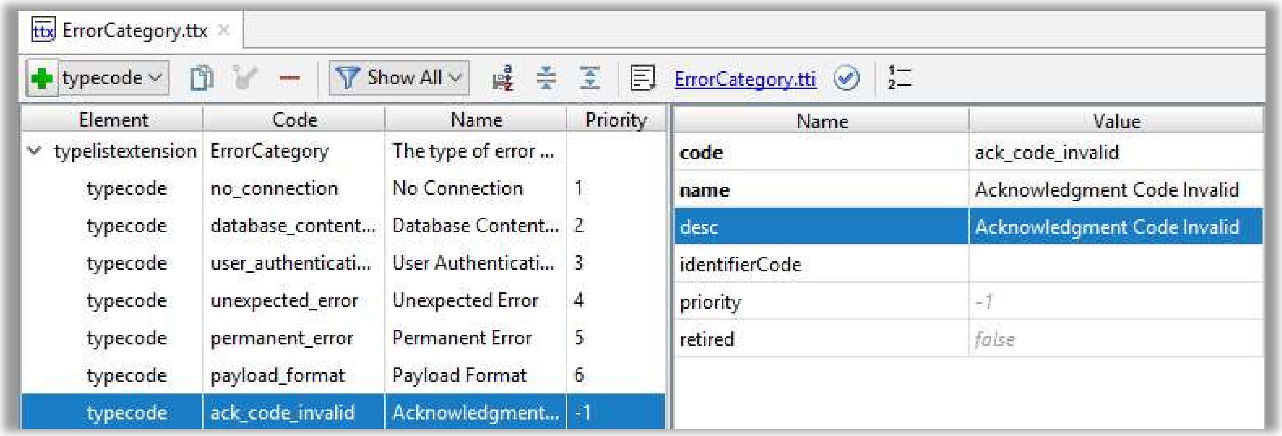
Solution 1: Configure synchronous acknowledgment

1. **Configure the ErrorCategory typelist.**

   a) Open **ErrorCategory** typelist extension.

   b) Add new typecode.



Element	Code	Name	Priority
typelistextension	ErrorCategory	The type of error ...	
typecode	no_connection	No Connection	1
typecode	database_content...	Database Content...	2
typecode	user_authenticati...	User Authenticati...	3
typecode	unexpected_error	Unexpected Error	4
typecode	permanent_error	Permanent Error	5
typecode	payload_format	Payload Format	6
typecode	ack_code_invalid	Acknowledgment...	-1

Name	Value
code	ack_code_invalid
name	Acknowledgment Code Invalid
desc	Acknowledgment Code Invalid
identifierCode	
priority	*-1*
retired	*false*

2. **Configure the FraudCheckTransport class.**

```
package si.ta.messaging.fraudcheck

uses gw.plugin.InitializablePlugin
uses gw.plugin.messaging.MessageTransport
uses si.ta.messaging.fraudcheck.fraudcheckwsc.fraudreportapi.FraudReportAPI

class FraudCheckTransport implements MessageTransport, InitializablePlugin{
  // Plugin parameters
  private static final var USERNAME = "Username"
  private static final var PASSWORD = "Password"
  // API parameters
  private static var _username : String
  private static var _password : String

  private var _fraudReportApi: FraudReportAPI

  override function send(message : Message, transformedPayload : String) {
    var responseCode = _fraudReportApi.checkForFraudReport(transformedPayload)

    // Acknowledge message based on the response code
    var ackMessage = processResponseCode(message, responseCode)
    print(ackMessage)
  }

  override function shutdown() {  }
  override function suspend() {
    _fraudReportApi = null
  }

  override function resume() {
    _fraudReportApi = new FraudReportAPI()
    _fraudReportApi.Config.Http.Authentication.Basic.Username = _username
    _fraudReportApi.Config.Http.Authentication.Basic.Password = _password
  }

  override property set DestinationID(destinationID : int) {  }

  override property set Parameters(map : Map<Object, Object>) {
    _username = map.get(USERNAME) as String
    _password = map.get(PASSWORD) as String
  }

  // Helper method

  /** This method processes a message based on its external system synchronous response code.
   */
  @Param("aMessage", "The message instance")
  @Param("responseCode", "Response code from the external system")
  private function processResponseCode(message: Message, responseCode: Integer) : String {
    var output = ""

    // Acknowledge message based on response code
    switch (responseCode) {
      case 1:
        message.reportAck()
        output =("Request processed, no fraud report found.")
        break
      case 2:
        message.reportAck()
        output = ("Request processed, fraud report found!")
        break
      case 4:
        message.reportError(ErrorCategory.TC_PAYLOAD_FORMAT)
        output = ("Request could not be processed (Payload Format Error)")
        break
      case 5:
        message.reportError(ErrorCategory.TC_DATABASE_CONTENTION)
        output = ("Request could not be processed (Database Unavailable)")
        break
      default:
        message.reportError(ErrorCategory.TC_ACK_CODE_INVALID)
        output = ("Request could not be processed (Acknowledgment Code Invalid)")
        break
    }
    return output
  }

}
```

3. **Deploy code changes.**

   a) From the **Studio** menu, **Restart** the server.

4. **Perform verification steps.**