possible format names follow:

- **CSVR0100**  Basic server configuration
- **CSVR0200**  Add or remove suffixes from this server
- **CSVR0300**  Add, change, or remove directory indexing rules
- **CSVR0400**  Add or change the attributes for publishing users in an LDAP directory.
- **CSVR0500**  Add or change the network server publishing attributes associated with the LDAP server.
- **CSVR0600**  Add or change referral server information
- **CSVR0700**  Server auditing information
- **CSVR0800**  IP address information
- **CSVR0900**  Server administration information

See "Format of Input Data" for a description of these formats.

Error code

```plaintext
I/O; CHAR(*)
```

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Format of Input Data

For details about the format of the input data, see the following sections. For details about the fields in each format, see "Field Descriptions" on page 245.

CSVR0100 Format

This format is used to change basic server configuration information.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Read only</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Server is replica</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>Security</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>Nonencrypted port number</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>Encrypted port number</td>
</tr>
<tr>
<td>Offset</td>
<td>Dec</td>
<td>Hex</td>
</tr>
<tr>
<td>--------</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>32</td>
<td>20</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>36</td>
<td>24</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>40</td>
<td>28</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>44</td>
<td>2C</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>48</td>
<td>30</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>52</td>
<td>34</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>56</td>
<td>38</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>60</td>
<td>3C</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>64</td>
<td>40</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>68</td>
<td>48</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>72</td>
<td>48</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>76</td>
<td>4C</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>80</td>
<td>50</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>84</td>
<td>54</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>88</td>
<td>58</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>92</td>
<td>5C</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>96</td>
<td>60</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>100</td>
<td>64</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>104</td>
<td>64</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>108</td>
<td>68</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>112</td>
<td>70</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>116</td>
<td>74</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>120</td>
<td>78</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>124</td>
<td>7C</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>128</td>
<td>80</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>132</td>
<td>84</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>136</td>
<td>88</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>140</td>
<td>8C</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>144</td>
<td>90</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>148</td>
<td>94</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>152</td>
<td>98</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>156</td>
<td>9C</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>160</td>
<td>A0</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>164</td>
<td>A4</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>168</td>
<td>A8</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>Offset</td>
<td>Dec</td>
<td>Hex</td>
</tr>
<tr>
<td>--------</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>172</td>
<td>172</td>
<td>AC</td>
</tr>
<tr>
<td>176</td>
<td>176</td>
<td>B0</td>
</tr>
<tr>
<td>180</td>
<td>180</td>
<td>B4</td>
</tr>
<tr>
<td>184</td>
<td>184</td>
<td>B8</td>
</tr>
<tr>
<td>188</td>
<td>188</td>
<td>BC</td>
</tr>
<tr>
<td>192</td>
<td>192</td>
<td>C0</td>
</tr>
<tr>
<td>196</td>
<td>196</td>
<td>C4</td>
</tr>
<tr>
<td>200</td>
<td>200</td>
<td>C8</td>
</tr>
<tr>
<td>204</td>
<td>204</td>
<td>CC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>208</td>
<td>208</td>
<td>D0</td>
</tr>
<tr>
<td>212</td>
<td>212</td>
<td>D4</td>
</tr>
<tr>
<td>216</td>
<td>216</td>
<td>D8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>220</td>
<td>220</td>
<td>DC</td>
</tr>
<tr>
<td>224</td>
<td>224</td>
<td>E0</td>
</tr>
<tr>
<td>228</td>
<td>228</td>
<td>E4</td>
</tr>
<tr>
<td>232</td>
<td>232</td>
<td>E8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**CSVR0200 Format**

This format is used to add or remove suffixes from the server. The input data consists of a header and a series of change entries. The header identifies the number of suffixes to be added or removed. Each change entry identifies a suffix and the action to be performed (add or remove the suffix).

**Note:** Removing a suffix from a server will result in the loss of all directory entries with that suffix.
CSVR0300 Format
This format is used to add, change, or remove directory indexes. Creating indexes for one or more attributes allows for faster retrieval of directory entries based on those attributes. The input data consists of a header and a series of change entries. The header identifies the number of indexes to be added, changed, or removed. Each change entry identifies an attribute and the action to be performed (add, change, or remove the indexes).

Starting with V4R5M0, this format is not supported. Database index information is to be changed using an LDAP client or the Directory Management Tool (DMT) starting with V4R5M0.
CSVR0400 Format

This format is used to set the attributes for publishing users in an LDAP directory. User information from the System Distribution Directory (SDD) can be published to an LDAP server by the Synchronize System Distribution Directory to LDAP (QGLDSSDD) API and from iSeries Navigator. The publishing attributes define how to publish user information.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>BINARY(4)</td>
<td>Offset to the server name</td>
</tr>
<tr>
<td>4</td>
<td>BINARY(4)</td>
<td>Length of server name</td>
</tr>
<tr>
<td>8</td>
<td>BINARY(4)</td>
<td>LDAP port number</td>
</tr>
<tr>
<td>12</td>
<td>BINARY(4)</td>
<td>Connection type</td>
</tr>
<tr>
<td>16</td>
<td>BINARY(4)</td>
<td>Offset to parent distinguished name</td>
</tr>
<tr>
<td>20</td>
<td>BINARY(4)</td>
<td>Length of parent distinguished name</td>
</tr>
<tr>
<td>24</td>
<td>BINARY(4)</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Variable length string fields:
- CHAR(*) Server name
- CHAR(*) Parent distinguished name

CSVR0500 Format

This format is used to set the network server publishing attributes associated with the server.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>BINARY(4)</td>
<td>Offset to change entries</td>
</tr>
<tr>
<td>4</td>
<td>BINARY(4)</td>
<td>Number of change entries</td>
</tr>
</tbody>
</table>

Add or change publishing agent change entries:
- Displacement to next entry
- Action
- Displacement to publishing agent name
- Length of publishing agent name
- Displacement to server name
- Length of server name
- Displacement to bind DN
- Length of bind DN
- Displacement to bind credentials
- Length of bind credentials
- LDAP port number
- Connection type
- Displacement to parent distinguished name
- Length of parent distinguished name
- Disable publishing agent
### CSVR0600 Format

This format is used to change referral server configuration information. The input data consists of a header and a series of change entries. The header identifies the master server information and the number of referral servers. This replaces the referral server information, if any, that is currently configured.

#### Offset Table

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>BINARY(4) Offset to change entries</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>BINARY(4) Number of change entries</td>
</tr>
</tbody>
</table>

Change entries:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>BINARY(4) Displacement to next entry</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>BINARY(4) Displacement to referral server URL</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>BINARY(4) Length of referral server URL</td>
</tr>
</tbody>
</table>
**CSVR0700 Format**
This format is used to change the server auditing configuration information.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td>CHAR(*)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>BINARY(4)</td>
</tr>
</tbody>
</table>

**CSVR0800 Format**
This format is used to change the IP address configuration information. The input data consists of a header and a series of change entries. The header identifies the number of IP addresses in the list. This replaces the IP address information that is currently configured. At least one IP address value must be specified for the server.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td>Type</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

IP address entries:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHAR(*)</td>
</tr>
</tbody>
</table>

**CSVR0900 Format**
This format is used to change the server administration information.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td>Type</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHAR(*)</td>
</tr>
</tbody>
</table>

**Field Descriptions**

ACL model. Indicator of the ACL model to use. The following special values may be specified:
-1 The value of this field does not change.
1 Use the ACL model that supports attribute-level ACL permissions. This may cause compatibility problems with replication and applications that manage access-class level permissions defined in releases prior to V5R1M0. Once enabled, this capability can be disabled only by reconfiguring your server and deleting the directory database.

Note: Starting with V5R3M0, this field is ignored for format CSVR0100.

**Action.** The action to be performed for a given entry. The following values may be specified:

1 Add suffix, index rule, or publishing agent
2 Change index rule or publishing agent
3 Remove suffix, index rule, or publishing agent

Note: Change is valid only for the CSVR0300 and CSVR0500 formats.

**Administrator DN.** A distinguished name that has access to all objects in the directory. When either the administrator DN or the administrator password field is changed, both must be specified. This field is specified in UTF-16 (CCSID 13488). To leave the value unchanged, specify a length and offset to this field of zero.

**Administrator password.** The password used when connecting to the directory server using the administrator DN. When either the administrator DN or the administrator password field is changed, both must be specified. This field is specified in UTF-16 (CCSID 13488). To leave the value unchanged, specify a length and offset to this field of zero.

**Attribute index entries.** The list of changes to be made to the attribute indexes.

**Attribute name.** The name of a directory object attribute for which database indexes will be created. This field is specified in UTF-16 (CCSID 13488). The following special value may be specified:

*DEFAULT Specifies the index types to be created for those attributes that have no explicit rules defined.

Note: The *DEFAULT attribute entry may be removed or added. Adding or removing *DEFAULT attribute is equivalent to not creating any indexes, or creating indexes for all attributes, depending on the index types specified.

**Bind credentials.** The password used when connecting to the directory server using the bind DN. When either the bind DN or the bind credentials field is changed, both must be specified. This field is specified in UTF-16 (CCSID 13488). To leave the value unchanged, specify a length and displacement to this field of zero.

**Bind DN.** A distinguished name to use when publishing objects to the directory. When either the bind DN or the bind credentials field is changed, both must be specified. This field is specified in UTF-16 (CCSID 13488). To leave the value unchanged, specify a length and displacement to this field of zero.

**Change entry.** A structure identifying a change to be made. The structure identifies the suffix, attribute, or publishing agent and the operation to be performed (add, change, or delete).

**Change log indicator.** The indicator of whether to have a change log for entries that are added, changed or deleted. The following values may be specified:

0 No, do not have a change log
1 Yes, have a change log
Connection type. The type of connection to use to the LDAP server. The following values may be specified:

1  Nonsecure
2  Secured, using SSL
-1  The value remains the same

Current cipher protocols. The cipher protocols that the server will allow when using encrypted connections. The following values may be specified:

-1  The value remains the same

Or the sum of one or more of the following values:

0x0100  Triple Data Encryption Standard (DES) Secure Hash Algorithm (SHA) (U.S.)
0x0200  DES SHA (U.S.)
0x0400  Rivest Cipher 4 (RC4) SHA (U.S.)
0x0800  RC4 Message Digest 5 (MD5) (U.S.)
0x1000  RC2 MD5 (export)
0x2000  RC4 MD5 (export)
0x4000  Advanced Encryption Standard (AES) SHA 128 bit (U.S.)
0x8000  Advanced Encryption Standard (AES) SHA 256 bit (U.S.)

Database path. The path to an existing library containing the directory database objects. This is an integrated file system path name, for example, /QSYS.LIB/DIRSRV.LIB. By changing this field, you make the current directory contents inaccessible. By changing the field back to its original value, you restore the original directory contents. The library must exist in a system ASP or a basic user ASP (ASP value of 1 to 32). The library cannot exist in an independent ASP (ASP value greater than 32). This field is specified in UTF-16 (CCSID 13488). To leave the value unchanged, specify a length and offset to this field of zero.

Disable publishing agent. Indicates whether or not the publishing agent is disabled. The following values may be specified:

0  The publishing agent is enabled.
1  The publishing agent is disabled.

Displacement to attribute name. The displacement, in bytes, from the start of the current entry to the attribute name field.

Displacement to bind credentials. The displacement, in bytes, from the start of the current entry to the bind credentials field.

Displacement to bind DN. The displacement, in bytes, from the start of the current entry to the bind DN field.

Displacement to IP address. The displacement, in bytes, from the start of the current entry to the IP address field.

Displacement to Kerberos key tab file. The displacement, in bytes, from the start of the current entry to the Kerberos key tab file field.
**Displacement to Kerberos principal.** The displacement, in bytes, from the start of the current entry to the Kerberos principal field.

**Displacement to Kerberos realm.** The displacement, in bytes, from the start of the current entry to the Kerberos realm field.

**Displacement to next entry.** The displacement, in bytes, from the start of the current entry to the next entry in the input data.

**Displacement to parent distinguished name.** The displacement, in bytes, from the start of the current entry to the parent distinguished name field.

**Displacement to publishing agent name.** The displacement, in bytes, from the start of the current entry to the publishing agent name field.

**Displacement to referral server URL.** The displacement, in bytes, from the start of the current entry to the referral server URL field.

**Displacement to server name.** The displacement, in bytes, from the start of the current entry to the server name field.

**Displacement to suffix.** The displacement, in bytes, from the start of the current entry to the suffix field.

**Encrypted port number.** The port number to use for encrypted connections. The standard port number for encrypted connections (SSL) is 636. Valid port numbers are in the range 1 to 65535. The following special value may be specified:

-1  The value of this field does not change.

**Event notification registration indicator.** Indicator of whether to allow client to register for event notification. The following special values may be specified:

-1  The value of this field does not change.
0   Do not allow clients to register for event notification.
1   Allow clients to register for event notification.

**Index type.** The kind of database indexes that will be created for an attribute. Creating database indexes improved the performance of directory searches on those attributes. The following values may be specified:

0   No indexes will be maintained for the specified attribute
1   Equal

**Note:** For a delete request, 0 must be specified for this field.

**IP address.** The IPv4 or IPv6 address of the client for which the directory server will accept connections. The IP address must already exist to be specified. A value of hexadecimal zeroes and leading zeroes is not allowed. An IPv4 address is expressed in standard dotted-decimal form `www.xxx.yyy.zzz`; for example, `130.99.128.1`. An IPv6 address always has at least one occurrence of a colon (`:`) in the format. Some possible IPv6 address formats would be `::x` (for example, `::1`) or `::w.xxx.y.zzz` (for example, `::9.130.4.169`). For further IPv6 examples and explanation, refer to the Usage Notes section in the Convert IPv4 and IPv6 Addresses Between Text and Binary Form (inet_pton) API. This field is specified in UTF-16 (CCSID 13488).
The following special value may be specified:

*ALL All IP addresses defined on the local system will be bound to the server.

Kerberos administrator ID. The name of the Kerberos administrator. This field is specified in UTF-16 (CCSID 13488). The following special value may be specified:

*NONE No value is specified.

To leave the value unchanged, specify a length and offset to this field of zero.

Kerberos administrator realm. The realm where the kerberos administrator is registered. This field is specified in UTF-16 (CCSID 13488). The following special value may be specified:

*NONE No value is specified.

To leave the value unchanged, specify a length and offset to this field of zero.

Kerberos authentication indicator. The following special values may be specified:

-1 The value of this field does not change.
0 Do not support Kerberos authentications.
1 Support Kerberos authentications. Ensure all Kerberos fields are specified.

Kerberos key tab file. The integrated file system path name for the key tab file that contains the server’s secret key used for authentication. The QDIRSRV user profile is given authorization to read this file. This field is specified in UTF-16 (CCSID 13488). The following special value may be specified:

*NONE No value is specified.

To leave the value unchanged, specify a length and offset or displacement to this field of zero.

Kerberos principal. The principal in the key tab file to use for authentication. This field is specified in UTF-16 (CCSID 13488). The following special value may be specified:

*NONE No value is specified.

To leave the value unchanged, specify a length and offset or displacement to this field of zero.

Kerberos realm. The realm where the principal is registered to use for authentication. This field is specified in UTF-16 (CCSID 13488). The following special value may be specified:

*NONE No value is specified.

To leave the value unchanged, specify a length and offset or displacement to this field of zero.

Kerberos to DN mapping indicator.

-1 The value of this field does not change.
0 Map the Kerberos ID to pseudo DN. A pseudo DN can be used to uniquely identify an LDAP user object of the form ‘ibm-kerberosName=principal@realm’ or ‘ibm-kn=principal@realm’.
Use associated DN in directory. The LDAP server will attempt to find an entry in the directory that contains the kerberos principle and realm as one of its attributes. Once found, this DN will then be used to determine the client’s authorizations to the directory.

**Key ring file.** The path name of the SSL key ring file. A key ring file must be configured when using SSL. The following special value may be specified:

*NONE No value is specified.

**Note:** Starting with V4R4M0, this field is ignored for format CSVR0100. This field is specified in UTF-16 (CCSID 13488).

To leave the value unchanged, specify a length and offset to this field of zero.

**LDAP port number.** The LDAP server’s TCP/IP port. The following values may be specified:

-1 The value remains the same

**Length of administrator DN.** The length, in UTF-16 (CCSID 13488) characters, of the administrator DN field.

**Length of administrator password.** The length, in UTF-16 (CCSID 13488) characters, of the administrator password field.

**Length of attribute name.** The length, in UTF-16 (CCSID 13488) characters, of the the attribute name field.

**Length of bind credentials.** The length, in UTF-16 (CCSID 13488) characters, of the bind credentials field.

**Length of bind DN.** The length, in UTF-16 (CCSID 13488) characters, of the bind DN field.

**Length of database path.** The length, in UTF-16 (CCSID 13488) characters, of the database path field.

**Length of IP address.** The length, in UTF-16 (CCSID 13488) characters, of the IP address field.

**Length of Kerberos administrator ID.** The length, in UTF-16 (CCSID 13488) characters, of the Kerberos administrator ID field.

**Length of Kerberos administrator realm.** The length, in UTF-16 (CCSID 13488) characters, of the Kerberos administrator realm field.

**Length of Kerberos key tab file.** The length, in UTF-16 (CCSID 13488) characters, of the Kerberos key tab file field.

**Length of Kerberos principal.** The length, in UTF-16 (CCSID 13488) characters, of the Kerberos principal field.

**Length of Kerberos realm.** The length, in UTF-16 (CCSID 13488) characters, of the Kerberos realm field.

**Length of key ring file.** The length, in UTF-16 (CCSID 13488) characters, of the key ring file field.

**Length of master server URL.** The length, in UTF-16 (CCSID 13488) characters, of the master server URL field.
**Length of parent distinguished name.** The length, in UTF-16 (CCSID 13488) characters, of the parent distinguished name field.

**Length of projected suffix.** The length, in UTF-16 (CCSID 13488) characters, of the projected suffix field.

**Length of publishing agent name.** The length, in UTF-16 (CCSID 13488) characters, of the publishing agent name. The length can be at most 50 characters.

**Length of referral server.** The length, in UTF-16 (CCSID 13488) characters, of the referral server name.

**Length of referral server URL.** The length, in UTF-16 (CCSID 13488) characters, of the referral server URL field.

**Length of server administration URL.** The length, in UTF-16 (CCSID 13488) characters, of the server administration URL field.

**Length of server name.** The length, in UTF-16 (CCSID 13488) characters, of the server name field.

**Length of suffix.** The length, in UTF-16 (CCSID 13488) characters, of the suffix field.

**Length of update DN.** The length, in UTF-16 (CCSID 13488) characters, of the update DN field.

**Length of update password.** The length, in UTF-16 (CCSID 13488) characters, of the update password field.

**Level indicator.** The level indicator of the data supplied for a format. See the format descriptions for possible uses and values of this field.

**Level of authority integration.** The level of i5/OS authority integration to use to determine if a distinguished name (DN) can become an LDAP administrator. Allowing a user profile to become an LDAP administrator can be done by setting the 'Level of authority integration' to '1' and then authorizing specific user profiles to the 'Directory Server Administrator' function of the operating system through iSeries Navigator’s Application support. The Change Function Usage Information (QSYCHFUI) API, with a function ID of QIBM_DIRSRV_ADMIN, can also be used to change the list of users that are allowed to be an LDAP administrator. The user profile can be mapped to a DN as a projected user (for example, for user profile 'FRED', and the projected suffix of 'systemA', the projected user’s DN would be os400-profile=FRED,cn=accounts,os400-sys=systemA).

The following special values may be specified:

- `-1` The value of this field does not change.
- `0` Do not apply 'Directory Server Administrator' function identifier to bound distinguished names to determine LDAP administrators.
- `1` Allow bound distinguished names that refer directly to user profiles to become LDAP administrators if the user profile is identified in the 'Directory Server Administrator' function identifier.

**Log client messages.** Whether the directory server will log client messages in the server joblog. The following values may be specified:

- `-1` The value of this field does not change.
- `0` The directory server will not log client messages in the server joblog.
- `1` The directory server will log client messages in the server joblog.

**Master server URL.** The uniform resource locator (URL) of the master server. This field is specified in UTF-16 (CCSID 13488). The following special value may be specified:
*NONE*  
No value is specified.

To leave the value unchanged, specify a length and offset to this field of zero.

**Maximum connections.** The maximum number of simultaneous connections that can be established with the server. The following special values may be specified:

-1 The value of this field does not change.
0 Do not limit the number of connections.

**Note:** Starting with V5R1M0, this field is no longer supported and is ignored if a value is passed.

**Maximum event registrations for connection.** The following special values may be specified:

-1 The value of this field does not change.
0 Do not limit the number of event registrations for connection.

**Maximum event registrations for server.** The following special values may be specified:

-1 The value of this field does not change.
0 Do not limit the number of event registrations for server.

**Maximum age of change log entries.** The maximum age, in seconds, of change log entries that can be stored. If the maximum is reached, the change log entries will be deleted starting with the oldest entry. This value only used if 'Change log indicator' is set to 1. The following special values may be specified:

-1 The value of this field does not change.
0 Do not limit the age of change log entries.

**Maximum number of change log entries.** The maximum number of change log entries that can be stored. If the maximum is reached, the change log entries will be deleted starting with the oldest entry. This value only used if 'Change log indicator' is set to 1. The following special values may be specified:

-1 The value of this field does not change.
0 Do not limit the number of change log entries.

**Maximum operations per transaction.** The maximum number of operations that are allowed for each transaction. Transaction support allows a group of directory changes to be handled as a single transaction. The following special values may be specified:

-1 The value of this field does not change.

**Maximum pending transactions.** The maximum number of pending transactions allowed. Transaction support allows a group of directory changes to be handled as a single transaction. The following special value may be specified:

-1 The value of this field does not change.

**Nonencrypted port number.** The port number to be used for nonencrypted connections. The standard port number is 389. Valid port numbers are in the range 1 to 65535. The following special value may be specified:
-1 The value of this field does not change.

**Number of change entries.** The number of change entries present in the input data.

**Number of database connections.** The number of database connections used by the server. Valid numbers are in the range 4 to 32. The following special value may be specified:

-1 The value of this field does not change.

**Offset to administrator DN.** The offset, in bytes, from the start of the input data area to the administrator DN field.

**Offset to administrator password.** The offset, in bytes, from the start of the input data area to the administrator password field.

**Offset to change entries.** The offset, in bytes, from the start of the input data area to the first change entry.

**Offset to database path.** The offset, in bytes, from the start of the input data area to the database path field.

**Offset to Kerberos administrator ID.** The offset, in bytes, from the start of the input data area to the Kerberos administrator ID field.

**Offset to Kerberos administrator realm.** The offset, in bytes, from the start of the input data area to the Kerberos administrator realm field.

**Offset to Kerberos key tab file.** The offset, in bytes, from the start of the input data area to the Kerberos key tab file field.

**Offset to key ring file.** The offset, in bytes, from the start of the input data area to the key ring file field.

**Offset to master server URL.** The offset, in bytes, from the start of the input data area to the master server URL field.

**Offset to parent distinguished name.** The offset, in bytes, from the start of the input data area to the parent distinguished name field.

**Offset to projected suffix.** The offset, in bytes, from the start of the input data area to the projected suffix field.

**Offset to referral server.** The offset, in bytes, from the start of the input data area to the referral server field.

**Offset to server administration URL.** The offset, in bytes, from the start of the input data area to the server administration URL field.

**Offset to server name.** The offset, in bytes, from the start of the input data area to the server name field.

**Offset to suffix.** The offset, in bytes, from the start of the input data area to the suffix field.

**Offset to update DN.** The offset, in bytes, from the start of the input data area to the update DN field.
**Offset to update password.** The offset, in bytes, from the start of the input data area to the update password field.

**Parent distinguished name.** The parent distinguished name for published objects. For example, if the parent distinguished name is "ou=rochester, o=ibm, c=us", a published directory object for user John Smith might be "cn=john smith, ou=rochester, o=ibm, c=us". This field is specified in UTF-16 (CCSID 13488). To leave the value unchanged, specify a length and offset to this field of zero.

**Password format.** The format of the encrypted password. The following values may be specified:

-1 The value of this field does not change.
1 Unencrypted. The clear text password is stored in a validation list and can be returned by searches or used for DIGEST-MD5 SASL authentication.
2 SHA. (Default)
3 MD5.
4 Crypt (The password is one-way hashed using a modified DES algorithm. The "crypt" algorithm originally was used by many Unix operating systems for password protection.)

**Projected suffix.** The suffix under which all projected objects for this server reside including user and group profiles. This field is specified in UTF-16 (CCSID 13488).

**Publishing agent name.** The agent that will publish information to a directory server and parent distinguished name. This field is specified in UTF-16 (CCSID 13488).

The following publishing agent names are predefined by the operating system:

*AS400_COMPUTERS* This agent name is used for publishing system information such as the system and printers.
*AS400_PRINTSHARES* This agent name is used for publishing print shares to an Active Directory server.
*AS400_USERS* This agent name is used for publishing System Distribution Directory users.
*OS400_TC1_QOS* This agent name is used for publishing TCP/IP Quality of Service policy information.

**Read only.** Whether the directory server will allow updates to be made to the directory contents. The following values may be specified:

-1 The value of this field does not change.
0 Places the directory server into update mode to allow directory updates. This is the normal mode of operation.
1 Places the directory server into read-only mode.

**Read only projected suffix.** Whether the directory server will allow updates to be made to the projected suffix. The following values may be specified:

-1 The value of this field does not change.
0 Places the directory server projected suffix into update mode to allow updates. This is the normal mode of operation.
1 Places the directory server projected suffix into read-only mode.

**Read only schema.** Whether the directory server will allow updates to be made to the directory schema. The following values may be specified:

-1 The value of this field does not change.
0 Places the directory server schema into update mode to allow updates. This is the normal mode of operation.
1 Places the directory server schema into read-only mode.
Referral port. An optional port number to be returned to a client when a request is made for a directory object that does not reside on this server. The referral port and referral server together are used to form a referral URL. The referral server and port fields must be configured when changing the Server is replica field to make this server a replica. Valid port numbers are in the range 1 to 65535.

Starting with V4R5M0, this field is ignored for format CSVR0100. Referral server information can be changed using the CSVR0600 format of the QgIdChgDirSvrA API. The following special values may be specified:

0 No port number is returned as part of the referral.
-1 The value of this field does not change.

Referral server. The IP name or address of a server to return to a client when a request is made for a directory object that does not reside on this server. The referral port and referral server are used together to form a referral URL. The referral server and port fields must be configured when changing the Server is replica field to make this server a replica. In this case, the referral is typically to the master server. The following special value may be specified:

*NONE No value is specified.

Note: Starting with V4R5M0, this field is ignored for format CSVR0100. This field is specified in UTF-16 (CCSID 13488). To leave the value unchanged, specify a length and offset to this field of zero.

Referral server URL. The uniform resource locator (URL) of the referral server. This field is specified in UTF-16 (CCSID 13488).

Reserved. A reserved field. This field must be set to zero.

Schema checking level. The level of schema checking performed by the server. The following values may be specified:

-1 The value does not change.
0 None.
1 LDAP version 2.
2 LDAP version 3 strict.
3 LDAP version 3 lenient.

Search size limit. The maximum number of entries that the server will return for a given search request. The following special values may be specified:

-1 The value of this field does not change.
0 Do not limit the number of entries returned.

Search time limit. The maximum time, in seconds, that the server will spend performing a given search request. The following special values may be specified:

-1 The value of this field does not change.
0 Do not limit the search time.

Security. Whether the server should use encrypted connections. The following values may be specified:
-1 The value does not change
1 Allow nonencrypted connections only
2 Allow encrypted connections only
3 Allow both encrypted and nonencrypted connections

Security audit option for objects. When the QAUDCTL system value is set to OBJAUD, then object auditing can be done in the directory. See the iSeries Security Reference book for information about Directory Server auditing. The following special values may be specified:

-1 The value of this field does not change.
0 Do not do object auditing of the directory objects.
1 Audit changes to directory objects.
2 Audit all access to directory objects. This includes search, compare and change.

Server is replica. Whether the server is a master server or a replica server. When this field is changed to make the server a replica, the update DN, update password, and referral fields must be specified. The following values may be specified:

-1 The value of this field does not change.
0 The server is a master for the directory suffixes present on the server.
1 The server is a replica server for the directory suffixes present on the server.

Server administration URL. The server administration URL. This field is specified in UTF-16 (CCSID 13488). To leave the value unchanged, specify a length and offset to this field of zero.

Server name. The name of the server. This field is specified in UTF-16 (CCSID 13488). To leave the value unchanged, specify a length and offset to this field of zero.

SSL authentication method. The method used during SSL authentication. The following values may be specified:

-1 The value does not change.
1 Server authentication.
3 Server and client authentication.

Suffix. The name of the directory suffix to be added or removed from the server. This field is specified in UTF-16 (CCSID 13488).

Suffix change entries. The list of suffixes to be added or deleted.

Terminate idle connections. The server will terminate idle connections when necessary.

Starting with V5R1M0, this field is no longer supported and is ignored if a value is passed. The following values may be specified:

0 Do not terminate idle connections.
1 Terminate idle connections.

Transaction time limit. The maximum time, in seconds, that the server will spend performing a transaction request. Transaction support allows a group of directory changes to be handled as a single transaction. The following special values may be specified:
The value of this field does not change.

**Update DN.** The distinguished name that the master server must use when propagating directory updates to this replica server. This field may be specified only when the server is a replica. When either the update DN or the update password field is changed, both must be specified. This field is specified in UTF-16 (CCSID 13488). The following special value may be specified:

*NONE  
No value is specified.

To leave the value unchanged, specify a length and offset to this field of zero.

**Update password.** The password used when connecting to this server using the update DN. This field may be specified only when the server is a replica. When either the update DN or the update password field is changed, both must be specified. This field is specified in UTF-16 (CCSID 13488). To leave the value unchanged, specify a length and offset to this field of zero. The following special value may be specified:

*NONE  
No value is specified.

**Error Messages**

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF209 E</td>
<td>Library &amp;l not found.</td>
</tr>
<tr>
<td>CPFA0A9 E</td>
<td>Object not found.</td>
</tr>
<tr>
<td>CPFA0DB E</td>
<td>Object name not a QSYS object.</td>
</tr>
<tr>
<td>CPFA314 E</td>
<td>Memory allocation error.</td>
</tr>
<tr>
<td>GLD0204 E</td>
<td>Attribute name not valid.</td>
</tr>
<tr>
<td>GLD0205 E</td>
<td>Administrator DN not valid.</td>
</tr>
<tr>
<td>GLD0208 E</td>
<td>Key ring file name not valid.</td>
</tr>
<tr>
<td>GLD0209 E</td>
<td>Update DN not valid.</td>
</tr>
<tr>
<td>GLD020A E</td>
<td>Suffix not valid.</td>
</tr>
<tr>
<td>GLD020B E</td>
<td>Referral server name not valid.</td>
</tr>
<tr>
<td>GLD020D E</td>
<td>Index rule already defined for attribute.</td>
</tr>
<tr>
<td>GLD020E E</td>
<td>Index rule not found for attribute.</td>
</tr>
<tr>
<td>GLD0211 E</td>
<td>Value &amp;l specified at offset &amp;2 in input format &amp;3 is not valid.</td>
</tr>
<tr>
<td>GLD0212 E</td>
<td>Field &amp;l required when server is using SSL.</td>
</tr>
<tr>
<td>GLD0215 E</td>
<td>IBM Directory Server for iSeries server has not been configured.</td>
</tr>
<tr>
<td>GLD0217 E</td>
<td>A value was specified in list entry &amp;l that is not valid. Reason code &amp;2.</td>
</tr>
<tr>
<td>GLD0219 E</td>
<td>Administrator DN and password both required.</td>
</tr>
<tr>
<td>GLD021A E</td>
<td>Field not allowed when server is not a replica.</td>
</tr>
<tr>
<td>GLD021B E</td>
<td>Field is required when server is a replica.</td>
</tr>
<tr>
<td>GLD021C E</td>
<td>The caller of the API must have *ALLOBJ and *IOSYSCFG special authority to configure the server.</td>
</tr>
<tr>
<td>GLD021D E</td>
<td>Error occurred when processing the input list of entries.</td>
</tr>
<tr>
<td>GLD021E E</td>
<td>&amp;l password is not valid.</td>
</tr>
<tr>
<td>GLD021F E</td>
<td>The caller of the API must have *AUDIT special authority to set the server auditing information.</td>
</tr>
<tr>
<td>GLD0221 E</td>
<td>Offset &amp;l specified in input data is not valid.</td>
</tr>
<tr>
<td>GLD0222 E</td>
<td>Length &amp;l specified in input data is not valid.</td>
</tr>
<tr>
<td>GLD0223 E</td>
<td>Database path not valid.</td>
</tr>
<tr>
<td>GLD0227 E</td>
<td>Distinguished names cannot be modified while the server is active.</td>
</tr>
<tr>
<td>GLD0229 E</td>
<td>Validation list not found.</td>
</tr>
<tr>
<td>GLD022D E</td>
<td>Publishing &amp;l agent not found.</td>
</tr>
</tbody>
</table>
Export LDIF File (QgldExportLdif)

Required Parameter Group:

1. Input data
2. Length of input data
3. Format name
4. Error code

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDUAPI
Threading: No


 Authorities and Locks

 Directory Authority

The caller must provide the administrator DN and password if the caller does not have *ALLOBJ and *IOSYS_ECFG special authorities and the caller is not a Directory Services administrator. The caller is a Directory Services administrator if the Directory Services server has been configured to grant administrator access to authorized users and the caller is authorized to the ‘Directory Services Administrator’ function of the operating system.

 Object Authorities

The caller must have Execute (*X) authority to each directory in the path name preceding the name of the LDIF file. The caller must have Write (*W) authority to the LDIF file.

Required Parameter Group

Input data

```
INPUT; CHAR(*)
```

Input data required to identify the LDIF file and the administrator name and password. The content and format of this structure are determined by the format name. See "Format of Input Data" on page 259 for a description of these formats.

Length of input data

```
INPUT; BINARY(4)
```

The length of the input data structure.

Format name

```
INPUT; CHAR(8)
```

API introduced: V4R3
The content and format of the input data. The possible format name follows:

- "LDIF0100 Format" Export LDIF file.
- "LDIF0200 Format" Export LDIF file with options.

See "Format of Input Data" for a description of this format.

Error code
I/O; CHAR(*)
The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Format of Input Data
For details about the format of the input data, see the following section. For details about the fields in each format, see "Field Descriptions" on page 260.

### LDIF0100 Format

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td>Dec</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td></td>
<td>CHAR(*)</td>
<td>LDIF file</td>
</tr>
<tr>
<td></td>
<td>CHAR(*)</td>
<td>Administrator DN</td>
</tr>
<tr>
<td></td>
<td>CHAR(*)</td>
<td>Administrator password</td>
</tr>
<tr>
<td></td>
<td>CHAR(*)</td>
<td>Subtree DN</td>
</tr>
</tbody>
</table>

### LDIF0200 Format

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td>Dec</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>Offset</td>
<td>Dec</td>
<td>Hex</td>
</tr>
<tr>
<td>--------</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>32</td>
<td>20</td>
<td>BINARY(4)</td>
</tr>
</tbody>
</table>

Additional fields if indicator is equal to 1 or greater:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>24</td>
<td>BINARY(4)</td>
<td>Include cn=localhost</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>28</td>
<td>BINARY(4)</td>
<td>Include cn=pwdpolicy</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>2C</td>
<td>BINARY(4)</td>
<td>Include nested replication contexts</td>
<td></td>
</tr>
</tbody>
</table>

Additional fields if indicator is equal to 2 or greater:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>30</td>
<td>BINARY(4)</td>
<td>Include operational attributes</td>
<td></td>
</tr>
</tbody>
</table>

Variable length string fields:

<table>
<thead>
<tr>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR(*)</td>
<td>LDIF file</td>
</tr>
<tr>
<td>CHAR(*)</td>
<td>Administrator DN</td>
</tr>
<tr>
<td>CHAR(*)</td>
<td>Administrator password</td>
</tr>
<tr>
<td>CHAR(*)</td>
<td>Subtree DN</td>
</tr>
</tbody>
</table>

### Field Descriptions

**Additional fields indicator.** Additional fields supplied for a format. See the format descriptions for possible uses and values of this field.

**Administrator DN.** The distinguished name of the server administrator. This field is specified in UCS-2 (CCSID 13488).

**Administrator password.** The password for the server administrator. This field is specified in UCS-2 (CCSID 13488).

**Include cn=localhost.** Indicates whether data located under the distinguished name cn=localhost should be included in the exported data. If format LDIF0100 is used, cn=localhost is not included in the exported data. The following values may be specified:

0  The contents of cn=localhost are not included in the exported data.
1  The contents of cn=localhost are included in the exported data.

The value 1 can only be specified when no subtree DN is specified.

**Include cn=pwdpolicy.** Indicates whether data located under the distinguished name cn=pwdpolicy should be included in the exported data. If format LDIF0100 is used, cn=pwdpolicy is not included in the exported data. The following values may be specified:

0  The contents of cn=pwdpolicy are not included in the exported data.
1  The contents of cn=pwdpolicy are included in the exported data.

The value 1 can only be specified when no subtree DN is specified.

**Include nested replication contexts.** Indicates whether nested replication contexts should be included in the exported data. For example, if a directory contains the replication contexts o=acme and cn=external
users, o=acme, this option can be used to export data under the distinguished name o=acme while excluding all entries under the distinguished name cn=external users, o=acme. If format LDIF0100 is used, nested replication contexts are included in the exported data. The following values may be specified:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Data from nested replication contexts is not included in the exported data.</td>
</tr>
<tr>
<td>1</td>
<td>Data from nested replication contexts is included in the exported data.</td>
</tr>
</tbody>
</table>

The value 0 can only be specified if a subtree DN is specified.

**Include operational attributes.** Indicates whether the following four operational attributes will be exported: creatorsName, createTimestamp, modifiersName, and modifyTimestamp. If no value is specified, these operational attributes will be included in the exported data. The following values may be specified:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The operational attributes are not included in the exported data.</td>
</tr>
<tr>
<td>1</td>
<td>The operational attributes are included in the exported data.</td>
</tr>
</tbody>
</table>

**LDIF file.** The integrated file system path name of the LDIF file to be used. This field is specified in UCS-2 (CCSID 13488).

**Length of administrator DN.** The length, in Unicode characters, of the administrator DN field.

**Length of administrator password.** The length, in Unicode characters, of the administrator password field.

**Length of LDIF file.** The length, in Unicode characters, of the LDIF file field.

**Length of subtree DN.** The length, in Unicode characters, of the subtree DN field.

**Offset to administrator DN.** The offset, in bytes, from the start of the input data to the administrator DN field.

**Offset to administrator password.** The offset, in bytes, from the start of the input data to the administrator password field.

**Offset to LDIF file.** The offset, in bytes, from the start of the input data to the LDIF file field.

**Offset to subtree DN.** The offset, in bytes, from the start of the input data to the subtree DN field.

**Subtree DN.** The distinguished name (DN) of the root of a directory subtree to export to the LDIF file. This object, and all descendant objects will be exported. To export the entire directory tree, specify 0 (zero) for the offset to subtree DN and length of subtree DN fields. This field is specified in UCS-2 (CCSID 13488).

**Error Messages**

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLD0202 E</td>
<td>Administrator DN or password not correct.</td>
</tr>
<tr>
<td>GLD0213 E</td>
<td>Error opening or creating file.</td>
</tr>
<tr>
<td>GLD0215 E</td>
<td>Server has not been configured.</td>
</tr>
<tr>
<td>GLD0218 E</td>
<td>*ALLOBJ and *IOSYSCFG special authorities required.</td>
</tr>
<tr>
<td>GLD022B E</td>
<td>Cannot find object &amp;1.</td>
</tr>
</tbody>
</table>

API Introduced: V4R3
Import LDIF File (QgldImportLdif)

Required Parameter Group:

1. Input data  
   Input Data: CHAR(*)

2. Length of input data  
   Input Length: Binary(4)

3. Format name  
   Input Format name: Char(8)

4. Error code  
   I/O Error code: Char(*)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDUAPI
Threadsafe: No


The IBM Directory Server must be stopped to use this API. To stop the server, use the End TCP/IP Server (ENDTCPSVR SVR(*DIRSRV)) command.

Authorities and Locks

Directory Authority

The caller must provide the administrator DN and password if the caller does not have *ALLOBJ and *IOSYSCFG special authorities and the caller is not a IBM Directory Server administrator. The caller is a IBM Directory Server administrator if the IBM Directory Server for iSeries has been configured to grant administrator access to authorized users and the caller is authorized to the 'Directory Server Administrator' function of the operating system.

Object Authorities

The caller must have Execute (*X) authority to each directory in the path name preceding the name of the LDIF file. The caller must have Read (*R) authority to the LDIF file.

Required Parameter Group

Input data

Input Data: CHAR(*)

Input data required to identify the LDIF file and the administrator name and password. The content and format of this structure are determined by the format name. See "Format of Input Data" on page 263 for a description of these formats.

Length of input data

Input Length: Binary(4)

The length of the input data structure.

Format name

Input Format name: Char(8)

The content and format of the input data. The possible format name follows:

- LDIF0100 Import LDIF file
- LDIF0200 Import LDIF file with options

See "Format of Input Data" on page 263 for a description of this format.

Error code
The structure in which to return error information. For the format of the structure, see Error parameter.

Format of Input Data
For details about the format of the input data, see the following section. For details about the fields in each format, see "Field Descriptions."

LDIF0100 Format

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td>Dec</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>BINARY(4) Offset to LDIF file</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>BINARY(4) Length of LDIF file</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>BINARY(4) Offset to administrator DN</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>BINARY(4) Length of administrator DN</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>BINARY(4) Offset to administrator password</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>BINARY(4) Length of administrator password</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHAR(*) LDIF file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHAR(*) Administrator DN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHAR(*) Administrator password</td>
</tr>
</tbody>
</table>

LDIF0200 Format

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td>Dec</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>BINARY(4) Offset to LDIF file</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>BINARY(4) Length of LDIF file</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>BINARY(4) Offset to administrator DN</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>BINARY(4) Length of administrator DN</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>BINARY(4) Offset to administrator password</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>BINARY(4) Length of administrator password</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>BINARY(4) Level indicator</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Additional fields if level indicator is equal to 1 or greater:</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>BINARY(4) Replicate imported data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Variable length string fields:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHAR(*) LDIF file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHAR(*) Administrator DN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHAR(*) Administrator password</td>
</tr>
</tbody>
</table>

Field Descriptions
Additional fields indicator. Additional fields supplied for a format. See the format descriptions for possible uses and values of this field.
**Administrator DN.** The distinguished name of the server administrator. This field is specified in UTF-16 (CCSID 13488).

**Administrator password.** The password for the server administrator. This field is specified in UTF-16 (CCSID 13488).

**LDIF file.** The integrated file system path name of the LDIF file to be used. This field is specified in UTF-16 (CCSID 13488).

**Length of administrator DN.** The length, in UTF-16 (CCSID 13488) characters, of the administrator DN field.

**Length of administrator password.** The length, in UTF-16 (CCSID 13488) characters, of the administrator password field.

**Length of LDIF file.** The length, in UTF-16 (CCSID 13488) characters, of the LDIF file field.

**Level indicator.** The level indicator of the data supplied for a format. See the format descriptions for possible uses and values of this field.

**Offset to administrator DN.** The offset, in bytes, from the start of the input data to the administrator DN field.

**Offset to administrator password.** The offset, in bytes, from the start of the input data to the administrator password field.

**Offset to LDIF file.** The offset, in bytes, from the start of the input data to the LDIF file field.

**Replicate imported data.** Indicates whether imported data should be replicated to replica servers. This option could be used, for example, when initializing an additional master server so that it does not attempt to replicate data to servers already containing this data. If format LDIF0100 is used, imported data is replicated. The following values may be specified Whether the directory server will replicate imported data. The following values may be specified:

0  Imported data is not replicated.
1  Imported data is replicated. The value 1 can only be specified if a subtree DN is specified.

**Error Messages**

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLD0202 E</td>
<td>Administrator DN or password not correct.</td>
</tr>
<tr>
<td>GLD0213 E</td>
<td>Error opening or creating file.</td>
</tr>
<tr>
<td>GLD0215 E</td>
<td>Server has not been configured.</td>
</tr>
<tr>
<td>GLD0218 E</td>
<td>*ALLOBJ and *I0SYSXF special authorities required.</td>
</tr>
<tr>
<td>GLD0225 E</td>
<td>&amp;1 items added to directory, &amp;2 items not added.</td>
</tr>
<tr>
<td>GLD0226 E</td>
<td>LDAP server is read-only.</td>
</tr>
</tbody>
</table>

API introduced: V4R3
List Directory Server Attributes (QgldLstDirSvrA)

Required Parameter Group:

1  Qualified user space name             Input  Char(20)
2  Format name                           Input  Char(8)
3  Error code                            I/O    Char(*)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDUAPI
Threadsafe: No

The List Directory Server Attributes (QgldLstDirSvrA) API retrieves a list of directory server attributes including the following:
- Suffixes present on the server
- Attribute indexes maintained by the underlying database
- Network server publishing attributes associated with the LDAP server.
- IP address information

Authorities and Locks

User Space Library Authority
  *EXECUTE

User Space Authority
  *CHANGE

User Space Lock
  An exclusive, no-read lock is obtained on the list space.

Required Parameter Group

Qualified user space name
  INPUT; CHAR(20)
  The user space that is to receive the created list. The first 10 characters contain the user space name, and the second 10 characters contain the name of the library where the user space is located. The content and format of this space is determined by the format name. See “Format of Output Data” on page 266 for a description of these formats.

Format name
  INPUT; CHAR(8)
  The content and format of the data to be retrieved. The possible format names follow:
  LSVR0200  Retrieve a list of suffixes on the server.
  LSVR0300  Retrieve a list of database indexes maintained by the server.
  LSVR0500  Retrieve a list of network server publishing attributes associated with the LDAP server.
  LSVR0600  Retrieve a list of referral servers.
  LSVR0800  Retrieve a list of IP addresses

  See “Format of Output Data” on page 266 for a description of these formats.

Error code
  I/O; CHAR(*)
  The structure in which to return error information. For the format of the structure, see Error Code Parameter
Format of Output Data

The user space contains:
- A user area
- A generic area
- An input parameter section
- A header section
- A list data section:
  - LSVR0200
  - LSVR0300
  - LSVR0500
  - LSVR0600
  - LSVR0800

For details about the user area and generic header, see User Space Format for List APIs. For details about the remaining items, see the following sections. For detailed descriptions of the fields in the list that is returned, see “Field Descriptions” on page 268.

When you retrieve list entry information from the list space, do not use the entry size that is returned in the generic header. Instead, use the displacement to next entry field that is returned in each list entry. If you do not use the displacement to next entry field, the results may not be valid.

LSVR0200 Format

The LSVR0200 format is used to retrieve a list of the directory suffixes present on this server.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Dec</td>
<td>Hex</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>CHAR(*)</td>
<td></td>
</tr>
</tbody>
</table>

LSVR0300 Format

The LSVR0300 format is used to retrieve information about database indexes maintained by the server. The indexes are used to speed up retrieval of objects when a directory server client searches for specified object attributes.

Starting with V4R5M0, this format is not supported. Database index information is to be retrieved using an LDAP client or the Directory Management Tool (DMT) starting with V4R5M0.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Dec</td>
<td>Hex</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>12</td>
</tr>
</tbody>
</table>
### LSVR0500 Format

The LSVR0500 format is used to retrieve the network server publishing attributes associated with the server.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td>Field</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Attribute name</td>
</tr>
</tbody>
</table>

### LSVR0600 Format

The LSVR0600 format is used to retrieve a list of referral servers.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td>Field</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>Displacement to next entry</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Displacement to publishing agent name</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>Length of publishing agent name</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>Displacement to server name</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>Length of server name</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>Displacement to bind DN</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>Length of bind DN</td>
</tr>
<tr>
<td>32</td>
<td>20</td>
<td>LDAP port number</td>
</tr>
<tr>
<td>36</td>
<td>24</td>
<td>Connection type</td>
</tr>
<tr>
<td>40</td>
<td>28</td>
<td>Displacement to parent distinguished name</td>
</tr>
<tr>
<td>44</td>
<td>2C</td>
<td>Length of parent distinguished name</td>
</tr>
<tr>
<td>48</td>
<td>30</td>
<td>Publishing agent disabled</td>
</tr>
<tr>
<td>52</td>
<td>34</td>
<td>Reserved</td>
</tr>
<tr>
<td>56</td>
<td>38</td>
<td>Kerberos authentication indicator</td>
</tr>
<tr>
<td>60</td>
<td>3C</td>
<td>Displacement to Kerberos key tab file</td>
</tr>
<tr>
<td>64</td>
<td>40</td>
<td>Length of Kerberos key tab file</td>
</tr>
<tr>
<td>68</td>
<td>44</td>
<td>Displacement to Kerberos principal</td>
</tr>
<tr>
<td>72</td>
<td>48</td>
<td>Length of Kerberos principal</td>
</tr>
<tr>
<td>76</td>
<td>4C</td>
<td>Displacement to Kerberos realm</td>
</tr>
<tr>
<td>80</td>
<td>50</td>
<td>Length of Kerberos realm</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Publishing agent name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Server name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bind DN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Parent distinguished name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Kerberos key tab file</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Kerberos principal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Kerberos realm</td>
</tr>
</tbody>
</table>
### LSVR0800 Format

The LSVR0800 format is used to retrieve a list of the IP addresses to which the directory server connects.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>BINARY(4)</td>
<td>Displacement to next entry</td>
</tr>
<tr>
<td>4</td>
<td>BINARY(4)</td>
<td>Displacement to referral server URL</td>
</tr>
<tr>
<td>8</td>
<td>BINARY(4)</td>
<td>Length of referral server URL.</td>
</tr>
<tr>
<td>12</td>
<td>BINARY(4)</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>CHAR(*)</td>
<td>Referral server URL.</td>
</tr>
</tbody>
</table>

### Field Descriptions

**Attribute name.** The name of a directory object attribute for which database indexes will be maintained. This field is specified in UTF-16 (CCSID 13488). The following special value may also be returned:

*DEFAULT* The rules for this attribute apply to all attributes for which no explicit rules have been defined.

**Bind DN.** A distinguished name to use when publishing objects to the directory. This field is specified in UTF-16 (CCSID 13488). The following special value may be specified:

*NONE* No value is specified.

**Connection type.** The type of connection to use to the LDAP server. The following values may be returned:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Nonsecure</td>
</tr>
<tr>
<td>2</td>
<td>Secured, using SSL</td>
</tr>
</tbody>
</table>

**Displacement to attribute name.** The displacement, in bytes, from the start of the current entry to the attribute name field.

**Displacement to bind DN.** The displacement, in bytes, from the start of the current entry to the bind DN field.

**Displacement to IP address.** The displacement, in bytes, from the start of the current entry to the IP address field.

**Displacement to Kerberos key tab file.** The displacement, in bytes, from the start of the current entry to the Kerberos key tab file field.
Displacement to Kerberos principal. The displacement, in bytes, from the start of the current entry to the Kerberos principal field.

Displacement to Kerberos realm. The displacement, in bytes, from the start of the current entry to the Kerberos realm field.

Displacement to next entry. The displacement, in bytes, from the start of the current entry to the next entry.

Displacement to parent distinguished name. The displacement, in bytes, from the start of the current entry to the parent distinguished name field.

Displacement to publishing agent name. The displacement, in bytes, from the start of the current entry to the publishing agent name field.

Displacement to referral server URL. The displacement, in bytes, from the start of the current entry to the referral server URL field.

Displacement to server name. The displacement, in bytes, from the start of the current entry to the server name field.

Displacement to suffix. The displacement, in bytes, from the start of the current entry to the suffix.

Format name specified. The format name specified on the call to this API.

Index type. The kind of database indexes that will be created for an attribute. Creating database indexes improved the performance of directory searches on those attributes. The following values may be returned:

0  No indexes will be created for the attribute.
1  Equal

IP address. The IPv4 or IPv6 address of the client for which the directory server will accept connections. The IP address must already exist to be specified. A value of hexadecimal zeroes and leading zeroes is not allowed. An IPv4 address is expressed in standard dotted-decimal form www.xxx.yyy.zzz; for example, 130.99.128.1. An IPv6 address always has at least one occurrence of a colon (\':\') in the format. Some possible IPv6 address formats would be: ::x (for example, ::1) or ::w.x.y.zzz (for example, ::9.130.4.169). For further IPv6 examples and explanation, refer to the Usage Notes section in the Convert IPv4 and IPv6 Addresses Between Text and Binary Form (inet_pton) API. This field is specified in UTF-16 (CCSID 13488).

The following special value may be returned:

*ALL  All IP addresses defined on the local system will be bound to the server.

Kerberos authentication indicator. The following special values may be specified:

0  Do not support Kerberos authentications.
1  Support Kerberos authentications.

Kerberos key tab file. The integrated file system path name for the key tab file that contains the server’s secret key used for authentication. The QDIRSRV user profile is given authorization to read this file. This field is specified in UTF-16 (CCSID 13488). The following special value may be specified:
*NONE No value is specified.

**Kerberos principal.** The principal in the key tab file to use for authentication. This field is specified in UTF-16 (CCSID 13488). The following special value may be specified:

*NONE No value is specified.

**Kerberos realm.** The realm where the principal is registered to use for authentication. This field is specified in UTF-16 (CCSID 13488). The following special value may be specified:

*NONE No value is specified.

**LDAP port number.** The LDAP server's TCP/IP port.

**Length of attribute name.** The length, in UTF-16 (CCSID 13488) characters, of the attribute name field.

**Length of bind DN.** The length, in UTF-16 (CCSID 13488) characters, of the bind DN field.

**Length of IP address.** The length, in UTF-16 (CCSID 13488) characters, of the IP address field.

**Length of Kerberos key tab file.** The length, in UTF-16 (CCSID 13488) characters, of the Kerberos key tab file field.

**Length of Kerberos principal.** The length, in UTF-16 (CCSID 13488) characters, of the Kerberos principal field.

**Length of Kerberos realm.** The length, in UTF-16 (CCSID 13488) characters, of the Kerberos realm field.

**Length of parent distinguished name.** The length, in UTF-16 (CCSID 13488) characters, of the parent distinguished name field.

**Length of publishing agent name.** The length, in UTF-16 (CCSID 13488) characters, of the publishing agent name field.

**Length of referral server URL.** The length, in UTF-16 (CCSID 13488) characters, of the referral server URL field.

**Length of server name.** The length, in UTF-16 (CCSID 13488) characters, of the server name field.

**Length of suffix.** The length, in UTF-16 (CCSID 13488) characters, of the suffix field.

**Length of update DN.** The length, in UTF-16 (CCSID 13488) characters, of the update DN field.

**Parent distinguished name.** The parent distinguished name to be used. This field is specified in UTF-16 (CCSID 13488).

**Publishing agent name.** The agent which will publish information to a directory server and parent distinguished name. This field is specified in UTF-16 (CCSID 13488).

The following publishing agent names are predefined by the operating system:

* **AS400_COMPUTERS** This agent name is used for publishing system information such as the system and printers.

* **AS400_PRINTSHARES** This agent name is used for publishing print shares to an Active Directory server.
*AS400_USERS
This agent name is used for publishing System Distribution Directory users.

*OS400_TCI_QOS
This agent name is used for publishing TCP/IP Quality of Service policy information.

Publishing agent disabled. Indicates whether or not the publishing agent is disabled. The configuration data still exists, but publishing has been disabled for the publishing agent. The following values may be returned:

0   The publishing agent is enabled.
1   The publishing agent is disabled.

Referral server URL. The uniform resource locator (URL) of the referral server. This field is specified in UTF-16 (CCSID 13488).

Reserved. A reserved field. This field must be set to zero.

Server name. The name of the server. This field is specified in UTF-16 (CCSID 13488).

Suffix. The directory name for the starting point of a directory information tree. This field is specified in UTF-16 (CCSID 13488).

Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF24B4 E</td>
<td>Severe error while addressing parameter list.</td>
</tr>
<tr>
<td>GLD0215 E</td>
<td>Server has not been configured.</td>
</tr>
<tr>
<td>GLD022F E</td>
<td>Format not supported.</td>
</tr>
</tbody>
</table>

API introduced: V4R3

Publish Directory Object (QgldPubDirObj)

Required Parameter Group:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Input data</td>
<td>Input</td>
<td>Char(*)</td>
</tr>
<tr>
<td>2</td>
<td>Length of input data</td>
<td>Input</td>
<td>Binary(4)</td>
</tr>
<tr>
<td>3</td>
<td>Format name</td>
<td>Input</td>
<td>Char(8)</td>
</tr>
<tr>
<td>4</td>
<td>Error code</td>
<td>I/O</td>
<td>Char(*)</td>
</tr>
</tbody>
</table>

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDPAPI
Threadsafe: No

The Publish Directory Object (QgldPubDirObj) API publishes objects to the directory server. It can be used to perform the following publishing requests:

• Add a new object to the directory.
• Delete an object from the directory.
• Change an object in the directory.
• Change the relative distinguished name of an object in the directory server.

Before this API can be called, the Directory Services property page for the system must be configured. This can be done from iSeries Navigator or by using the Change Directory Server Attributes...
QgldChgDirSrvA) API. The directory server indicates the server to which objects will be published. The parent distinguished name indicates the suffix in the directory to which objects will be published. This parent distinguished name is referred to as a publish point.

**Authorities and Locks**

*ALLOBJ* special authority is required to use this API.

**Required Parameter Group**

**Input data**

INPUT; CHAR(*)

A variable that contains the input data. See “Format of Input Data” for a description of the data associated with a specific format name.

**Length of input data**

INPUT; BINARY(4)

The length of the input data area. The maximum value for this parameter is 16,776,704.

**Format name**

INPUT; CHAR(8)

The format name identifying the type of publishing request. The possible format names follow:

- **POBJ0100** Add a new object to the directory server.
- **POBJ0200** Delete an object from the directory server.
- **POBJ0300** Change an object in the directory server.
- **POBJ0400** Change the relative distinguished name of an object in the directory server.

See “Format of Input Data” for a description of these formats.

**Error code**

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

**Format of Input Data**

For details about the format of the input data, see the following sections. For details about the fields in each format, see “Field Descriptions” on page 275.

**POBJ0100 Format**

This format is used to add a new object to the directory server.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td>Field</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>BINARY(4) Offset to publishing agent name</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>BINARY(4) Length of publishing agent name</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>BINARY(4) Offset to object RDN</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>BINARY(4) Length of object RDN</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>BINARY(4) Offset to attribute entries</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>BINARY(4) Number of attribute entries</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>CHAR(40) Reserved</td>
</tr>
<tr>
<td></td>
<td>CHAR(*)</td>
<td>Publishing agent name</td>
</tr>
</tbody>
</table>

272 IBM Systems - iSeries: Lightweight Directory Access Protocol (LDAP) APIs
### POBJ0200 Format

This format is used to delete an object from the directory server.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>BINARY(4)</td>
<td>Displacement to next value</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>BINARY(4)</td>
<td>Displacement to attribute name</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>C</td>
<td>BINARY(4)</td>
<td>Length of attribute name</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td></td>
<td>BINARY(4)</td>
<td>Number of attribute values</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td></td>
<td>BINARY(4)</td>
<td>Attribute value data type</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td></td>
<td>CHAR(8)</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CHAR(*)</td>
<td>Attribute name</td>
</tr>
</tbody>
</table>

### POBJ0300 Format

This format is used to change an object in the directory server.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>BINARY(4)</td>
<td>Offset to publishing agent name</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>BINARY(4)</td>
<td>Length of publishing agent name</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>C</td>
<td>BINARY(4)</td>
<td>Offset to object RDN</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td></td>
<td>BINARY(4)</td>
<td>Length of object RDN</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td></td>
<td>BINARY(4)</td>
<td>Delete directory subtree</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td></td>
<td>CHAR(44)</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CHAR(*)</td>
<td>Publishing agent name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CHAR(*)</td>
<td>Object RDN</td>
</tr>
</tbody>
</table>
### POBJ0400 Format

This format is used to change the relative distinguished name (RDN) of an object in the directory server.

- **Offset**: The position in the data stream where the offset to the publishing agent name is located.
- **Hex**: The hexadecimal representation of the offset.
- **Type**: The type of data (BINARY or CHAR).
- **Field**: The field being modified or the length of the data.

#### Offset Table

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>BINARY(4)</td>
<td>Offset to publishing agent name</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>BINARY(4)</td>
<td>Length of publishing agent name</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>BINARY(4)</td>
<td>Offset to object RDN</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>12</td>
<td>BINARY(4)</td>
<td>Length of object RDN</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>10</td>
<td>BINARY(4)</td>
<td>Offset to new object RDN</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>14</td>
<td>BINARY(4)</td>
<td>Length of new object RDN</td>
</tr>
</tbody>
</table>

#### Example

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>BINARY(4)</td>
<td>Displacement to next entry</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>BINARY(4)</td>
<td>Change type</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>BINARY(4)</td>
<td>Displacement to attribute entries</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>12</td>
<td>BINARY(4)</td>
<td>Number of attribute entries</td>
</tr>
</tbody>
</table>

#### Attribute Entries

- **Offset**: The position in the data stream where the offset to the attribute entries starts.
- **Hex**: The hexadecimal representation of the offset.
- **Type**: The type of data (BINARY or CHAR).
- **Field**: The field being modified or the length of the data.

#### Attribute Values

- **Offset**: The position in the data stream where the offset to the attribute values starts.
- **Hex**: The hexadecimal representation of the offset.
- **Type**: The type of data (BINARY or CHAR).
- **Field**: The field being modified or the length of the data.

#### POBJ0400 Format Details

- **POBJ0400 Format** is used to change the relative distinguished name (RDN) of an object in the directory server.
- The format includes fields for specifying the offset to the publishing agent name, length of the publishing agent name, offset to object RDN, length of object RDN, offset to new object RDN, and length of new object RDN.
- The format also includes fields for specifying the number of attribute entries and attribute values.

---

274 IBM Systems - iSeries: Lightweight Directory Access Protocol (LDAP) APIs
<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>18</td>
<td>1B</td>
<td>BINARY(4)</td>
<td>Delete old RDN</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>1C</td>
<td>CHAR(36)</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CHAR(*)</td>
<td>Publishing agent name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CHAR(*)</td>
<td>Object RDN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CHAR(*)</td>
<td>New object RDN</td>
</tr>
</tbody>
</table>

**Field Descriptions**

**Add object if it does not exist.** Create the object if a request is made to modify an object that does not exist. The following values may be specified:

0  Do not create the object if it does not exist.
1  Create the object if it does not exist. All required attributes for the object must be specified on the API in order for the object to be successfully created.

**Attribute name.** The name of a directory object attribute. This field is specified in UTF-16 (CCSID 13488).

**Attribute value.** The value of a directory object attribute.

**Attribute value data type.** The type of data for the attribute values. The following values may be specified.

1  The attribute values are specified in UTF-16 (CCSID 13488).
2  The attribute values contain binary data.
3  The attribute values contain integer data.
4  The attribute values contain boolean data.

**Change type.** The type of change being made to a directory object. The following values may be specified:

1  Add a new attribute
2  Delete an attribute
3  Replace an attribute
4  Add an attribute if it does not exist
5  Add an attribute value if it does not exist
6  Delete an attribute if it exists
7  Delete an attribute value if it exists

**Delete directory subtree.** The directory object and any child directory objects should be deleted. The following values may be specified:

0  Do not delete the directory subtree. Only the directory object itself will be deleted.
1  Delete the directory subtree.
2  Delete the directory subtree. The root of the subtree will not be deleted.

**Delete old RDN.** The old relative distinguished name (RDN) of a directory object should be deleted. The following values may be specified:
0  Do not delete the old RDN. The old RDN attribute value will be retained as an attribute of the object.
1  Delete the old RDN.

**Displacement to attribute entries.** The displacement, in bytes, from the start of the current entry to the attribute entries.

**Displacement to attribute name.** The displacement, in bytes, from the start of the current entry to the attribute name field.

**Displacement to attribute value.** The displacement, in bytes, from the start of the current entry to the attribute value field.

**Displacement to attribute values.** The displacement, in bytes, from the start of the current entry to the attribute values.

**Displacement to next entry.** The displacement, in bytes, from the start of the current entry to the next entry in the input data.

**Displacement to next value.** The displacement, in bytes, from the start of the current value to the next value in the input data.

**Length of attribute name.** The length, in Unicode characters, of the attribute name field.

**Length of attribute value.** The length of the attribute value field. If the attribute value is specified in UTF-16 (CCSID 13488), this is the length in Unicode characters. If the attribute value contains binary data, this is the length in bytes. If the attribute value contains integer or boolean data, this field must contain the value 4.

**Length of new object RDN.** The length, in Unicode characters, of the new object RDN field.

**Length of object RDN.** The length, in Unicode characters, of the object RDN field.

**Length of publishing agent name.** The length, in Unicode characters, of the publishing agent name field.

**New object RDN.** The new relative distinguished name (RDN) of the directory object. This field is specified in UTF-16 (CCSID 13488).

**Number of attribute entries.** The number of attribute entries.

**Number of attribute values.** The number of attribute values.

**Number of modification entries.** The number of modification entries.

**Object RDN.** The relative distinguished name (RDN) of the directory object being published. This name, combined with the publishing point specified during configuration, form a distinguished name (DN). This field is specified in UTF-16 (CCSID 13488). For example, if the publishing point is ‘O=ACME Corp., C=US’ and the object RDN is ‘CN=Bart’, the object DN to be published is ‘CN=Bart, O=ACME Corp., C=US’.

**Offset to attribute entries.** The offset, in bytes, from the start of the input data area to the attribute entries.

**Offset to modification entries.** The offset, in bytes, from the start of the input data area to the modification entries.
Offset to new object RDN. The offset, in bytes, from the start of the input data area to the new object RDN field.

Offset to object RDN. The offset, in bytes, from the start of the input data area to the object RDN field.

Offset to publishing agent name. The offset, in bytes, from the start of the input data area to the publishing agent name field.

Publishing agent name. The agent making the publishing request. This determines where in the directory the object will be published. The publishing agent information must be configured using the QglDChgDirSvrA API before calling this API. This field is specified in UTF-16 (CCSID 13488).

The following publishing agent names are predefined by the operating system:

* AS400_COMPUTERS This agent name is used for publishing system information such as the system and printers.
* AS400_PRINTSHARES This agent name is used for publishing print shares to an Active Directory server.
* AS400_USERS This agent name is used for publishing System Distribution Directory users.
* OS400_TC1_QOS This agent name is used for publishing TCP/IP Quality of Service policy information.

Reserved. A reserved field. This field must be set to binary zero.

Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPA314 E</td>
<td>Memory allocation error.</td>
</tr>
<tr>
<td>CPFB802 E</td>
<td>The caller of the API must have *ALLOBJ special authority.</td>
</tr>
<tr>
<td>CPFB803 E</td>
<td>Publishing agent &amp;1 is not configured or has been disabled.</td>
</tr>
<tr>
<td>CPFB805 E</td>
<td>Value specified in input data is not valid.</td>
</tr>
</tbody>
</table>

API introduced: V4R4

Retrieve Directory Server Attributes (QglDRetDirSvrA)

Required Parameter Group:

1. Receiver variable Output Char(*)
2. Length of receiver variable Input Binary(4)
3. Format name Input Char(8)
4. Error code I/O Char(*)

Default Public Authority: *USE
Library Name/Service Program: QSYS/QGLDUAPI
Threadsafe: No

The Retrieve Directory Server Attributes (QglDRetDirSvrA) API retrieves information about the directory server configuration. It can be used to retrieve information about:

- General server properties
- Encrypted communications configuration. The Secure Sockets Layer (SSL) is used for encrypted communications.
- Performance settings
- Auditing settings
Authorities and Locks

To retrieve format RSVR0700, Server auditing information, the caller of this API must have either *ALLOBJ or *AUDIT special authorities.

For all other formats, no i5/OS special authority is required.

Required Parameter Group

Receiver variable

OUTPUT; CHAR(*)

The variable to receive output data. See “Format of Output Data” for a description of the format of the output data associated with a specific format name.

Length of receiver variable

INPUT; BINARY(4)

The length of the receiver variable area.

Format name

INPUT; CHAR(8)

The format name identifying the type of information to be retrieved. The possible format names follow:

- RSVR0100 Basic server configuration
- RSVR0400 Attributes for publishing users in an LDAP directory
- RSVR0700 Server auditing information
- RSVR0900 Server administration information

See “Format of Output Data” for a description of these formats.

Error code

I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error Code Parameter.

Format of Output Data

For details about the format of the output data, see the following sections. For details about the fields in each format, see “Field Descriptions” on page 281.

RSVR0100 Format

This format is used to retrieve basic server configuration information.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td>Type</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>Offset</td>
<td>Type</td>
<td>Field</td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>Dec</td>
<td>Hex</td>
<td>Field</td>
</tr>
<tr>
<td>32</td>
<td>20</td>
<td>BINARY(4) Current cipher protocols</td>
</tr>
<tr>
<td>36</td>
<td>24</td>
<td>BINARY(4) Installed cipher protocols</td>
</tr>
<tr>
<td>40</td>
<td>28</td>
<td>BINARY(4) Search time limit</td>
</tr>
<tr>
<td>44</td>
<td>2C</td>
<td>BINARY(4) Search size limit</td>
</tr>
<tr>
<td>48</td>
<td>30</td>
<td>BINARY(4) Maximum connections</td>
</tr>
<tr>
<td>52</td>
<td>34</td>
<td>BINARY(4) Reserved</td>
</tr>
<tr>
<td>56</td>
<td>38</td>
<td>BINARY(4) Referral port</td>
</tr>
<tr>
<td>60</td>
<td>3C</td>
<td>BINARY(4) Password format</td>
</tr>
<tr>
<td>64</td>
<td>40</td>
<td>BINARY(4) Offset to referral server</td>
</tr>
<tr>
<td>68</td>
<td>44</td>
<td>BINARY(4) Length of referral server</td>
</tr>
<tr>
<td>72</td>
<td>48</td>
<td>BINARY(4) Offset to administrator distinguished name (DN)</td>
</tr>
<tr>
<td>76</td>
<td>4C</td>
<td>BINARY(4) Length of administrator DN</td>
</tr>
<tr>
<td>80</td>
<td>50</td>
<td>BINARY(4) Offset to update DN</td>
</tr>
<tr>
<td>84</td>
<td>54</td>
<td>BINARY(4) Length of update DN</td>
</tr>
<tr>
<td>88</td>
<td>58</td>
<td>BINARY(4) Reserved</td>
</tr>
<tr>
<td>92</td>
<td>5C</td>
<td>BINARY(4) Reserved</td>
</tr>
<tr>
<td>96</td>
<td>60</td>
<td>BINARY(4) Offset to database path</td>
</tr>
<tr>
<td>100</td>
<td>64</td>
<td>BINARY(4) Length of database path</td>
</tr>
<tr>
<td>104</td>
<td>68</td>
<td>BINARY(4) Reserved</td>
</tr>
<tr>
<td>108</td>
<td>6C</td>
<td>BINARY(4) SSL authentication method</td>
</tr>
<tr>
<td>112</td>
<td>70</td>
<td>BINARY(4) Number of database connections</td>
</tr>
<tr>
<td>116</td>
<td>74</td>
<td>BINARY(4) Schema checking level</td>
</tr>
<tr>
<td>120</td>
<td>78</td>
<td>BINARY(4) Offset to master server URL</td>
</tr>
<tr>
<td>124</td>
<td>7C</td>
<td>BINARY(4) Length of master server URL</td>
</tr>
<tr>
<td>128</td>
<td>80</td>
<td>BINARY(4) Change log indicator</td>
</tr>
<tr>
<td>132</td>
<td>84</td>
<td>BINARY(4) Maximum number of change log entries</td>
</tr>
<tr>
<td>136</td>
<td>88</td>
<td>BINARY(4) Terminate idle connections</td>
</tr>
<tr>
<td>140</td>
<td>8C</td>
<td>BINARY(4) Kerberos authentication indicator</td>
</tr>
<tr>
<td>144</td>
<td>90</td>
<td>BINARY(4) Offset to Kerberos key tab file</td>
</tr>
<tr>
<td>148</td>
<td>94</td>
<td>BINARY(4) Length of Kerberos key tab file</td>
</tr>
<tr>
<td>152</td>
<td>98</td>
<td>BINARY(4) Kerberos to DN mapping indicator</td>
</tr>
<tr>
<td>156</td>
<td>9C</td>
<td>BINARY(4) Offset to Kerberos administrator ID</td>
</tr>
<tr>
<td>160</td>
<td>A0</td>
<td>BINARY(4) Length of Kerberos administrator ID</td>
</tr>
<tr>
<td>164</td>
<td>A4</td>
<td>BINARY(4) Offset to Kerberos administrator realm</td>
</tr>
<tr>
<td>168</td>
<td>A8</td>
<td>BINARY(4) Length of Kerberos administrator realm</td>
</tr>
<tr>
<td>172</td>
<td>AC</td>
<td>BINARY(4) Event notification registration indicator</td>
</tr>
<tr>
<td>176</td>
<td>B0</td>
<td>BINARY(4) Maximum event registrations for connection</td>
</tr>
<tr>
<td>180</td>
<td>B4</td>
<td>BINARY(4) Maximum event registrations for server</td>
</tr>
<tr>
<td>184</td>
<td>B8</td>
<td>BINARY(4) Maximum operations per transaction</td>
</tr>
<tr>
<td>188</td>
<td>BC</td>
<td>BINARY(4) Maximum pending transactions</td>
</tr>
<tr>
<td>Offset</td>
<td>Dec</td>
<td>Hex</td>
</tr>
<tr>
<td>--------</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>192</td>
<td>192</td>
<td>C0</td>
</tr>
<tr>
<td>196</td>
<td>196</td>
<td>C4</td>
</tr>
<tr>
<td>200</td>
<td>200</td>
<td>C8</td>
</tr>
<tr>
<td>204</td>
<td>204</td>
<td>CC</td>
</tr>
<tr>
<td>208</td>
<td>208</td>
<td>D0</td>
</tr>
<tr>
<td>212</td>
<td>212</td>
<td>D4</td>
</tr>
<tr>
<td>216</td>
<td>216</td>
<td>D8</td>
</tr>
<tr>
<td>220</td>
<td>220</td>
<td>DC</td>
</tr>
<tr>
<td>224</td>
<td>224</td>
<td>E0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**RSVR0400 Format**

This format is used to retrieve the attributes for publishing users in an LDAP directory. User information from the system distribution directory can be published to an LDAP server by the Synchronize System Distribution Directory to LDAP (QGLDSSDD) API and from iSeries Navigator. The publishing attributes define how to publish user information.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Dec</th>
<th>Hex</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>BINARY(4)</td>
<td>Bytes returned</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>BINARY(4)</td>
<td>Bytes available</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>BINARY(4)</td>
<td>Offset to server name</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>C</td>
<td>BINARY(4)</td>
<td>Length of server name</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>10</td>
<td>BINARY(4)</td>
<td>LDAP port number</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>14</td>
<td>BINARY(4)</td>
<td>Connection type</td>
</tr>
<tr>
<td>24</td>
<td>24</td>
<td>18</td>
<td>BINARY(4)</td>
<td>Offset to parent distinguished name.</td>
</tr>
<tr>
<td>28</td>
<td>28</td>
<td>1C</td>
<td>BINARY(4)</td>
<td>Length of parent distinguished name.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CHAR(*)</td>
<td>Server name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CHAR(*)</td>
<td>Parent distinguished name.</td>
</tr>
</tbody>
</table>
RSVR0700 Format
This format is used to retrieve server auditing configuration information.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td>Dec</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>BINARY(4)</td>
</tr>
</tbody>
</table>

RSVR0900 Format
This format is used to retrieve server administration information.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Type</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>Hex</td>
<td>Dec</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>BINARY(4)</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>BINARY(4)</td>
</tr>
</tbody>
</table>

Field Descriptions
ACL model. The ACL model that is being used. The following special values may be returned:

- 0: The ACL model being used supports access-class level permissions. This is the ACL model the directory server used prior to V5R1M0.
- 1: The ACL model being used supports both access-class level permissions and attribute-level ACL permissions.

Administrator DN. A distinguished name (DN) that has access to all objects in the directory. This field is specified in UTF-16 (CCSID 13488).

Bytes available. The number of bytes of data available to be returned. All available data is returned if enough space is provided.

Bytes returned. The number of bytes of data returned.

Change log indicator. The indicator of whether a change log exists for entries that have been added, changed and deleted. The following values may be returned:

- 0: No, a change log does not exist
- 1: Yes, a change log exists

Connection type. The type of connection to use to the LDAP server. The following values may be returned:

- 1: Nonsecure
- 2: Secured, using SSL
**Current cipher protocols.** The cipher protocols that the server allows when using encrypted connections. The value is the sum of zero or more of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0100</td>
<td>Triple Data Encryption Standard (DES) Secure Hash Algorithm (SHA) (U.S.)</td>
</tr>
<tr>
<td>0x0200</td>
<td>DES SHA (U.S)</td>
</tr>
<tr>
<td>0x0400</td>
<td>Rivest Cipher 4 (RC4) SHA (U.S)</td>
</tr>
<tr>
<td>0x0800</td>
<td>RC4 Message Digest (MD) 5 (U.S)</td>
</tr>
<tr>
<td>0x1000</td>
<td>RC2 MD5 (export)</td>
</tr>
<tr>
<td>0x2000</td>
<td>RC4 MD5 (export)</td>
</tr>
<tr>
<td>0x4000</td>
<td>Advanced Encryption Standard (AES) SHA 128 bit (U.S.)</td>
</tr>
<tr>
<td>0x8000</td>
<td>Advanced Encryption Standard (AES) SHA 256 bit (U.S.)</td>
</tr>
</tbody>
</table>

**Database path.** The integrated file system path name of the library containing the directory database. This field is specified in UTF-16 (CCSID 13488).

**Encrypted port number.** The port number to use for encrypted connections. The standard port number for encrypted connections is 636.

**Event notification registration indicator.** Indicator of whether to allow client to register for event notification. The following special values may be returned:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Do not allow clients to register for event notification.</td>
</tr>
<tr>
<td>1</td>
<td>Allow clients to register for event notification.</td>
</tr>
</tbody>
</table>

**Installed cipher protocols.** The cipher protocols installed on the system. Refer to the current cipher protocols field for a description of the values.

**Kerberos administrator ID.** The name of the Kerberos administrator. This field is specified in UTF-16 (CCSID 13488). The following special value may be returned:

*NONE No value is specified.

**Kerberos administrator realm.** The realm in which the kerberos administrator is registered. This field is specified in UTF-16 (CCSID 13488). The following special value may be returned:

*NONE No value is specified.

**Kerberos authentication indicator.** The following special values may be returned:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Do not support Kerberos authentications.</td>
</tr>
<tr>
<td>1</td>
<td>Support Kerberos authentications.</td>
</tr>
</tbody>
</table>

**Kerberos key tab file.** The integrated file system path name for the key tab file that contains the server’s secret key used for authentication. This field is specified in UTF-16 (CCSID 13488). The following special value may be returned:

*NONE No value is specified.

**Kerberos to DN mapping indicator.**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Map the Kerberos ID to pseudo DN. A pseudo DN can be used to uniquely identify an LDAP user object of the form ‘ibm-kerberosName=principal@realm’ or ‘ibm-kn=principal@realm’.</td>
</tr>
</tbody>
</table>
Use associated DN in directory. The LDAP server will attempt to find an entry in the directory that contains the kerberos principle and realm as one of its attributes. Once found, this DN will then be used to determine the client’s authorizations to the directory.

**LDAP port number.** The LDAP server’s TCP/IP port.

**Length of administrator DN.** The length, in UTF-16 (CCSID 13488) characters, of the administrator DN field.

**Length of database path.** The length, in UTF-16 (CCSID 13488) characters, of the database path field.

**Length of Kerberos administrator ID.** The length, in UTF-16 (CCSID 13488) characters, of the Kerberos Administrator ID field.

**Length of Kerberos administrator realm.** The length, in UTF-16 (CCSID 13488) characters, of the Kerberos administrator realm field.

**Length of Kerberos key tab file.** The length, in UTF-16 (CCSID 13488) characters, of the Kerberos key tab file field.

**Length of master server URL.** The length, in UTF-16 (CCSID 13488) characters, of the master server URL field.

**Length of parent distinguished name.** The length, in UTF-16 (CCSID 13488) characters, of the parent distinguished name field.

**Length of projected suffix.** The length, in UTF-16 (CCSID 13488) characters, of the projected suffix field.

**Length of server administration URL.** The length, in UTF-16 (CCSID 13488) characters, of the server administration URL field.

**Length of server name.** The length, in UTF-16 (CCSID 13488) characters, of the server name field.

**Length of referral server.** The length, in UTF-16 (CCSID 13488) characters, of the referral server field.

**Length of update DN.** The length, in UTF-16 (CCSID 13488) characters, of the update DN field.

**Level of authority integration.** The level of i5/OS authority integration to use to determine if a distinguished name (DN) can become an LDAP administrator. The following special values may be specified:

0 Do not apply ‘Directory Server Administrator’ (QIBM_DIRSrv_ADMIN) function identifier to bound distinguished names to determine LDAP administrators.

1 Allow bound distinguished names that refer directly to user profiles to become LDAP administrators if the user profile is identified in the ‘Directory Server Administrator’ (QIBM_DIRSrv_ADMIN) function identifier.

**Log client messages.** Whether the directory server will log client messages in the server joblog. The following values may be returned:

0 The directory server will not log client messages in the server joblog.

1 The directory server will log client messages in the server joblog.

**Master server URL.** The uniform resource locator (URL) of the master server. This field is specified in UTF-16 (CCSID 13488). The following special value may be returned:
Maximum connections. Returns the maximum number of simultaneous connections that can be established with the server.

Starting with V5R1M0, this field is no longer supported and the value returned is 0. The following special value may be returned:

0   Do not limit the number of connections.

Maximum event registrations for connection. The following special values may be returned:

0   Do not limit the number of event registrations for connection.

Maximum event registrations for server. The following special values may be returned:

0   Do not limit the number of event registrations for server.

Maximum age of change log entries. The age, in seconds, of change log entries that can be stored. If the maximum is reached, the change log entries will be deleted starting with the oldest entry. This value only valid if ‘Change log indicator’ is set to 1. The following special values may be returned:

0   The age of change log entries is not limited.

Maximum number of change log entries. The maximum number of change log entries that can be stored. If the maximum is reached, the change log entries will be deleted starting with the oldest entry. This value only valid if ‘Change log indicator’ is set to 1. The following special values may be returned:

0   The number of change log entries is not limited.

Maximum operations per transaction. The maximum number of operations that are allowed for each transaction. Transaction support allows a group of directory changes to be handled as a single transaction.

Maximum pending transactions. The maximum number of pending transactions allowed. Transaction support allows a group of directory changes to be handled as a single transaction.

Number of database connections. The number of database connections used by the server.

Offset to administrator DN. The offset, in bytes, from the start of the receiver variable to the administrator DN field.

Offset to database path. The offset, in bytes, from the start of the receiver variable to the database path field.

Offset to Kerberos administrator ID. The offset, in bytes, from the start of the input data area to the Kerberos administrator ID field.

Offset to Kerberos administrator realm. The offset, in bytes, from the start of the input data area to the Kerberos administrator realm field.
Offset to Kerberos key tab file. The offset, in bytes, from the start of the input data area to the Kerberos key tab file field.

Offset to master server URL. The offset, in bytes, from the start of the receiver variable to the master server URL field.

Offset to parent distinguished name. The offset, in bytes, from the start of the receiver variable to the parent distinguished name field.

Offset to projected suffix. The offset, in bytes, from the start of the input data area to the projected suffix field.

Offset to referral server. The offset, in bytes, from the start of the receiver variable to the referral server field.

Offset to server administration URL. The offset, in bytes, from the start of the receiver variable to the server administration URL field.

Offset to server name. The offset, in bytes, from the start of the receiver variable to the server name field.

Offset to update DN. The offset, in bytes, from the start of the receiver variable to the update DN field.

Parent distinguished name. The parent distinguished name for published objects. For example, if the parent distinguished name is ‘ou=rochester, o=ibm, c=us’, a published directory object for user John Smith might be ‘cn=john smith, ou=rochester, o=ibm, c=us’. This field is specified in UTF-16 (CCSID 13488).

Password format. The format of the encrypted password. The following values may be returned:

1 Unencrypted. The clear text password is stored in a validation list and can be returned by searches or used for DIGEST-MD5 SASL authentication.
2 SHA. (Default)
3 MD5.
4 Crypt (The password is one-way hashed using a modified DES algorithm. The ‘crypt’ algorithm originally was used by many UNIX operating systems for password protection.)

Projected suffix. The suffix under which all projected objects for this server reside including user and group profiles. This field is specified in UTF-16 (CCSID 13488).

Read only. Whether the directory server allows changes to be made to the directory contents. The following values may be returned:

0 The directory server is not read only. Updates are allowed to the directory.
1 The directory server is read only. Updates are not allowed to the directory.

Read only projected suffix. Whether the directory server will allow updates to be made to the projected suffix. The following values may be returned:

0 The directory server projected suffix is not read only. Updates are allowed to the projected suffix.
1 The directory server projected suffix is read only. Updates are not allowed to the projected suffix.

Read only schema. Whether the directory server will allow updates to be made to the directory schema. The following values may be returned:

Lightweight Directory Access Protocol (LDAP) APIs 285
The directory server schema is not read only. Updates are allowed to the schema.

The directory server schema is read only. Updates are not allowed to the schema.

**Referral port.** An optional port number to be returned to a client when a request is made for a directory object that does not reside on this server. The referral port and referral server together are used to form a referral URL. The following special value may be returned:

0 The LDAP port is not specified, the client should use the default LDAP port.

**Referral server.** The IP name of a server to return to a client when a request is made for a directory object that does not reside on this server. This field is specified in UTF-16 (CCSID 13488). The referral port and referral server are used together to form a referral URL. The following special value may be returned:

*NONE No value is specified.

**Reserved.** A reserved field. This field must be set to zero.

**Schema checking level.** The level of schema checking performed by the server. The following values may be returned:

0 None.
1 LDAP version 2.
2 LDAP version 3 strict.
3 LDAP version 3 lenient.

**Search size limit.** The maximum number of entries that the server will return for a given search request. The following special value may be returned:

0 Do not limit the number of entries returned.

**Search time limit.** The maximum time, in seconds, that the server will spend performing a given search request. The following special value may be returned:

0 Do not limit the search time.

**Security.** Whether the server is to use encrypted connections. The following values may be returned:

0 Allow unencrypted connections only.
1 Allow encrypted connections only.
2 Allow both encrypted and unencrypted connections.

**Note:** SSL is used for encrypted connections to the server.

**Security audit option for objects.** When the QAUDCTL system value is set to *OBJAUD, then object auditing can be done in the directory. See the [iSeries Security Reference](#) book for information about Directory Server auditing. The following special values may be returned:

0 Do not do object auditing of the directory objects.
1 Audit changes to directory objects.
2 Audit all access to directory objects. This includes search, compare and change.
**Server is replica.** Whether the server is a master server or a replica server. The following values may be returned:

0  The server is a master server for the directory suffixes present on the server.
1  The server is a replica server for the directory suffixes present on the server.

**Server administration URL.** The server administration URL. This field is specified in UTF-16 (CCSID 13488).

**Server name.** The name of the server. This field is specified in UTF-16 (CCSID 13488).

**SSL authentication method.** The method used during SSL authentication. The following values may be returned:

1  Server authentication.
3  Server and client authentication.

**Terminate idle connections.** The server will terminate idle connections when necessary. The following values may be returned:

0  Do not terminate idle connections.
1  Terminate idle connections.

**Note:** Starting with V5R1M0, this field is no longer supported and the value returned is 0.

**Transaction time limit.** The maximum time, in seconds, that the server will spend performing a transaction request. Transaction support allows a group of directory changes to be handled as a single transaction.

**Unencrypted port number.** The port number to be used for unencrypted connections. The standard port number is 389.

**Update DN.** The distinguished name that the master server must use when propagating directory updates to this replica server. This field is specified in UTF-16 (CCSID 13488). The following value may be returned:

*NONE  No value is specified.

**Use encrypted connections.** Whether this server should use encrypted connections when making updates to the replica server. The following values may be returned:

0  Use unencrypted connections.
1  Use encrypted connections.

**Version.** Returns the version of the LDAP server.

**Error Messages**

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPFA314 E</td>
<td>Memory allocation error.</td>
</tr>
<tr>
<td>GLD016E E</td>
<td>*ALLOBJ or *AUDIT special authority required.</td>
</tr>
</tbody>
</table>
Synchronize System Distribution Directory to LDAP (QGLDSSDD)

Required Parameter Group:

<table>
<thead>
<tr>
<th></th>
<th>Option</th>
<th>Input</th>
<th>Char(10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Option</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>LDAP user ID</td>
<td>Input</td>
<td>Char(1024)</td>
</tr>
<tr>
<td>3</td>
<td>LDAP user ID password</td>
<td>Input</td>
<td>Char(128)</td>
</tr>
<tr>
<td>4</td>
<td>No longer used</td>
<td>Input</td>
<td>Char(1024)</td>
</tr>
<tr>
<td>5</td>
<td>No longer used</td>
<td>Input</td>
<td>Char(128)</td>
</tr>
<tr>
<td>6</td>
<td>Error Code</td>
<td>I/O</td>
<td>Char(*)</td>
</tr>
</tbody>
</table>

Default Public Authority: *EXCLUDE
Threadsafe: No

The Synchronize System Distribution Directory to LDAP (QGLDSSDD) API publishes system distribution directory entries to an LDAP directory and keeps the LDAP directory synchronized with changes made in the system distribution directory. The following users from the system distribution directory are published:

- Local users
- Remote users that have been added to the local system and have a Simple Mail Transfer Protocol (SMTP) address

The system distribution directory users that are not published are:

- Shadowed users
- Remote users that do not have a SMTP address

The Directory Services property page must be set up. In V4R4 and later, users are automatically published when you set up users in the Directory Services property page for the LDAP server to publish under. Prior to V4R4, this API (QGLDSSDD) must be called regularly to publish the users because publishing users is not automatic prior to V4R4. See "Usage Notes on page 294" for the procedures for setting up the Directory Services property page.

If you are using SSL, the SSL key database information is configured using Digital Certificate Manager. See "Usage Notes on page 294" for information on accessing the Digital Certificate Manager.

When using a V4R4 or later iSeries Navigator client to publish users to a V4R4 or later server, the following no longer applies because this is done automatically. The synchronization is restricted to one LDAP server and one distinguished name to publish to. If you need to change the LDAP server or distinguished name that the system distribution directory information gets published to, first end the synchronization (using option value *END). Then change the LDAP server attributes from iSeries Navigator or from the Change Directory Server Attributes (QgldChgDirSrvA) API. You can then use option *ALL to initialize all the system distribution directory data to the new LDAP server or distinguished name.

Before users can be published, the host and domain name must be set using the Change TCP/IP Domain (CHGTCPPDMN) command. The keywords that must be set are HOSTNAME and DMNNAME.
LDAP uses the distinguished name (dn) as the key for the user. For the system distribution directory entries in LDAP, the distinguished name is the common name (cn) combined with the distinguished name that LDAP is being published to. See “Distinguished Name (dn) and Common Name (cn)” on page 290 for more information.

Note that if changes are made in the LDAP directory, these changes are not synchronized back to the system distribution directory.

Some entries are automatically prevented from being published to LDAP. They are the *ANY system distribution directory entries and some other entries that are IBM-supplied starting with Q (QSECOFR, QDOC, QSYS, QDFTOWN, QUSER for example). A specific user can be prevented from being published to LDAP by doing the following:

1. Add the user-defined field QREPL QLDAP to the system distribution directory. This needs to be done only once per system.

   CHGSYSDIRA USRFNFLD((QREPL QLDAP +ADD +DATA 4))

2. Specify *NO as the value for the QREPL QLDAP user-defined field for those users that you do not want to replicate to LDAP. Any other value or absence of the QREPL QLDAP user-defined field will replicate the user. It is recommended that you either leave the QREPL QLDAP value blank or specify *YES if you want the user to be replicated.

   For example, using Work with Directory Entries (WRKDIRE), option 1 to add a user or option 2 to change a user, press the F20 key to specify user-defined fields. When using the ADDDIRE or CHGDIRE commands, specify USRFNFLD((QREPL QLDAP *NO)) to prevent the user from being replicated.

3. If the user is already replicated to LDAP, and *NO is specified in the QREPL QLDAP user-defined field, then the user will be deleted from the LDAP directory. Likewise, if the value of the QREPL QLDAP user-defined field is changed to anything but *NO, then the user will be added to the LDAP directory.

As an administrator, you must understand some additional items that are needed to synchronize the system distribution directory to LDAP. These include the following:

- inetOrgPerson and publisher object classes used in synchronization.
- How the system distribution directory fields map to LDAP attributes.
- What is a distinguished name and common name and why they are important for synchronization.
- How the i5/OS user profile field is used in LDAP.

See Directory Services (LDAP): Question and Answers for additional information on publishing users.

**inetOrgPerson and publisher Object Class**

If your LDAP server is not on i5/OS, you must ensure that the inetOrgPerson and publisher object classes are defined in the schema file of the server. The inetOrgPerson object class is used in LDAP to store the system distribution directory information. The publisher object class requires a new attribute, publisherName. See SecureWay Directory Schema for documentation on the inetOrgPerson and publisher object class.

**System Distribution Directory to LDAP Mapping**

The system distribution directory entry is published to the LDAP directory by using the inetOrgPerson object class. The following table describes the mapping of system distribution directory fields to attributes of the inetOrgPerson object class.
Table 1: System Distribution Directory Fields Mapped to LDAP Attributes

<table>
<thead>
<tr>
<th>System Distribution Directory Field</th>
<th>LDAP Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>User profile</td>
<td>UID</td>
</tr>
<tr>
<td>Descriptions</td>
<td>description</td>
</tr>
<tr>
<td>Last name</td>
<td>sn (surname), cn (common name)</td>
</tr>
<tr>
<td>First name</td>
<td>givenName, cn (common name)</td>
</tr>
<tr>
<td>Preferred name</td>
<td>cn (common name)</td>
</tr>
<tr>
<td>Full name</td>
<td>cn (common name)</td>
</tr>
<tr>
<td>User ID</td>
<td>cn (common name)</td>
</tr>
<tr>
<td>Department</td>
<td>departmentNumber</td>
</tr>
<tr>
<td>Job title</td>
<td>title</td>
</tr>
<tr>
<td>Telephone number 1 &amp; 2</td>
<td>telephoneNumber</td>
</tr>
<tr>
<td>FAX telephone number</td>
<td>facsimileTelephoneNumber</td>
</tr>
<tr>
<td>Office</td>
<td>roomNumber</td>
</tr>
<tr>
<td>Address lines 1-4</td>
<td>registeredAddress</td>
</tr>
<tr>
<td>SMTP name</td>
<td>mail</td>
</tr>
</tbody>
</table>

If the field is blank in the system distribution directory, then the attribute is not created in LDAP for that user, with the following exceptions:

- **Last name**: If last name is blank, then the user ID is used in the LDAP directory for the surname (sn) attribute.
- **SMTP name**: When a user has a SMTP name, the SMTP userID (SMTPAUSRID) and SMTP domain (SMTPDMN), or SMTP route (SMTPRTE) is used in the following format: SMTPAUSRID@SMTPDMN or SMTPRTE if they just have a route. For local users, if the SMTP name is blank, then the User ID and address fields are used for the mail attribute in the format 'UserID?Address@Domain'. Domain is the value specified on the Change TCP/IP Domain (CHGTCPDMN) command and the '?' is the default SMTP User ID delimiter value specified on the Change SMTP Attributes (CHGSMTPA) command.

**Distinguished Name (dn) and Common Name (cn)**

LDAP uses the distinguished name (dn) as the key for the user. For the system distribution directory entries in LDAP, the **distinguished name** is the common name (cn) combined with the distinguished name that LDAP is being published to.

The user will have the following common names in LDAP. The first nonblank one will be used in the distinguished name:

1. ‘First name’ ‘Middle Name’ ‘Last name’
2. ‘Preferred name’ ‘Last name’
3. ‘Full name’
4. ‘UserID’

For example, if a user has the following field values in the system distribution directory,

- First name: Jonathan
- Middle name: T.
- Preferred name: John
- Last name: Smith
- Full name: Smith, John T.
• User ID: JSMITH

the user will have the following common names (cn):
• cn=Jonathan T. Smith
• cn=John Smith
• cn=“Smith, John T.”
• cn=JSMITH

If the distinguished name that LDAP is being published to is ‘ou=chicago,o=acme,c=us’, then the distinguished name of this user is ‘cn=Jonathan T. Smith,ou=chicago,o=acme,c=us’ using the first cn in the list. The cn value is enclosed in quotation marks if it contains a comma, pound sign, plus sign, equal sign, less than or greater than sign, or a semicolon. Leading blanks from the system distribution directory fields are removed for the cn value. For example, if the first name is ‘Jane’, the cn value will use ‘Jane’. Also, the system distribution directory field values containing quotation marks will not be used when deriving the cn values as described above.

Attention: If you have two users in the system distribution directory that will resolve to the same distinguished name, they will overlay each other in the LDAP directory. Sometimes overlaying names is what you want if you are merging multiple system distribution directories into one LDAP directory. If you have different users with the same name, ensure they have different distinguished names to prevent overlaying each other.

This API can run on other i5/OS systems to synchronize the system distribution directory on those systems to the same LDAP server and distinguished name being published to. If you have the same user on multiple i5/OS systems, they will become one user in the LDAP directory. The distinguished name (dn) identifies the user. Note that you can run this API from multiple i5/OS systems to different directory servers or to the same directory server, but different distinguished name that LDAP is being published to. You may want to do this if you would like to ensure that information from different system distribution directories does not overlap each other.

User Profile (UID) for i5/OS Users

For local users, the user profile field is used to set the UID attribute in the LDAP directory. This API does not publish passwords for security reasons. Therefore, when the LDAP server is on an i5/OS, the UID attribute is used to see if that user exists on the i5/OS. The password is verified with the password that is passed from the client.

If you are publishing the system distribution directory information to a different i5/OS or to a system that is not an i5/OS, then you will need to set the userPassword attribute for those users that you want to access the LDAP directory. You would set the userPassword attribute for the user after you use the QGLDSSD API to publish the system distribution directory users. The following shows a client command from a UNIX shell that is used to set the userPassword attribute of two users:

```
ldapmodify -h ldapserver -f /path/filename
-D cn=Admin -w password
```

The ldapserver is the server name that was configured in the Directory Services system property. The /path/filename file contains the distinguished name and password for the users. An example file with two user entries would be:

```plaintext
dn:cn=Jonathan T. Smith,ou=chicago,o=acme,c=us
changetype: modify
replace: userPassword
userPassword:secret

dn:cn=Barb Jones,ou=chicago,o=acme,c=us
changetype: modify
replace: userPassword
userPassword:secret
```
Authorities and Locks
*ALLOBJ and *IOSYS_CFG special authority is required to use this API.

Required Parameter Group

Option

INPUT; CHAR(10)

The option to use for publishing system distribution directory information to the LDAP directory. The valid values are:

*ALL    All the local users and all the remote users that have been added from this system and that have an
SMTP name will be replicated from the system distribution directory to the LDAP directory. The LDAP
directory is on the LDAP server specified in the Directory Services dialog of iSeries Navigator. These
users will be placed in the LDAP tree under the distinguished name that is specified in the Directory
Services dialog. See Table 1 for information concerning the system distribution directory fields that will
be used in the LDAP directory.

The *ALL option value also sets up the necessary objects needed to synchronize the system distribution
directory changes to the LDAP directory after the LDAP directory is replicated.

You must request the *ALL option value first, but it can be specified more than once. For example, to
reload the LDAP directory, you would use the *CHG option value to send any pending changes to the
LDAP directory followed by the *ALL option value. If you change which LDAP server or distinguished
name you want the system distribution directory entries to be replicated to, you can use the *ALL
option value to replicate to that server or distinguished name.

*CHG    The system distribution directory entries that were added, changed, removed, or renamed since the
*ALL or previous *CHG option value was used are updated in the LDAP directory.

Changes made to the system distribution directory users in the LDAP directory are overwritten by
changes made in the system distribution directory for the attributes listed above. All other attributes of
inetOrgPerson that are changed in LDAP by using an LDAP client are not overwritten by the *CHG
option value.

*END    End the synchronization of the system distribution directory to LDAP.

If the LDAP user ID is passed in, then this first synchronizes any changes from the system distribution
directory to the LDAP directory since the last synchronization request. For example,

CALL PGM(QSYS/QGLDSODD)
PARM(*END  'LDAPuserID'  'LDAPpassword' 0 0 0)

If the LDAP user ID is not passed in, then the synchronization is just ended and the changes left in the
queue from the last synchronization request are not published. For example,

CALL PGM(QSYS/QGLDSODD)
PARM(*END 0 0 0 0 0)

The users in the LDAP directory where publishing is being ended are not deleted. They are left in the
LDAP directory. Changes made to the system distribution directory after publishing is ended are no
longer queued.

To start replication again after this value is used, call this API with the *ALL option value. A *CHG
option value will result in an error.

*RESET   Ensures that all the objects exist for this replication function and clears the queue that keeps track of the
changes made to the system distribution directory.

Specify zero for the LDAP user ID, LDAP user ID password, key database file, and key database
password when you use this value. For example,

CALL PGM(QSYS/QGLDSODD)
PARM(*RESET 0 0 0 0 0)
LDAP user ID
INPUT; CHAR(1024)

The LDAP user ID that has administrator authority to add, change, and remove entries in the LDAP entry. The valid values are:

*CFG
Use the configured LDAP user ID that can be specified when publishing users (using iSeries Navigator). To use kerberos authentication, you must configure publishing users to authenticate using kerberos. When *CFG is specified for LDAP user ID, then depending on what has been configured to authenticate for users will be used whether that is an administrator ID and password or kerberos.

See "Usage Notes" on page 294 for the procedure of configuring the Directory Services property page. If the Directory Services property page is not configured, and the *CFG value is passed, then error GLD0310 with reason code 12 is signalled. If a value is passed in other than *CFG and kerberos authentication was configured, then error GLD0310 will occur.

A null-terminated string containing the LDAP user ID that has administrator authority to add, change, and remove entries in the LDAP entry.

An example user ID is cn=Admin. Specify a zero-length string if the LDAP server does not require authority checking or the option value *RESET is specified.

LDAP user ID password
INPUT; CHAR(128)

The password for the LDAP user ID. The valid values are:

*CFG
Use the configured LDAP user ID password that can be specified when publishing users (using iSeries Navigator). Specify *CFG if kerberos authentication was configured.

See "Usage Notes" on page 294 for the procedure of configuring the Directory Services property page. If the Directory Services property page is not configured, and the *CFG value is passed, then error GLD0310 with reason code 12 is signalled. If a value is passed in other than *CFG and kerberos authentication was configured, then error GLD0310 will occur.

A null-terminated string containing the password for the LDAP user ID.

Specify a zero-length string if the LDAP server does not require authority checking or the option value *RESET is specified.

No longer used (Formerly 'Key database file')
INPUT; CHAR(1024)

Specify zero (0) as a placeholder for this parameter as it is no longer used. If a value is specified, it will be ignored for compatibility reasons. If you need SSL key database information configured, it is now configured using Digital Certificate Manager. See "Usage Notes" on page 294 below for more information on Digital Certificate Manager.

No longer used (Formerly 'Key database password')
INPUT; CHAR(128)

Specify zero (0) as a placeholder for this parameter as it is no longer used. If a value is specified, it will be ignored for compatibility reasons. If you need SSL key database information configured, it is now configured using Digital Certificate Manager. See "Usage Notes" on page 294 below for more information on Digital Certificate Manager.

Error code
I/O; CHAR(*)

The structure in which to return error information. For the format of the structure, see Error code parameter.
**Note:** All character data is assumed to be represented in the CCSID (coded character set identifier) currently in effect for the job. If the CCSID of the job is 65535, the data is assumed to be represented in the default CCSID of the job.

**Usage Notes**

If the system distribution directory field values for two users result in the same distinguished name, then these names will overlay each other in the LDAP directory. To ensure this does not happen when not intended, you must have unique names for your users before you synchronize the system distribution directory to an LDAP directory.

Use the Convert SMTP Names (CVTNAMSMTP) command if you have not already done so to convert the Simple Mail Transfer Protocol (SMTP) fields to the system distribution directory. The SMTP information is loaded when the option value *ALL is used from this API. If, however, you do not do CVTNAMSMTP when you change the SMTP information using the Work with Names for SMTP (WRKNAMSMTP) command, those changes do not go to the LDAP directory. After you use the CVTNAMSMTP command, the SMTP name is in the system distribution directory in the user-defined fields SMTPAUSTRID SMTP, SMTPDMN SMTP, and SMTPRTE SMTP. When these fields are updated by using the system distribution directory commands (WRKDIRE, ADDDIRE, CHGDIRE), then LDAP is kept synchronized. If you cannot do CVTNAMSMTP, then the other option is to periodically use the option value *ALL to reload the LDAP directory to update all the system distribution directory information including the SMTP information.

**Synchronization Procedure**

A procedure of synchronizing the system distribution directory with an LDAP directory is as follows:

1. The Directory Services property page for the LDAP server to publish must be set up. Use iSeries Navigator, select 'Properties' of the system, and then 'Directory Services'. In V4R4 and later, Directory Services will bring up a list of information to publish. Select 'Users' from this list to configure this information. If your iSeries Navigator or system is prior to V4R4, then just the Directory Services properties are set and no list is displayed. The LDAP server to publish to must be specified and must exist. The distinguished name to publish under must be specified and must be one the server supports. All the users in the system distribution directory will be placed under the distinguished name (DN) that is specified. See the Directory Services (LDAP) topic for more information on using iSeries Navigator to configure the system properties for Directory Services.

Configuring the Directory Services property also can be done using the Change Directory Server Attributes (QgldChgDirSrvA) API.

2. If you are synchronizing the system distribution directory to an LDAP server that is not on an i5/OS, then you need to ensure that the inetOrgPerson and publisher object classes are defined in the schema file for the server. The publisher object class requires a new attribute, publisherName, so be sure publisherName is also defined in a schema file. See SecureWay Directory Schema for documentation on the inetOrgPerson and publisher object class.

3. Ensure the TCP/IP host and domain name are set. Use the Change TCP/IP Domain (CHGTCPDMN) command and prompt by using F4.

4. Use Change SMTP Attribute (CHGSMTPA) command to set the user ID delimiter value. You can keep the default set to ‘?’. Be sure you press Enter so the SMTP attributes are created.

5. If you need SSL certificate information configured, it is configured using Digital Certificate Manager. You can get to Digital Certificate Manager from iSeries Navigator under ‘Network - Internet - Digital ID’.

6. If you are on V4R4 or later, and selected 'Users' in the list when configuring Directory Services property page, then the system distribution directory users will automatically be published to LDAP and you will not need to do the following step. You could optionally call it to reinitialize system distribution directory data to an LDAP server if needed.
Call the Synchronize System Distribution Directory to LDAP API with the *ALL option value. For example, from the command line, type:

```
CALL PGM(QSYS/QGLDSSDD)
    PARM(*ALL 'LDAPuserID' 'LDAPpassword' 0 0)
```

The LDAP user ID must have sufficient authority to add, change, and remove entries in the LDAP directory.

If you have the LDAP user ID and password configured in the Directory Services property page, you can call the API using *CFG. For example, from the command line, type:

```
CALL PGM(QSYS/QGLDSSDD)
    PARM(*ALL *CFG *CFG 0 0)
```

For security reasons, it is recommended that you call this API using the *CFG option if the call is being logged in a job log.

7. If you are on V4R4 or later, and selected 'Users' in the list when configuring Directory Services property page, then the system distribution directory users will automatically be published to LDAP and you will not need to do the following step (although you can optionally call it manually). Periodically call QGLDSSDD to synchronize the LDAP directory with the system distribution directory. The command to synchronize the LDAP directory is:

```
CALL PGM(QSYS/QGLDSSDD)
    PARM(+CHG 'LDAPuserID' 'LDAPpassword' 0 0)
```

If you have the LDAP user ID and password configured in the Directory Services property page, you can call the API using *CFG. For example, from the command line, type:

```
CALL PGM(QSYS/QGLDSSDD)
    PARM(+CHG *CFG *CFG 0 0)
```

For security reasons, it is recommended that you call this API using the *CFG option if the call is being logged in a job log.

The CL program can be run from a job schedule entry to automatically run with scheduled frequency. Use the Add Job Schedule Entry (ADDJOBSCDE) command or the Work with Job Schedule Entries (WRKJOBSCDE) command to automatically schedule jobs.

### Error Messages

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Error Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPF3C90 E</td>
<td>Literal value cannot be changed.</td>
</tr>
<tr>
<td>CPF3CF1 E</td>
<td>Error code parameter not valid.</td>
</tr>
<tr>
<td>GLD0301 E</td>
<td>Error encountered when accessing the LDAP Directory Server.</td>
</tr>
<tr>
<td>GLD0302 E</td>
<td>Input option *CHG currently unavailable.</td>
</tr>
<tr>
<td>GLD0303 E</td>
<td>The caller of this API must have &amp;1 and &amp;2 special authorities.</td>
</tr>
<tr>
<td>GLD0304 E</td>
<td>Unable to export the system distribution directory entry &amp;1 &amp;2 to the LDAP Directory Server.</td>
</tr>
<tr>
<td>GLD0305 C</td>
<td>Synchronization between the system distribution directory and the LDAP directory server completed.</td>
</tr>
<tr>
<td>GLD0309 E</td>
<td>Value not valid for input parameter &amp;1.</td>
</tr>
<tr>
<td>GLD0310 E</td>
<td>Error occurred with QGLDSSDD API. Reason code &amp;1.</td>
</tr>
<tr>
<td>GLD0311 E</td>
<td>Input parameter &amp;1 is not valid. Reason code &amp;2.</td>
</tr>
<tr>
<td>GLD0312 D</td>
<td>Error encountered when setting up a secure connection to an LDAP server. The error number is &amp;1.</td>
</tr>
</tbody>
</table>

API introduced: V4R3

---

**Concepts**

These are the concepts for this category.
LDAP API Overview

Lightweight Directory Access Protocol (LDAP) is an Internet protocol to access directory servers. The directories on the Internet may be “pure” LDAP directories; that is, they only communicate through LDAP, or they may be X.500 or other types of servers that allow access through LDAP. Access to servers that are not pure LDAP servers is accomplished through an LDAP gateway. Gateways from LDAP to other protocols also are common. Client programs that allow a user to access an LDAP directory are called LDAP clients. Applications that extract information from an LDAP directory are referred to as LDAP-enabled.

The LDAP client is part of the i5/OS(TM). The LDAP client is used by i5/OS and customer applications for access to LDAP-enabled directories in the network. The directories being accessed may or may not be located on an i5/OS server. The applications access the LDAP client by using these client APIs. TCP/IP is always used to access remote directories, and the administrator can configure the connection to use the Secure Sockets Layer (SSL). Also, the administrator can select to use Kerberos.

The LDAP APIs are designed to provide a suite of functions that can be used to develop directory enabled applications. Directory-enabled applications typically connect to one or more directories and perform various directory-related operations, such as:

- Adding entries
- Searching the directory and obtaining the resulting list of entries
- Deleting entries
- Modifying entries
- Renaming entries

The type of information that is managed in the directory depends on the nature of the application. Directories are often used to provide public access to information about people, including:

- Phone numbers
- E-mail addresses
- Fax numbers
- Mailing addresses

Increasingly, directories are being used to manage and publish other types of information, including:

- Configuration information
- Public key certificates (managed by certification authorities)
- Access control information
- Locating information (how to find a service)

The LDAP APIs provide for both synchronous and asynchronous access to a directory. Asynchronous access makes it easy for your application to do other work while waiting for the results of a potentially lengthy directory operation to be returned by the server.

Typical API Usage

The basic interaction is as follows. A connection is made to an LDAP server by calling ldap_init (or ldap_ssl_init, which is used to establish a secure connection over Secure Sockets Layer (SSL)).

An LDAP bind operation is performed by calling ldap_simple_bind or ldap_sasl_bind. The bind operation is used to authenticate to the directory server. Note that the LDAP V3 API and protocol permits the bind to be skipped, in which case the access rights associated with anonymous access are obtained.

Next, other operations are performed by calling one of the synchronous or asynchronous routines (that is, ldap_search_s or ldap_search followed by ldap_result).
Results returned from these routines are interpreted by calling the LDAP parsing routines, which include operations such as:

- `ldap_first_entry`, `ldap_next_entry`
- `ldap_get_dn`
- `ldap_first_attribute`, `ldap_next_attribute`
- `ldap_get_values`
- `ldap_parse_result` (new for LDAP V3)
- `etc.`

The LDAP connection is terminated by calling `ldap_unbind`.

The `ldap_set_rebind_proc` routine can be used to define the entry-point of a routine to be called when an LDAP bind operation needs to occur when handling a client referral to another server.

### Displaying Results

Results obtained from the `ldap_search` routines can be accessed by calling `ldap_first_entry` and `ldap_next_entry` to step through the entries returned, `ldap_first_attribute` and `ldap_next_attribute` to step through an entry's attributes, `ldap_get_values` to retrieve a given attribute's value, and then calling `printf` or some other display or usage method to display the values.

### Uniform Resource Locators (URLs)

The `ldap_is_ldap_url` routines can be used to test a URL to see if it is an LDAP URL, to parse LDAP URLs into their component pieces, and to initiate searches directly using an LDAP URL.

Examples of these routines are `ldap_url_parse`, `ldap_url_search_s`, and `ldap_is_ldap_url`.

### Secure Socket Layer (SSL) Support

The LDAP APIs have been extended to support connections that are protected by the Secure Socket Layer (SSL) protocol. This can be used to provide strong authentication between the client and server, as well as data encryption of LDAP messages that flow between the client and the LDAP server. The `ldap_ssl_client_init()` and `ldap_ssl_init()` APIs are provided to initialize the SSL function, and to create a secure SSL connection (respectively).

When using [“ldap_ssl_client_init —Initialize the SSL Library.” on page 199](#), the application ID used is `QIBM_GLD_DIRSRV_CLIENT`, identified as client application “Directory Services Client” in Digital Certificate Manager (DCM). To use i5/OS application IDs other than the default which have an association to a certificate store and a particular certificate in that store, the following i5/OS-specific APIs are provided:

Version 2 API
- `ldap_app_ssl_start_np()` (deprecated)

Version 3 API
- `ldap_app_ssl_client_init_np()`

When using `ldap_ssl_init()`, the server is not contacted until the connection is used; that is, by `ldap_bind()` or `ldap_search()`. If an SSL error occurs while trying to connect, the SSL error code can be retrieved for the connection with the [“ldap_get_option()—Retrieve LDAP Options” on page 96](#) API using the `LDAP_OPT_EXT_ERROR` option.
LDAP Version Support

The LDAP toolkit has been enhanced to support both LDAP Version 2 and LDAP Version 3 APIs and protocols. The LDAP toolkit APIs and protocols are based on the Internet Draft, which is classified as a "work in progress."

The LDAP APIs provide typical directory functions such as read, write, and search. With the advent of support for LDAP Version 3 APIs and protocols, the following features are also supported:

- LDAP V3 referrals
- Improved internationalization with UTF-8 support for Distinguished Names (DNs) and strings that are passed into, and returned from the LDAP APIs (when running as an LDAP V3 application and LDAP_OPT_UTF8_IO is set to LDAP_UTF8_XLATE_OFF). The default, when running as an LDAP V3 or V2 application, for DNs and strings that are passed into or returned from LDAP APIs is limited to the local codepage character set.

In general, the connection-associated LDAP Version 3 APIs (APIs that have ld as one of their parameters) are designed to accept and return string data in either UTF-8 encoded format or in the local code page format, depending on the LDAP_OPT_UTF8_IO option value set using the ldap_set_option() API to LDAP_UTF8_XLATE_ON (the default) or LDAP_UTF8_XLATE_OFF.

The following LDAP APIs (and related APIs) accept and return UTF-8 encoded string data when the LDAP_OPT_UTF8_IO option is set to LDAP_UTF8_XLATE_OFF. Otherwise, they accept or return string data in the local code page (the default).

- ldap_add (and family)
- ldap_bind (and family)
- ldap_compare (and family)
- ldap_delete (and family)
- ldap_parse_reference_np
- ldap_get_dn
- ldap_get_values
- ldap_modify (and family)
- ldap_parse_result
- ldap_rename (and family)
- ldap_search (and family)
- ldap_url_search (and family)

APIs that are NOT associated with a connection (APIs that do not have ld as one of their parameters), always expect and return string data (DNs, for example) in local code page.

The following LDAP APIs (and related APIs) will accept and return string data in the local code page.

- ldap_init
- ldap_ssl_init
- ldap_explode_dn
- ldap_explode_rdn
- ldap_server_locate
- ldap_server_conf_save
- ldap_is_ldap_url
- ldap_default_dn_set/get

As a non-standard extension to the API set on i5/OS™ only, two APIs have been added that allow input of string data in UTF8. These are:

- ldap_explode_dn_utf8
- ldap_explode_rdn_utf8
• The ability for an application to access schema information published by LDAP V3 servers (see Accessing Schema Information).

• The ability for certain LDAP Version 3 operations to be extended with the use of controls. Controls can be sent to a server, or returned to the client with any LDAP message. This type of control is called a server control.

The LDAP API also supports a client-side extension mechanism, which can be used to define client controls. The client-side controls affect the behavior of the LDAP client library, and are never sent to the server. Note that client-side controls are not defined for this client library.

A common data structure is used to represent both server-side and client-side controls:

```c
typedef struct ldapcontrol {
    char *ldctl_oid;
    struct berval ldctl_value;
    char ldctl_iscritical;
} LDAPControl, *PLDAPControl;
```

The LDAPControl fields have the following definitions:

`ldctl_oid`
- The control type, represented as a string.

`ldctl_value`
- The data associated with the control. The control may not include data.

`ldctl_iscritical`
- Whether the control is critical or not. If the field is non-zero, the operation is carried out only if it is recognized and supported by the server (or the client for client-side controls).

If using any of the ber_xxx functions to set up the berval structure, you must specify QSYS/QGLDBRDR as one of the the bind service programs when creating the program.

With this toolkit, an application that uses the ldap_open API defaults to the LDAP V2 protocol. In this way, existing LDAP applications will continue to work, and can interoperate with both LDAP V2 servers and LDAP V3 servers.

An application that uses the ldap_init API defaults to the LDAP V3 protocol (with optional bind). An LDAP V3 application will not necessarily interoperate with an LDAP server that supports only LDAP V2 protocols.

An application can use the ldap_set_option API to change its LDAP protocol version. This should be done after using ldap_open or ldap_init but before issuing a bind or other operation that results in contacting the server.

### Accessing Schema Information

LDAP V3 servers permit applications to access schema and other related information. For example, the ldapsearch utility can be used to obtain the subschemasubentry, attributeTypes, and objectClasses from IBM(R) SecureWay(R) Directory Server. First use ldapsearch to get the root DSE to find the entry containing the schema (called the subschemasubentry) for the server, as follows:

```
ldapsearch -V 3 -h hostname -p port
   -s base -b "cn=schema" "objectClass=*" subschemaSubentry
```

The subschemasubentry on SecureWay directories is cn=schema by default. To retrieve the schema itself, search on the subschemasubentry entry, as follows:

```
ldapsearch -V 3 -h hostname -p port
   -s base -b "cn=schema" "objectClass=*"
```
The "-V 3" option is used to force ldapsearch to bind as an LDAP V3 application.

API Prototype Changes

For many of the LDAP APIs, the prototype has changed. On many of the API prototypes where a "char *" is used, the prototype has changed to use a "const char *". This is the result of changes to the standards. i5/OS™ is providing a way to transition to the new prototypes. Inserting

#define _QGLDNOCONST

in applications code prior to the include of ldap.h causes the definition of the old prototypes that use "char *" to be made available. If _QGLDNOCONST, which is the default, is not defined, the definition of the new prototypes that use "const char *" is made available.

In some future release, the use of _QGLDNOCONST will be withdrawn.

Deprecated APIs

The following is a list of APIs that are still supported, although their use is deprecated. Use of the newer replacement APIs is strongly encouraged.

- ldap_ssl_start() - use ldap_ssl_client_init() and ldap_ssl_init()
- ldap_open() - use ldap_init()
- ldap_bind() - use ldap_simple_bind()
- ldap_bind_s() - use ldap_simple_bind_s()
- ldap_modrdn() - use ldap_rename()
- ldap_modrdn_s() - use ldap_rename_s()
- ldap_result2error() - use ldap_parse_result()
- ldap_perror() - use ldap_parse_result()

i5/OS™-specific APIs:
- ldap_app_ssl_start_np() - use ldap_app_ssl_client_init_np() and ldap_app_ssl_init_np().

LDAP Client API Error Conditions

When most LDAP APIs fail to complete successfully, ld_errno usually indicates one of the following errors. Under some conditions, ld_errno could indicate an error other than those listed here.

- LDAP_SUCCESS 0x00 - The request was successful.
- LDAP_OPERATIONS_ERROR 0x01 - An operations error occurred.
- LDAP_PROTOCOL_ERROR 0x02 - A protocol violation was detected.
- LDAP_TIMELIMIT_EXCEEDED 0x03 - An LDAP time limit was exceeded.
- LDAP_SIZELIMIT_EXCEEDED 0x04 - An LDAP size limit was exceeded.
- LDAP_COMPARE_FALSE 0x05 - A compare operation returned false.
- LDAP_COMPARE_TRUE 0x06 - A compare operation returned true.
- LDAP_STRONG_AUTH_NOT_SUPPORTED 0x07 - The LDAP server does not support strong authentication.
<table>
<thead>
<tr>
<th>LDAP Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAP_STRONG_AUTH_REQUIRED</td>
<td>0x08 - Strong authentication is required for the operation.</td>
</tr>
<tr>
<td>LDAP_PARTIAL_RESULTS</td>
<td>0x09 - Partial results only returned.</td>
</tr>
<tr>
<td>LDAP_REFERRAL</td>
<td>0x0A - Referral returned.</td>
</tr>
<tr>
<td>LDAP_ADMIN_LIMIT_EXCEEDED</td>
<td>0x0B - Administration limit exceeded.</td>
</tr>
<tr>
<td>LDAP_UNAVAILABLE_CRITICAL_EXTENSION</td>
<td>0x0C - Critical extension not supported.</td>
</tr>
<tr>
<td>LDAP_NO_SUCH_ATTRIBUTE</td>
<td>0x10 - The attribute type specified does not exist in the entry.</td>
</tr>
<tr>
<td>LDAP_UNDEFINED_TYPE</td>
<td>0x11 - The attribute type specified is not valid.</td>
</tr>
<tr>
<td>LDAP_INAPPROPRIATE_MATCHING</td>
<td>0x12 - Filter type not supported for the specified attribute.</td>
</tr>
<tr>
<td>LDAP_CONSTRAINT_VIOLATION</td>
<td>0x13 - An attribute value specified violates some constraint (for example, a postal address has too many lines, or a line that is too long).</td>
</tr>
<tr>
<td>LDAP_TYPE_OR_VALUE_EXISTS</td>
<td>0x14 - An attribute type or attribute value specified already exists in the entry.</td>
</tr>
<tr>
<td>LDAP_INVALID_SYNTAX</td>
<td>0x15 - An attribute value was specified that is not valid.</td>
</tr>
<tr>
<td>LDAP_NO_SUCH_OBJECT</td>
<td>0x20 - The specified object does not exist in the directory.</td>
</tr>
<tr>
<td>LDAP_ALIAS_PROBLEM</td>
<td>0x21 - An alias in the directory points to a nonexistent entry.</td>
</tr>
<tr>
<td>LDAP_INVALID_DN_SYNTAX</td>
<td>0x22 - A distinguished name was specified that is syntactically not valid.</td>
</tr>
<tr>
<td>LDAP_IS_LEAF</td>
<td>0x23 - The object specified is a leaf.</td>
</tr>
<tr>
<td>LDAP_ALIAS_DEREF_PROBLEM</td>
<td>0x24 - A problem was encountered when dereferencing an alias.</td>
</tr>
<tr>
<td>LDAP_INAPPROPRIATE_AUTH</td>
<td>0x30 - Inappropriate authentication was specified (for example, LDAP_AUTH_SIMPLE was specified and the entry does not have a user password attribute).</td>
</tr>
<tr>
<td>LDAP_INVALID CREDENTIALS</td>
<td>0x31 - Credentials that are not valid were presented (for example, the wrong password).</td>
</tr>
<tr>
<td>LDAP_INSUFFICIENT_ACCESS</td>
<td>0x32 - The user has insufficient access to perform the operation.</td>
</tr>
<tr>
<td>LDAP_BUSY</td>
<td>0x33 - The directory system agent is busy.</td>
</tr>
<tr>
<td>LDAP_UNAVAILABLE</td>
<td>0x34 - The directory system agent is unavailable.</td>
</tr>
<tr>
<td>LDAP_UNWILLING_TO_PERFORM</td>
<td>0x35 - The directory system agent is unwilling to perform the operation.</td>
</tr>
<tr>
<td>LDAP_LOOP_DETECT</td>
<td>0x36 - A loop was detected.</td>
</tr>
<tr>
<td>LDAP_NAMING_VIOLATION</td>
<td>0x40 - A naming violation occurred.</td>
</tr>
<tr>
<td>LDAP_OBJECT_CLASS_VIOLATION</td>
<td>0x41 - An object class violation occurred (for example, a must attribute was missing from the entry).</td>
</tr>
<tr>
<td>LDAP_NOT_ALLOWED_ON_NONLEAF</td>
<td>0x42 - The operation is not allowed on a nonleaf object.</td>
</tr>
<tr>
<td>LDAP_NOT_ALLOWED_ON_RDN</td>
<td>0x43 - The operation is not allowed on a relative distinguished name.</td>
</tr>
<tr>
<td>LDAP_ALREADY_EXISTS</td>
<td>0x44 - The entry already exists.</td>
</tr>
<tr>
<td>LDAP_NO_OBJECT_CLASS_MODS</td>
<td>0x45 - Object class modifications are not allowed.</td>
</tr>
<tr>
<td>LDAP_RESULTS_TOO_LARGE</td>
<td>0x46 - Results too large.</td>
</tr>
<tr>
<td>LDAP_AFFECTS_MULTIPLE_DSAS</td>
<td>0x47 - Affects multiple DSAS.</td>
</tr>
<tr>
<td>LDAP_OTHER</td>
<td>0x50 - An unknown error occurred.</td>
</tr>
<tr>
<td>LDAP_SERVER_DOWN</td>
<td>0x51 - The LDAP API cannot contact the LDAP server.</td>
</tr>
<tr>
<td>LDAP_LOCAL_ERROR</td>
<td>0x52 - Some local error occurred.</td>
</tr>
<tr>
<td>LDAP_ENCODING_ERROR</td>
<td>0x53 - An error was encountered while the API was encoding parameters to send to the LDAP server.</td>
</tr>
<tr>
<td>LDAP_DECODING_ERROR</td>
<td>0x54 - An error was encountered while the API was decoding a result from the LDAP server.</td>
</tr>
<tr>
<td>LDAP_TIMEOUT</td>
<td>0x55 - A time limit was exceeded while API was waiting for a result.</td>
</tr>
<tr>
<td>LDAP_AUTH_LINK_UNKNOWN</td>
<td>0x56 - The authentication method specified to ldap_bind() is not known.</td>
</tr>
<tr>
<td>LDAP_FILTER_ERROR</td>
<td>0x57 - A filter that is not valid was supplied to ldap_search() (for example, unbalanced parentheses).</td>
</tr>
<tr>
<td>LDAP_USER_CANCELLED</td>
<td>0x58 - User cancelled</td>
</tr>
<tr>
<td>LDAP_PARAM_ERROR</td>
<td>0x59 - An LDAP API was called with a bad parameter (for example, a NULL Id pointer).</td>
</tr>
<tr>
<td>LDAP_NO_MEMORY</td>
<td>0x5A - A memory allocation (for example, a malloc() call) failed in an LDAP API.</td>
</tr>
<tr>
<td>LDAP_CONNECT_ERROR</td>
<td>0x5B - Connection error</td>
</tr>
</tbody>
</table>
LDAP_NOT_SUPPORTED 0x5c - Not Supported
LDAP_CONTROL_NOT_FOUND 0x5d - Control not found
LDAP_NO_RESULTS_RETURNED 0x5e - No results returned
LDAP_MORE_RESULTS_TO_RETURN 0x5f - More results to return
LDAP_URL_ERR_NOTLDAP 0x60 - URL doesn’t begin with ldap://
LDAP_URL_ERR_NODN 0x61 - URL has no DN (required).
LDAP_URL_ERR_BADSIG 0x62 - URL scope string is invalid.
LDAP_URL_ERR_MEM 0x63 - Can’t allocate memory space.
LDAP_CLIENT_LOOP 0x64 - Client loop
LDAP_REFERRAL_LIMIT_EXCEEDED 0x65 - Referral limit exceeded
LDAP_SSL_ALREADY_INITIALIZED 0x70 - ldap_ssl_client_init successfully called previously in this process.
LDAP_SSL_INITIALIZE_FAILED 0x71 - SSL initialization call failed.
LDAP_SSL_INITIALIZE_NOT_CALLED 0x72 - Call ldap_ssl_client_init before attempting to use an ssl connection.
LDAP_SSL_PARAM_ERROR 0x73 - An invalid ssl parameter was previously specified.
LDAP_SSL_HANDSHAKE_FAILED 0x74 - Failed to connect to ssl server.
LDAP_SSL_GET_CIPHER_FAILED 0x75 - Failed to identify the maximum SSL encryption level for this host.
LDAP_SSL_KEYRING_NOT_FOUND 0x76 - SSL Keyring file not found
LDAP_SSL_PASSWORD_NOT_SPECIFIED 0x77 - SSL password not specified
LDAP_NO_EXPLICIT_OWNER 0x78 - No explicit owner found
LDAP_NO_LOCK 0x79 - Could not obtain lock
LDAP_DNS_NO_SERVERS 0x80 - No LDAP servers found
LDAP_DNS_TRUNCATED 0x81 - Warning truncated DNS results
LDAP_DNS_INVAILD_DATA 0x82 - Invalid DNS Data
LDAP_DNS_RESOLVE_ERROR 0x83 - Can’t resolve system domain or nameserver
LDAP_DNS_CONF_FILE_ERROR 0x84 - DNS Configuration file error
LDAP_XLATE_E2BIG 0x85 - Output buffer overflow
LDAP_XLATE_EINVAL 0x86 - Input buffer truncated
LDAP_XLATE_EILSEQ 0x87 - Unusable input character
LDAP_XLATE_NO_ENTRY 0x88 - No codeset point to map to
LDAP_REG_FILE_NOT_FOUND 0x89 - NT Registry - file not found
LDAP_REG_CANNOT_OPEN 0x8A - NT Registry - cannot open
LDAP_REG_ENTRY_NOT_FOUND 0x8B - NT Registry entry not found
LDAP_CONF_FILE_NOT_OPENED 0x8C - Plugin configuration file not opened
LDAP_PLUGIN_NOT_LOADED 0x8D - Plugin library not loaded
LDAP_PLUGIN_FUNCTION_NOT_RESOLVED 0x8E - Plugin function not resolved
LDAP_PLUGIN_NOT_INITIALIZED 0x8F - Plugin library not initialized
LDAP_PLUGIN_COULD_NOT_BIND 0x90 - Plugin function could not bind
LDAP_SASL_GSS_NO_SEC_CONTEXT 0x91 - gss_init_sec_context failed
Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:
IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:
IBM Corporation
Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, IBM License Agreement for Machine Code, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

(C) IBM 2006. Portions of this code are derived from IBM Corp. Sample Programs. (C) Copyright IBM Corp. 1998, 2006. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

### Programming Interface Information

This Application Programming Interfaces (API) publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of IBM i5/OS.
Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

Advanced 36
Advanced Function Printing
Advanced Peer-to-Peer Networking
AFP
AIX
AS/400
COBOL/400
CUA
DB2
DB2 Universal Database
 Distributed Relational Database Architecture
Domino
DPI
DRDA
eServer
GDDM
IBM
Integrated Language Environment
Intelligent Printer Data Stream
IPDS
i5/OS
iSeries
Lotus Notes
MVS
Netfinity
Net.Data
NetView
Notes
OfficeVision
Operating System/2
Operating System/400
OS/2
OS/400
PartnerWorld
PowerPC
PrintManager
Print Services Facility
RISC System/6000
RPG/400
RS/6000
SAA
SecureWay
System/36
System/370
System/38
System/390
VisualAge
WebSphere
xSeries

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

---

**Terms and Conditions**

Permissions for the use of these Publications is granted subject to the following terms and conditions.

**Personal Use:** You may reproduce these Publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these Publications, or any portion thereof, without the express consent of IBM. You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations. IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.