**BLACKBOARD GRADEBOOK WEB SERVICE INTEGRATION**



CONTENTS

[SCOPE 3](#_Toc402211430)

[RESOURCES 3](#_Toc402211431)

[SOAP WEB SERVICE LOCATION AND WSDL 4](#_Toc402211432)

[PRODUCTION ENVIRONMENT 4](#_Toc402211433)

[TEST ENVIRONMENT 4](#_Toc402211434)

[ORCHESTRATION SCHEME AND SERVICE ACTIVITY 5](#_Toc402211435)

[SERVICE ACTIVITY DIAGRAM 5](#_Toc402211436)

[SERVICE ACTIVITY DESCRIPTION 5](#_Toc402211437)

[START 5](#_Toc402211438)

[CONTEXT.WS 6](#_Toc402211439)

[COURSE.WS 7](#_Toc402211440)

[COURSEMEMBERSHIP.WS 8](#_Toc402211441)

[USER.WS 8](#_Toc402211442)

[GRADEBOOK.WS 9](#_Toc402211443)

[GRADEBOOK REPORT 10](#_Toc402211444)

[ERROR HANDLING 10](#_Toc402211445)

[BLACKBOARD LEARN WS SECURITY CONFIGURATION 11](#_Toc402211446)

[FTP SERVER LOCATION AND CREDENTIALS 11](#_Toc402211447)

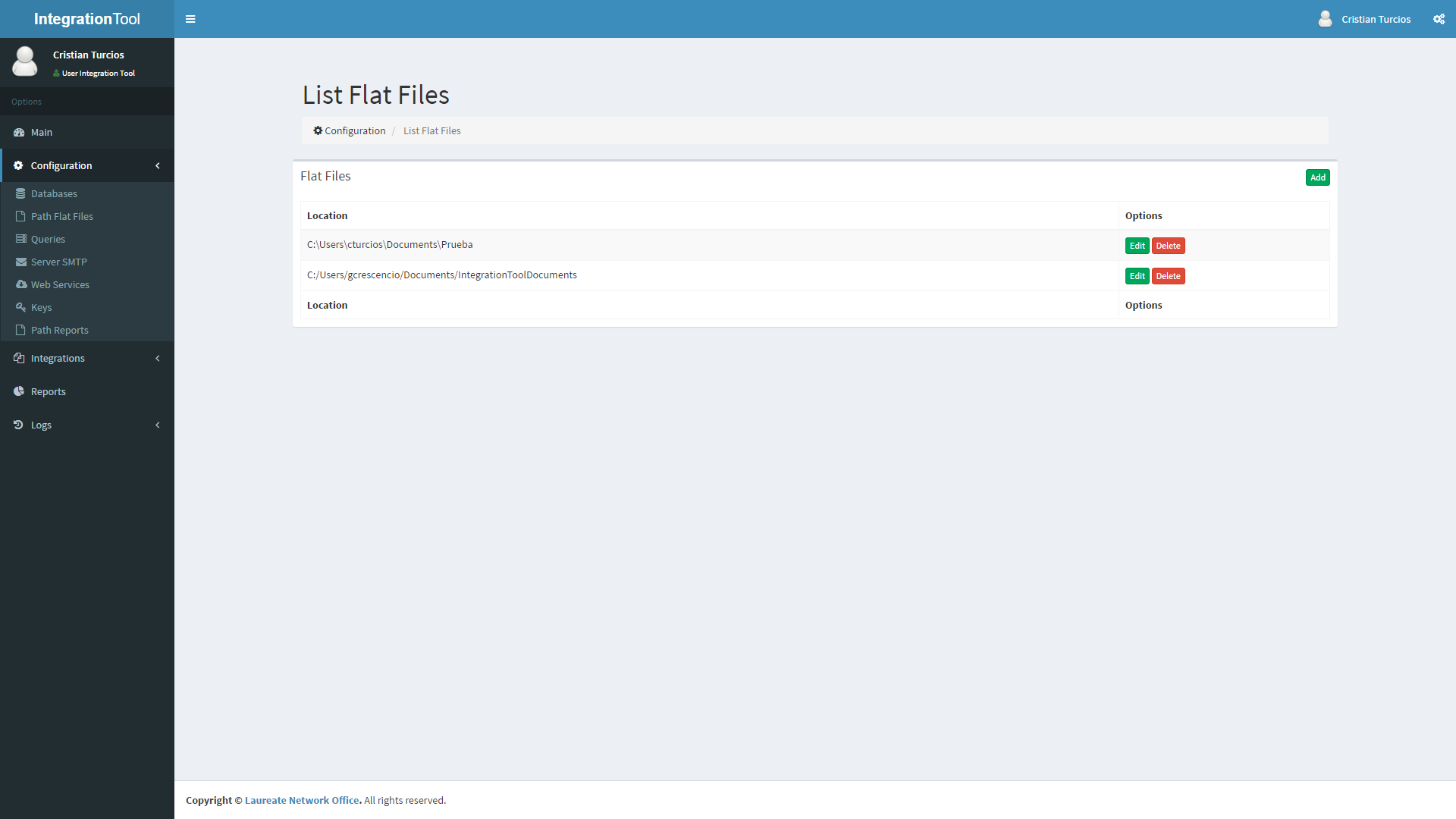
[GRADEBOOK REPORT FORMAT 12](#_Toc402211448)

# SCOPE

El siguiente documento detalla la funcionalidad y especificaciones técnicas que se requieren para realizar un proceso de integración entre distintos servidores de bases de datos ya sea MySql o SQL Server hacia una plataforma BlackBoard

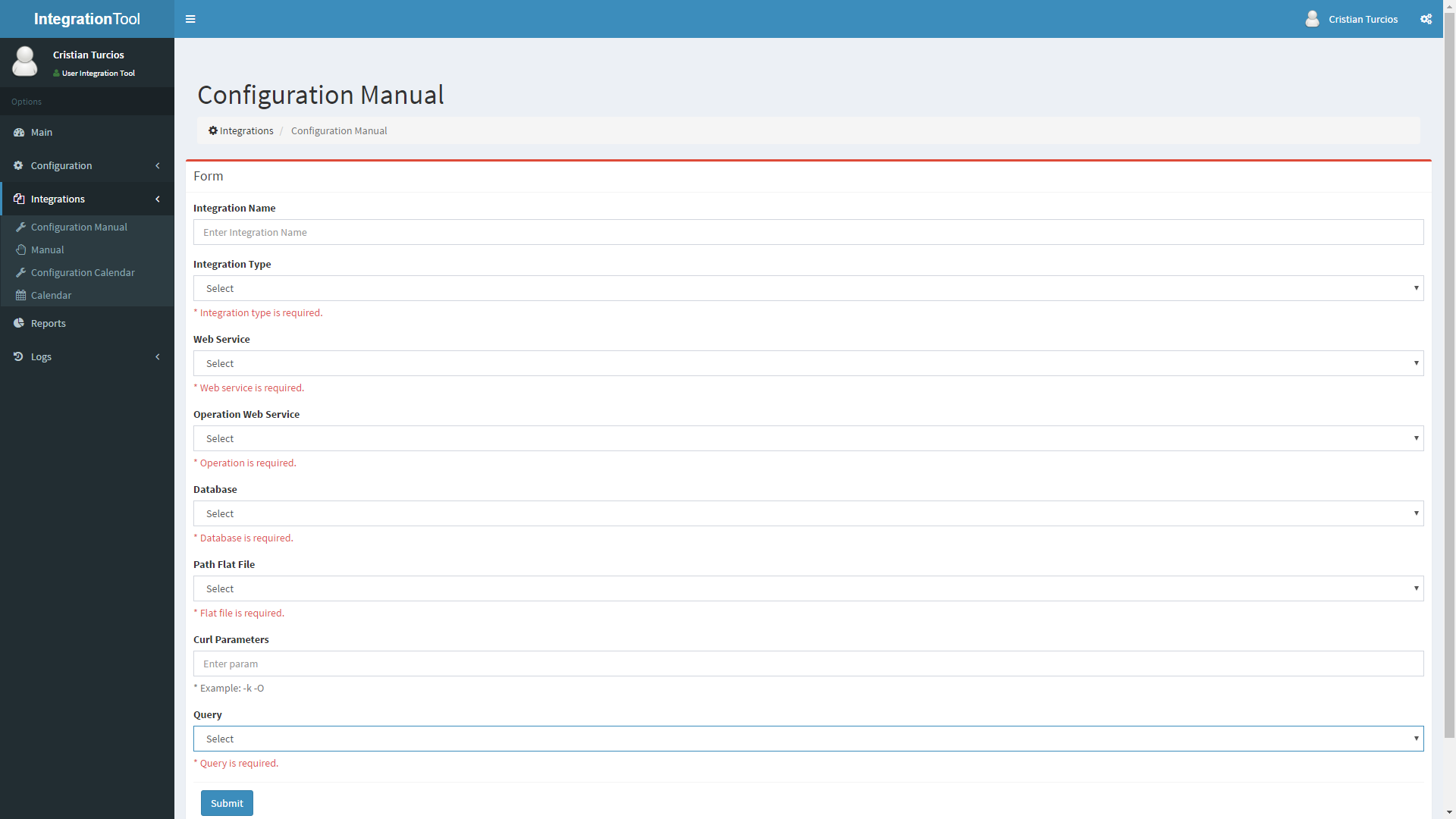
Se detalla el orden necesario para crear una integración:

* **Configuration - Databases:** (Bases de datos a las que se conectara el sistema y sacara la data necesaria para hacer la integración).
* **Configuration – Path Flat Files:** (Ubicación donde se guardarán los archivos generados por el Sistema en el servidor)
* **Configuration – Queries:** (distintos queries que se ejecutaran en la base de datos para sacar la información de esta misma)
* **Configuration – Server SMTP:** (credenciales de servidor SMTP que se usaran para el envió de correos en toda la aplicación)
* **Configuration – Web Services:** (distintos webs Services a los cuales se conectará la aplicación para poder realizar una integración)
* **Configuration -Keys:** (Clave de encriptación de los datos)
* **Configuration – Path Reports:** (Ubicación donde se guardarán los archivos generados por el Sistema en el servidor)



# RESOURCES

Todas las configuraciones anteriores son necesarias para poder mostrar los datos al realizar una integración manual de forma fluida.



* **Se detallan a continuación los elementos necesarios para poder generar una integración manual**:
  + **Integration Name**: Se debe especificar un nombre a la integración para poder identificarla en el Sistema.
  + **Integration Type**: Se debe especificar el tipo de integración que se hará, ya sea una integración de cursos, persona, organización, etc.
  + **Web Services**: Se especifica el web service al cual se conectará el Sistema para realizar la integracion.
  + **Operation Web Service**: Se debe especificar el tipo de operación que hará el web Services, ya sea Store, Refresh, Delete, etc.
  + **Databases:** Se debe seleccionar la base de datos a la cual el Sistema se conectará y sacará la data para realizar la integracion.
  + **Path Flat File:** Se especifica una de las distintas ubicaciones donde se guardarán los archivos en el servidor.
  + **Query:** Se especifica el query que se va a ejecutar en la base de datos para sacar la información y realizar la integracion hacia BlackBoard.
  + **[[LIST-IDS]]:** Parámetros que se introducirán en el query para poder realizar la consulta a la base de datos.
* **Se detallan a continuación los elementos necesarios para poder crear una integración Calendarizada**:

**Para poder crear una integracion calendarizada en su mayoría se utilizan los mismos parámetros que una integracion manual, se añaden únicamente nuevos parámetros como ser:**

* **Execution Start Date:** Se debe especificar la fecha de inicio de la integracion
* **Execution End Date:** Se debe especificar la fecha de fin de la integracion
* **Recurrence:** Se debe especificar la recurrencia que tendrá esta integracion, puede ser de una hora, un día, una semana, o un mes.
* **Emails:** si se desea tener un recordatorio por email cada vez que se genera una integracion calendarizada se debe poner uno o varios emails separados por coma para que genere una notificación por correo.



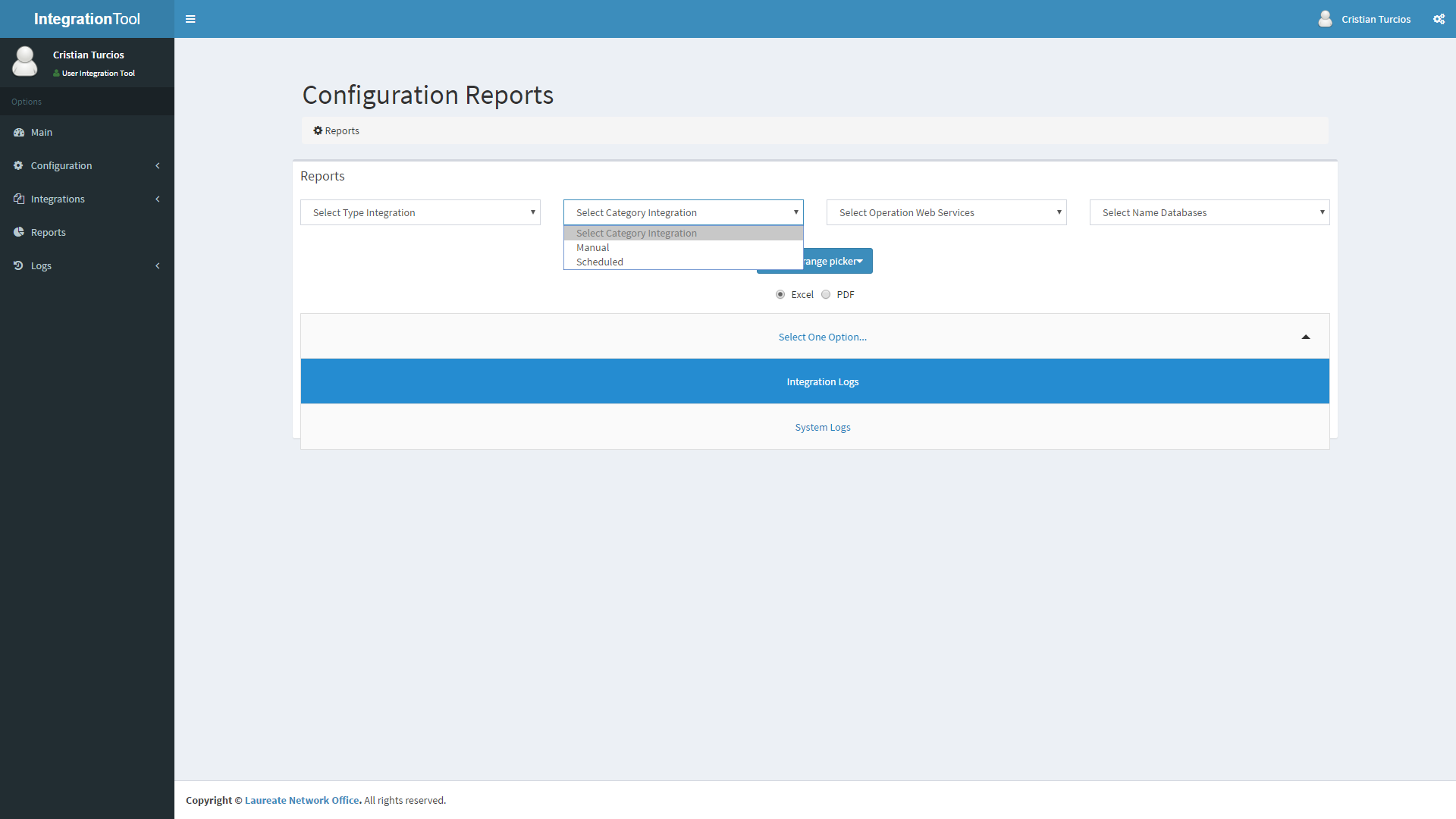
# SOAP WEB SERVICE LOCATION AND WSDL

## REPORTS

El Sistema posee la facilidad de generar reportes tanto en formato Excel, como en formato PDF.

## 

Los reportes pueden generarse ya sea por todos los logs que se encuentren en la base de datos o pueden generarse reportes filtrados por los datos mostrados en pantalla, incluso indicando fechas de inicio y fechas de finalización



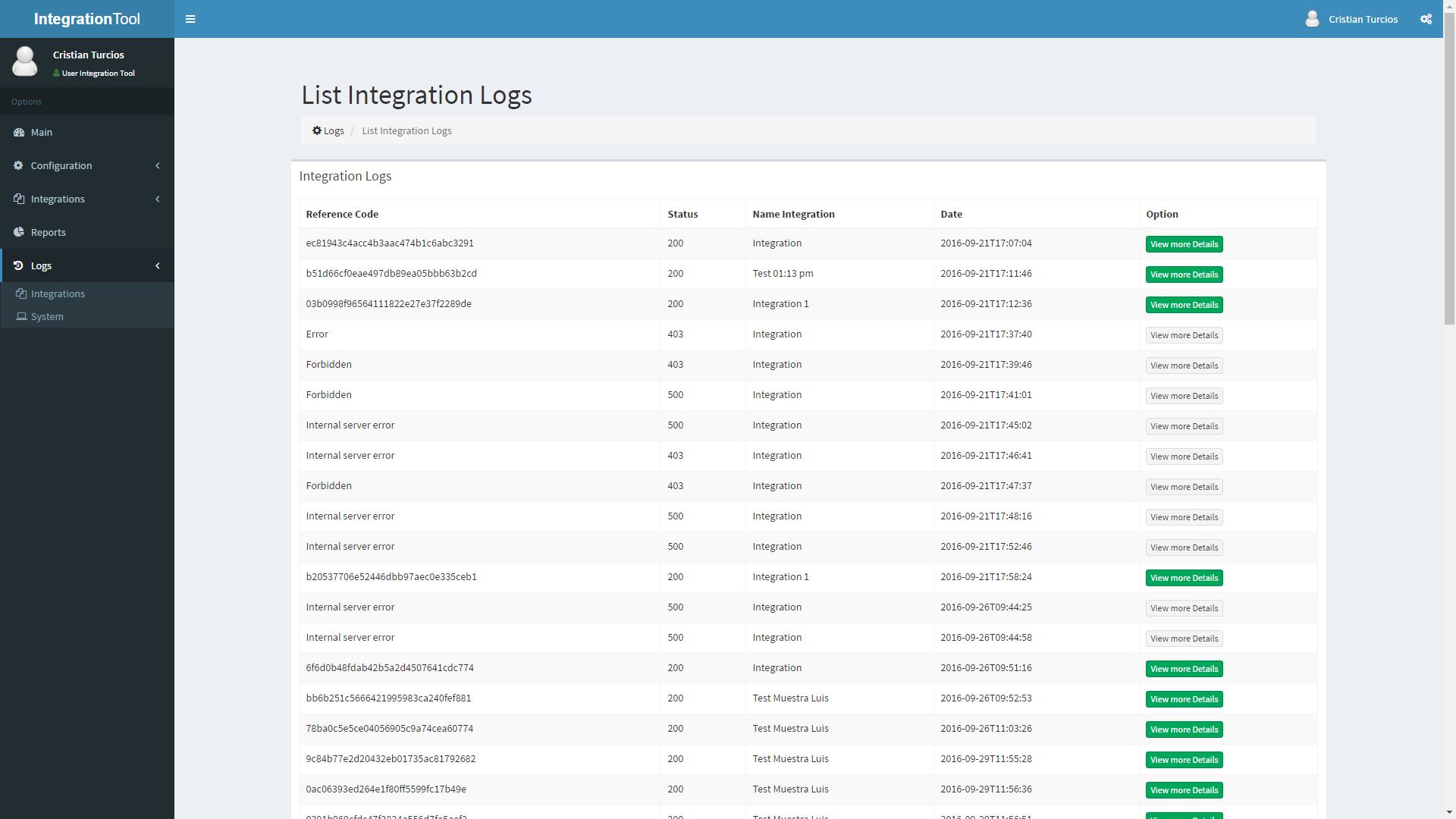
## INTEGRATION LOGS

Al momento de realizar una integracion si se encontró el web service y no se generó ninguna excepción de Sistema se genera un log de las integraciones que se pueden visualizar en la pestaña de Logs - Integrations, en esta ventana se puede visualizar el status de la integracion los tres tipos de status son:

* **Status 200:** el web service se encontró y la integracion se realizó en BlackBoard.

Nota: que devuelva un status 200 no significa que la integracion no tuvo problemas en BlackBoard, solo que encontró el web service y se tuvo acceso a él, con el referente code se puede ir a BlackBoard y verificar el estado o dar en el botón View More Details para conocer el estado de esta integracion.

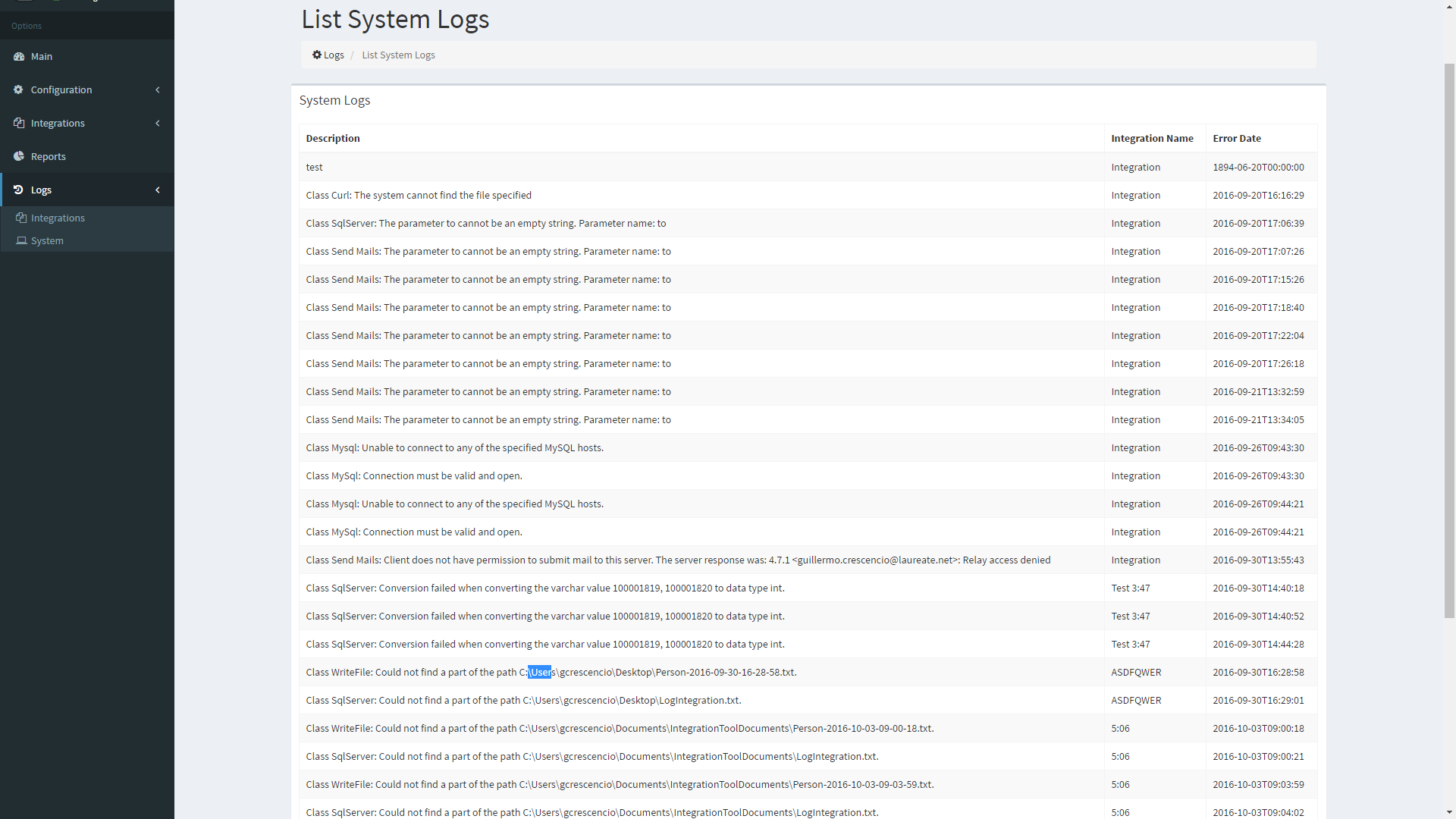
* **Status 500:** el web service al que se quiso acceder no pudo ser encontrado.
* **Status 403:** el nombre de usuario y contraseña que se usaron para consultar ese web Services no posee los permisos para consultarlo



# ORCHESTRATION SCHEME AND SERVICE ACTIVITY

## SYSTEM LOGS

Si al momento de realizar una integracion se genera un error del Sistema se puede consultar la pestaña de System logs y ver en qué lugar y que fue lo que causo la excepción para poderlo solucionar en subsecuentes integraciones.



## SERVICE ACTIVITY DESCRIPTION

### START

The service activity should start when a request from a service activity initiator is sent to a web service published by the integration environment (TIBCO).

This orchestration service should reside on the integration environment and will be responsible for carrying out the service activity and interacting with BlackBoard Learn web services.

The team responsible for creating the integration’s process definition should expose this service to the users of LNPS. This way, the end users will be able of initiate service activity instances on demand. This orchestration service (tentatively named **GradeReport**) will expose one operation named **getGradeReport**. The operation starts the service activity instance. The getGradeReport operation will require two input parameters:

1. **courseId (string)**: this parameter will be used to filter the number of records included in the final report. Only courses matching the pattern sent on this parameter should be included on the report file.
2. **notificationEmail (string)**: this parameter will be used to notify the end user that the service activity has concluded and that the gradebook file is available in the FTP server. This parameter should allow for one or more repetitions in order to notify to more than one email account at the same time.

### CONTEXT.WS

The first interaction required in the service activity is with the Context service. This service provides the initial operations required for session creation. These operations should be called in the following order:

1. **initialize**: this is the first operation of the service activity. It returns a session token to use on subsequent calls to any of the other services available on BlackBoard Learn. This call does not require any parameters and the response of the service is a character string that will be used as password in subsequent service calls.
2. **loginTool**: the service activity should call the loginTool operation which authenticates the session with the credentials of a registered proxy tool on BlackBoard Learn. If the proxy tool has the required entitlements, the session will be able to contact the web services allowed by its entitlements. The requests to loginTool should include the following parameters:
   1. *Password*: Shared secret of the login tool.

Value: Laureate.123 (test) / changeme (prod)

* 1. *clientVendorId*: id of the proxy tool’s vendor.

Value: TIBCO (test) / jmlb (prod)

* 1. *clientProgramId*: id of the proxy tool.

Value: TIBCO (test) / GradeExtractApp (prod)

* 1. *expectedLifeSeconds*: expected lifetime of the session.

Value: Depends, we usually use a high value (in milliseconds).

1. **emulateUser**: this operation allows access to the Gradebook web service operations. The request to emulateUser should include the following parameters:
2. *userToEmulate:* user account emulated through the session.

Value: lnps.tibco (test) / lnps.tibco (prod)

### COURSE.WS

After the session has been initialized and authenticated by Context.WS, the service activity should interact with the Course web service.

From this web service, the service activity will send a request to the **getCourse** operation. The result of this operation returns all the courses that comply with a specified search filter. In order to communicate with this operation, the service activity should provide the following parameters:

1. *filterType*: The filterType describes the type of query you wish to perform.

Value: 5 (Load by searchkey, searchoperator, searchvalue, searchdate, searchdateoperator and course-org flag)

1. *searchDate*: Only used with filterType=5. The creation date to compare against.

Value: 946684800 (2000-01-01 in UNIX time).

1. *searchDateOperator:* Only used with filterType=5. The comparison operator to use for the creation date.

Value: GreaterThan

1. *searchKey:* The searchKey is the field in which you are searching for the searchValue with the given searchOperator matching criteria. Possible values are: CourseId, CourseName, CourseDescription, CourseInstructor.

Value: CourseId

1. *searchOperator:* Only used with filterType=5. The searchOperator describes how to compare the searchValue to the CourseVO values for the given searchKey. Possible values are: Equals, Contains, StartsWith, IsNotBlank.

Value: Contains

1. *searchValue:* Only used with filterType=5. The value you are searching for.

Value: The value of this parameter is the input sent by the service activity initiator on the **courseId** parameter of the **getGradeReport** operation from the GradeReport orchestrator.

The process instance should store the following output parameters from this operation for inclusion in the final report and interaction with other operations: *courseId, batchUid, id, name, courseServiceLevel* and *available*.

## COURSEMEMBERSHIP.WS

The CourseMembership web service provides operations related to memberships with courses and groups. The service activity instance will interact with this service in order to obtain the enrollments of the courses queried with the getCourse operation of courses.WS

The **getCourseMembership** operation returns the list of enrollments for a given course. The parameters required to call this operation are:

1. *filterType:* The filterType describes the type of query you wish to perform.

Value: 2 (Load records by course Id's)

1. *courseId*: Id of the courses queried by getCourse.

Value: The value of this parameter should be the output parameter “*id”*of the operation getCourse from course.WS

The process instance should store the following output parameters from this operation for inclusion in the final report and interaction with other operations: *id, enrollmentdate, available and userId.*

## USER.WS

The user web service provides operations for accessing and updating the users, admin users (persons), users' address book entries. The service activity will interact with this service in order to obtain the personal information of the user enrolled in a course.

The **getUser** operation returns the list of users for a given filter. The parameters required to call this operation are:

1. *filterType:* The filterType describes the type of query you wish to perform.

Value: 2 (Load records by user Id's)

1. *Id*: Id of the user returned by getCourseMembership.

Value: The value of this parameter should be the output parameter “*userId”*of the operation getCourseMembership from courseMembership.WS

The process instance should store the following output parameters from this operation for inclusion in the final report and interaction with other operations: *id, name, userBatchUid, isAvailable and studentId.*

## GRADEBOOK.WS

The gradebook web service provides operations for accessing course gradebooks. The service activity instance will interact with this service in order to obtain the final grades of the users enrolled on the filtered courses.

There are two operations that need to be called for the service activity in the following order:

1. **getGradebookColumns:** returns all the gradebook column definitions for a given course. The requests to getGrades should include the following parameters:
   1. *filterType*: The filterType describes the type of query you wish to perform.

Value: 1 (Load records by course Id's)

* 1. *courseId:* Id of the courses queried by getCourse*.*

*Value:* The value of this parameter should be the output parameter “id” of the operation getCourse from course.WS

The process instance should store the following output parameters from this operation for inclusion in the final report and interaction with other operations: *id, columnDisplayName, externalGrade, deleted, position, aggregationModel, calculationType, dueDate, multipleAttempts, possible, scorable,* and *visible*.

1. **getGrades**: returns grade values from the gradebook based on specified filter. The requests to getGrades should include the following parameters:
   1. *filterType*: The filterType describes the type of query you wish to perform.

Value: 2 (Load records by courseId, userId and columnId)

* 1. *courseId:* Id of the course queried by getCourse.

Value: The value of this parameter should be the output parameter “id” of the operation getCourse from course.WS

* 1. *userIds*: Id of the user returned by getCourseMembership.

Value: The value of this parameter should be the output parameter “*userId”*of the operation getCourseMembership from courseMembership.WS

* 1. *columnId*: Id of the gradebook column returned by getGradebookColumns.

Value: The value of this parameter should be the output parameter “*id”*of the operation getGradebookColumns from gradebook.WS

The process instance should store the following output parameters from this operation for inclusion in the final report: *id, manualScore, manualGrade, status, grade* and *schemaGradeValue*.

## GRADEBOOK REPORT

Once the service activity instance completes its calls to the web services of BlackBoard Learn, it must create a comma delimited document with the output parameters obtained from each operation in the course, courseMembership, user and gradebook web services.

This file should be uploaded via SFTP to a predefined site hosted by LNPS.

Finally, the service activity instance sends an email to the recipients included in the input parameter **notificationEmail** of the getGradeReport operation. The email body should contain a message similar to the following:

“*The Gradebook report request is completed. You can download the document <gradebook report file> from <ftp site address>.*

*The document was created on: <completion timestamp>”*

## ERROR HANDLING

In case of errors that could prevent the creation of the gradebook report, the service activity instance should send an email with the following body:

“*The Gradebook report request ended with errors. The following is a trace of the error:*

*<Trace of the error>*

*Try again at a later time or contact your system administrator for further details.”*

# BLACKBOARD LEARN WS SECURITY CONFIGURATION

The BlackBoard Learn Web Services use WS-Security as the standard security protocol for its SOAP communication.

The SOAP requests to the Initialize operation of the Context web service are used to obtain a session token that will be used as password for subsequent calls to the other web services in BlackBoard Learn. However, since the SOAP requests to the Initialize operation also require credentials, the value “*nosession*” is used as the password to call this operation. The password type used by the web services of Blackboard Learn is *PasswordText.*

The username value used in all SOAP requests to the web services is called “*session”*.

The SOAP requests to the BlackBoard Learn also require a TTL value.

The following table details the security configuration required to interact with each operation of the service composition:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OPERATION | USERNAME | PASSWORD | PASSWORD TYPE | TTL |
| **initialize** | session | nosession | PasswordText | In milliseconds |
| **loginTool** | session | initialize.return | PasswordText | In milliseconds |
| **emulateUser** | session | Initialize.return | PasswordText | In milliseconds |
| **getCourse** | session | Initialize.return | PasswordText | In milliseconds |
| **getCourseMembership** | session | Initialize.return | PasswordText | In milliseconds |
| **getUser** | session | Initialize.return | PasswordText | In milliseconds |
| **getGradebookColumns** | session | Initialize.return | PasswordText | In milliseconds |
| **getGrades** | session | Initialize.return | PasswordText | In milliseconds |

# FTP SERVER LOCATION AND CREDENTIALS

The report file created by the service composition should be stored in the main FTP server of LNPS. The following configuration should be used when uploading the report file to the FTP server:

* Site: sftp.onlinehighered.com
* Protocol: SFTP
* Port: 22
* Username: TIBCO
* Password: L@ur3@t3.2014

The filename format of the gradebook reports should comply with the following specification:

* GradebookReport\_*<creation date in YYYYMMDD\_HHMISS>*

# GRADEBOOK REPORT FORMAT

The gradebook report created by the service composition will be a comma delimited file constructed with the output parameters of each operation called by the service activity. The following table indicates each of the output parameters to be included in the gradebook report, and the title to use for presentation purposes:

|  |  |  |  |
| --- | --- | --- | --- |
| **COLUMN TITLE** | **WEBSERVICE** | **OPERATION** | **PARAMETER** |
| **COURSE\_ID** | course.WS | getCourse | courseId |
| **COURSE\_BATCHUID** | course.WS | getCourse | batchUid |
| **COURSE\_PKID** | course.WS | getCourse | id |
| **COURSE\_TITLE** | course.WS | getCourse | name |
| **COURSE\_TYPE** | course.WS | getCourse | courseServiceLevel |
| **COURSE\_AVAILABLE** | course.WS | getCourse | available |
| **COLUMN\_NAME** | gradebook.WS | getGradebookColumns | columnDisplayName |
| **COLUMN\_PKID** | gradebook.WS | getGradebookColumns | id |
| **IS\_EXTERNAL\_GRADE** | gradebook.WS | getGradebookColumns | externalGrade |
| **COLUMN\_IS\_DELETED** | gradebook.WS | getGradebookColumns | deleted |
| **COLUMN\_PKID** | gradebook.WS | getGradebookColumns | position |
| **COLUMN\_MODEL** | gradebook.WS | getGradebookColumns | aggregationModel |
| **COLUMN\_CALC\_TYPE** | gradebook.WS | getGradebookColumns | calculationType |
| **COLUMN\_DUE\_DATE** | gradebook.WS | getGradebookColumns | dueDate |
| **COLUMN\_MULTI\_ATTEMPTS** | gradebook.WS | getGradebookColumns | multipleAttempts |
| **COLUMN\_POINTS\_POSSIBLE** | gradebook.WS | getGradebookColumns | possible |
| **COLUMN\_IS\_SCORABLE** | gradebook.WS | getGradebookColumns | scorable |
| **COLUMN\_IS\_VISIBLE** | gradebook.WS | getGradebookColumns | visible |
| **USER\_ID** | user.WS | getUser | name |
| **USER\_BATCHUID** | user.WS | getUser | userBatchUid |
| **USER\_PKID** | user.WS | getUser | id |
| **USER\_IS\_AVAILABLE** | user.WS | getUser | isAvailable |
| **USER\_STUDENT\_ID** | user.WS | getUser | studentId |
| **ENR\_PKID** | CourseMembership.WS | getCourseMembership | id |
| **ENR\_IS\_AVAILABLE** | CourseMembership.WS | getCourseMembership | available |
| **ENR\_DATE** | CourseMembership.WS | getCourseMembership | enrollmentDate |
| **GRADE\_DISPLAYED** | gradebook.WS | getGrades | schemaGradeValue |
| **GRADE** | gradebook.WS | getGrades | grade |
| **GRADE\_ID** | gradebook.WS | getGrades | id |
| **GRADE\_MANUAL** | gradebook.WS | getGrades | manualGrade |
| **GRADE\_SCORE\_MANUAL** | gradebook.WS | getGrades | manualScore |
| **GRADE\_STATUS** | gradebook.WS | getGrades | status |