|  |  |
| --- | --- |
| **Document Layout**1: [Overview](#Overview)2: [ERS Payroll Report Enhancement](#Payroll)3: [ERS Scheduler Enhancement](#Scheduler)4: [Development and Testing Approach](#Approach)5: [Bugs Fixes](#Bugs)6: [Document Revision History](#Revision) |   |

# 1: Overview

Academic Personnel requested five new ‘Academic Pay Period’ values be added to the Title Code System (TCS) to facilitate paying lecturers at the semester campuses of Berkeley and Merced, and the law schools at the 1/10th monthly rates. In addition, the new codes will enable UCRS to calculate service credit and/or HAPC for appointees in the new title codes. Title Codes changes in the TCS system directly impacts the Payroll Personnel System (PPS) system which in turn impacts the Effort Reporting System (ERS)

The ERS system uses Title Code data to weight earnings on the ERS Payroll Report. Additionally, Title Code information is used by the ERS Scheduler to determine an Employee’s reporting schedule (by Period). Currently, no UC campus is using the ERS Scheduler function, but title code program logic will be updated in the ERS Scheduler for potential future use.

**Title Codes**

1636 Lecturer – Academic Year 1/10

1637 Lecturer – Academic Year – 1/10 – Continuing

1646 Senior Lecturer – Academic Year – 1/10

1647 Senior Lecturer – Academic Year – 1/10 – Continuing

1654 Continuing Appointment – Temporary Augmentation – 1/10

|  |  |
| --- | --- |
| **Academic Pay Period (APP)**'0' - Not applicable '1' - Monthly payment equals 1/9 annual salary '2' - Monthly payment equals 1/12 annual salary‘3’ - Monthly payment equals 1/10 annual salary (new)blank - Not applicable **Appointment Basis Code (ABC)**'0' - Not applicable '1' - 9 month '2' - 10 month '3' - 11 month '4' - 11 month/Health Science Compensation Plans '5' - Summer Session 'blank' - Not applicable |  ERS Appointment Table |

PPS sends payroll data (containing title codes, appointment basis codes, and academic pay period codes) to the ERS system via the PPS-ERS Payroll Archive Report (PAR) system interface. PAR data is then imported into the ERS Earnings table by ERS program RunPARInterface.java.

The PAR file is a fixed format text file with title code data at the following locations

* Title Code 170-173
* Appointment Basis Code 206-206
* Academic Pay Period 207-207

Once the PAR data has been loaded, Title Code data is used to weight earnings on ERS Payroll Reports. Additionally, Title Code information is used by the ERS Scheduler to determine an Employee’s reporting schedule (by Period). A technical description of these two processes is described in section 2 (ERS Payroll Report Enhancement) and section 3 (ERS Scheduler Enhancement).

**Excerpt from ERS Help**

The Weighted Percent is a value that is calculated by ERS in situations where an employee has a combination of nine-month and eleven-month appointments. Nine month appointments are paid over twelve months. That means that for each three months worked the employee receives four checks. In the case of eleven month appointments the employee receives three checks for each three months worked. The weighted percent is necessary to equalize the paid percentages.

If the system were simply to average the nine month pay with the eleven month pay, the effort calculation would be distorted with the nine month effort being overstated and the eleven month effort being understated. The Weighted Percent solves this problem. When an employee is paid When all payments are made on the same Basis (either 9/12 or 11/12), the Weighted Percent value is copied from the Derived Percent value. When there is a mix of 9/12 and 11/12 payments, the 9/12 payments must be weighted to account for the fact that each 9/12 payment represents one-third effort rather than one-fourth effort. In these cases, the Weighted Percent is calculated as the Derived Percent multiplied by 0.75 and the result can then be combined with the 11/12 payments to arrive at a valid total. This is an unusual situation however, and in most cases employees will have a single Basis and the Weighted Percent value will be the same as the Derived Percent value as calculated by the payroll system.

**2:** **ERS Payroll Report Enhancement**

When Effort Reports are built, ERS calls java class ReportBuilder.java, method establishEffectiveEffort to calculate weighted earnings. One component of the weighted earnings calculation is looking for ‘mix earnings’ in the Effort Report’s earnings records. Mixed earnings is a mix of ‘appointment basis codes’ and ‘academic pay periods’.

For example,

ABC=1 paired with APP=3 represents 9 month appointment paid over 10 months

ABC=3 paired with APP=2 represents 11 month appointment paid over 12 months

If the Effort Report contains a mix of earnings, then a ‘weighting’ calculation is performed to ‘normalize’ the earnings on the payroll report. The weighted earning is found on the ERS Payroll Report in the ‘Weighted %’ column. If the Effort Report does not contain a mix of earnings, the ‘Weighted %’ column on the payroll report is identical to the payroll report ‘Derived %’ column.

Notice the call to ReportBuilder method ‘checkedMixed’ shown below. This method determines if mixed earnings exist in the effort report. The ‘checkedMixed’ method body is shown is the next page.

If mixed earnings exist, the ‘weight’ method is called to perform the weighted earnings calculation. The ‘weight’ method is shown in page 3.

****

Java class ReportBuilder.java, method checkMixed determines if Effort Report earnings are ‘mixed’.

Notice the new nineTenCount variable shown below identifies the new 9 month appointment paid over 10 months.

The ‘checkmixed’ method returns ‘true’ if a mix of earnings exists in the Effort Report’s earnings, otherwise false is returned.

****

Java class ReportBuilder.java, method ‘weight’ (shown below) performs the weighting calculation.

If mixed earnings exist, the weighting method considers five possible scenarios.

* Case 1: 9 month appointment paid over 9 months 🡪 1.0 weighting factor.
* Case 2: 9 month appointment paid over 12 months 🡪.75 weighting factor.
* **Case 3 9 month appointment paid over 10 months 🡪.90 weighting factor. (new)**
* Case 4 11 month appointment paid over 12 months 🡪 1.0 weighting factor.
* Case 5 12 month appointment paid over 12 months 🡪 1.0 weighting factor.

**New 10.12 Weighted Earnings Code:**

****

**Old Weighted Earnings Code:**

**private** List<Map<String, Comparable>> weight( List origin , **boolean** mixed, Set<WeightedEarning> wearns ) {

 List<Map<String, Comparable>> results = **new** ArrayList<Map<String, Comparable>>();

 EarningDAO earndao = **new** EarningDAO();

 WeightedEarningDAO wedao = **new** WeightedEarningDAO();

 **if** (origin==**null**) **return** results;

 **for** (Iterator iter = origin.iterator(); iter.hasNext();) {

 Earning earn = (Earning) iter.next();

 Map<String, Comparable> weighted = **new** HashMap<String, Comparable>();

 weighted.put(ERSConstants.*RPG\_EARNING\_KEY*,earn);

 **if** (mixed) {

 **if** (!is912(earn)) {

 weighted.put(ERSConstants.*RPG\_WEIGHTED\_KEY*, earn.getErnDerivedPct());

 } **else** {

 WeightedEarning wearn = wedao.get(earn.getImportSeqNbr());

 **if** (wearn==**null**) {

 // we don't already have one

 BigDecimal initial = earn.getErnDerivedPct();

 BigDecimal weightedPercent = initial.multiply(**new** BigDecimal(.75));

 weightedPercent = weightedPercent.setScale(4,BigDecimal.*ROUND\_HALF\_EVEN*);

 weighted.put(ERSConstants.*RPG\_WEIGHTED\_KEY*, weightedPercent);

 wearn = **new** WeightedEarning(earn.getImportSeqNbr());

 AuditImpl.*add*(wearn,ERSConstants.*SYSTEM\_USER*);

 wearn.setErnWeightedPct(weightedPercent);

 wearns.add(wearn);

 earn.setWeightedEarning(wearn);

 earndao.update(earn);

 wedao.save(wearn);

 } **else** {

 weighted.put(ERSConstants.*RPG\_WEIGHTED\_KEY*,wearn.getErnWeightedPct());

 }

 }

 } **else** {

 weighted.put(ERSConstants.*RPG\_WEIGHTED\_KEY*, earn.getErnDerivedPct());

 }

 results.add(weighted);

 }

 **return** results;

 }

Sample Effort Report with mixed earnings and weighted projects percentages to be certified.

See the Payroll Detail Report on the following page for a detailed explanation of the project percentages shown below.

****

The following Payroll Report contains 47 earning lines which span five projects. Only the first two projects (17 earning lines) are shown in this example. 46 earnings records have ABC:APP = 32 = 11 month appointment paid over 12 months. One earning (earning 6 in the sponsored project) was set to 13 = 9 month appointment paid over 10 months. This one earning triggers the mixed earnings / weighted calculation.



Next, notice the weighted earnings for the sponsored project is 1.0295 and the report total weight is 5.4965.

Therefore the sponsored project represent 19% of the total effort on this report.

This ties back to the Effort Report certification screen (prior page) sponsored project row, original payroll % column.

**3:** **ERS Scheduler Enhancement**

Title Code information is used by the ERS Scheduler to determine an Employee’s reporting schedule (by Period). There are four components to the schedule:

1. Schedule definitions defined in “Manage Schedule Types” on the ERS Admin menu
2. Reporting periods defined in “Manage Report Periods” on the ERS Admin menu
3. Employee schedule assignments as define in “Update Employee Schedule Assignment”
4. Processing of the PAR file

When a PAR file is processed, any non-excluded earnings (representing effort) are compared to current schedule definitions to determine the employee reporting / period schedule. The resulting schedule is set in the ERSEmpl (employee) table, column sch\_cd.

Currently, no UC campus is using the ERS Scheduler function, but ERS title code program logic will be updated for potential future use. Testing of this release will only focus on proper setting of the employee schedule (sch\_cd) in the ERSEmpl table.

The following are screen shots from “Manage Schedule Types” on the ERS Admin menu



Notice the new appointment 9/10 shown below.



Here are the database tables behind these screens. Table ERSSchType is the header table shown on the prior page. Table ERSSchAppt defines the appointment types connected to a particular schedule.



When the PAR interface is run, schedules are analyzed and the employee’s reporting schedule is derived. The following screen shots shows snippets of code from class InterfacePARProcessor.java, which is responsible for setting the employees schedule.

Class InterfacePARProcessor first loads a list of all ERS schedules (getSchedulerCheckers) and then iterates through all PAR records by employee (while loop). When the PAR employee changes, the setScheduleForCurrentEmployee method is called to set the employee’s schedule



Method setScheduleForCurrentEmp (called above) calls EarningsEffectiveValuesParser method findEffectiveValue (next page) to determine the correct effective value for:

* Personel Program Code (PPC)
* Exempt Status
* Title Code
* Appointment (new 9/10 value)
* Home Department

Method setScheduleForCurrentEmp (called above) is shown below. Notice the call to findEffectiveValue (below)

Which calls method testAppointmentCodes to determine the highest ranked of appointment code. If the resulting appointment code is null, then method setDefaultApptIfNeeded is called to set the default appointment.





Methods testAppointmentCodes and setDefaultApptIfNeeded are described in the prior page and shown below.



Once the call to method findEffectiveValaue is complete, the employee’s schedule has been identified and needs to be updated in the ERSEmpl (employee table). Method setScheduleType (shown below) sets column sch\_cd in the ERS\_EMPL table (details described on the next page).



Class ScheduleSetter, method setScheduleType (shown above) first looks to see if the employee’s current schedule was set manually. If so, the employee’s schedule is not changed.

Here are the schedule assignment values as defined in class ERSConstants.java



Method setScheduleType then looks to see if the number of schedules that match the employees PAR Appointment Basis Code and Academic Pay Period (and other parameters) is zero, one, or greater than one.

* If 0, the default schedule is assigned to the employee

see “Manage Schedule Types”

* if 1, the matching schedule is assigned to the employee
* if >1 the highest ranked schedule is assigned to the employee.

The ranking of Appointment Basis Code and Academic Pay Periods is:

1. 9/9 9 month appointment paid over 9 months
2. 9/10 9 month appointment paid over 10 months (new)
3. 9/12 9 month appointment paid over 12 months
4. 11/12 11 month appointment paid over 12 months

Once the employee’s schedule has been set, the employee table is updates as shown below.



Schedule assignment as shown on the ERS System Administration menu ‘Update Employee Schedule Assignment’





**4:** **Development and Testing Approach:**

**Weighted Earnings**

* Desktop Spring, ERS, Tomcat.
* Development database dbd1.ucop.edu.
* Find a mix of effort reports with sponsored and unsponsored projects.
* Manually update an earnings record and change Academic Pay Period to 3 (10 month).
* Run program ReportRemover to remove the current effort report.
* Run ‘Generate Effort Report’ on System Admin menu to generate a new report.
* Validate the ‘weighted %’ column on the new Effort Report**.**

**Schedule Maintenance**

* Desktop Spring, ERS, Tomcat.
* Development database dbd1.ucop.edu.
* Update the Academic schedule in Manage Schedule Types on the ERS Administration Menu, to included 9/10 schedule representing the new 9 month appointment paid over 10 months.
* Obtain a new PAR file containing unprocessed earnings.
* Identify a test group of earnings and employees that will flow thru the scheduler process.
* Set table ERSEarnings columns tc\_acad\_appt\_basis to 1 and tc\_acad\_pay\_period=3

representing the new 9 month appointment paid over 10 months.

* Run the PAR interface.
* Verify the “academic” schedule is assigned to the employee.

**JUNIT Updates**

* No new JUnit scripts have been added
* No existing JUnit scripts have been changed.

**5: Bugs Fixes**

These bugs and enhancements will be addressed in this release:

**Bug 3815**

Mixed Earnings are not accurately calculating 'normalized value' correctly for the weighted earnings.

Modify the system to correctly calculate mixed earning on the ERS payroll report.

Section 2 ( [ERS Payroll Report Enhancement](#Payroll) ) outlines the changes made to the weighting calculation on the ERS Payroll Report. Setup test scenarios with Debra Henn to validate weighted earnings are properly calculated on the ERS Payroll Report. Identify a reporting period and a group of Effort Reports to be uses as test cases. The group should contain effort reports with and without mixed earnings. In the ERSEarnings table, set values for columns TITLE\_CODE, TC\_NAME, TC\_ACAD\_APPT\_BASIS, TC\_ACAD\_PAY\_PERIOD. Delete existing Effort Reports using the SQL ‘delete’ script. Regenerate the Effort Report using the ‘Generate Effort Report for Employee’ menu option on the ERS System Administration menu. Note changes made and forward to Debra Henn. Provide Debra Henn with a URL to the UC Davis / Oracle environment for validation.

**Bug 3813**

Create 'how to' information for end users for release 10.11 Archive and Purge

Update online contextual help, the Installation and Operations Guide and training materials for the Archive and Purge process.

Add ‘ers\_panel\_titlebar’ panel to archive effort report prompt viewArchiveGen.jspf and archive effort report search results viewArchiveGenResults.jspf . Link the help icon in ‘ers\_panel\_titlebar’ to help text archive.jspf using the reference ‘adminHelp.do?page=archive’. Create archive help text document archive.jspf in WbeRoot/help/admin folder. Create an entry for archive.jspf in tiles-defs.xml to link archive.jspf to the help system.

**Bug 3812**

Online report views erroneously mark reports as overdue when added on same day as period due date

Correct system to eliminate online error message that incorrectly indicates overdue status for the compliance report.

When and Effort Report is added on the last day of a reporting period, the ‘add date’ of the Effort Report is the same as the period end date, but the time stamp on the ‘add date’ is after the time stamp (midnight – 00:00:00) of the period end date. This is why the effort report is tagged as late.

See class Effort Report.java method getOverdueFlag(). Change the time in ‘getAddDate()’ to midnight (00:00:00). Effectively, this takes time out of the date comparison. Now, the effort report ‘add date’ = the period ‘end date’. Since the effort report ‘add date’ is no longer after the period ‘end date’, the effort report is no longer over due.

**Enhancement 3801**

Enhance count in archive search results + excel export

Provide a system indicator to identify total available transactions in search request.

Add a HQL statement to ViewArchiveResultAction to count the number of rows in class ArchivedReport that match the search criteria. Display the count in viewArchiveGenResults.jspf .

Provide an ‘Excel’ export button at the bottom of the archive results screen. This button will export the search criteria and search results to excel. Add a new method to ViewArchiveResultsAction to generate excel results. Display the excel data in viewArchiveGenExcel.jspf. Add viewArchiveGenExcel.jspf to tiles-defs.xml.

**Enhancement 3799**

Handling of late pay transactions that belong to archived reports

Archive transactions, provide an audit trail and add to the archived reporting for the employee.

Earnings belonging to archived Effort Reports are easily identified. The period associated with the earning will have it’s period\_status =’A’ (archived) in table ERSRprtPeriod. Earnings for late pay transactions will remain in the ERSEarnings table until the archived period is purged.

When an Effort Report is retrieved from the archives, the ERSEarnings table needs to queried for earnings belonging to the archived effort report. The Late Pay Report should be printed after the first Effort Report. Add method retreiveLatePayTransactions() to class ViewArchiveExportAction() to compile a list of earnings belonging to the archived effort report. Add method jasperPrintLatePayAfterArchive() to class ArchiveJasperPrint() to Jasper format the list of late pay earnings and pass to Jasper Report archiveLatePayAfterArchive.jrxml. Create fictitious ERSEarnings rows for an archived effort report. Retreive the Effort Report from the archives and verify the archived Effort Report late pay transactions. Forward report to Debra Henn for format acceptance.

**Bug 1579**

Decentralized resource validation on org hierarchy broken by unnecessary join on PI info

Adjust tables to eliminate system generated failure error.

See menu option’ View Permissions’ on the ‘ERS System Administration Menu’

Adding a ‘Sponsored Project Resource’ results in an error message, “you are not authorized to the sponsored project”.

To fix this problem see class UCOPAdminAuthorizerImpl() HQL statement IS\_AUTHORIZED\_RESOURCE\_SPONSORED\_PROJECT\_BY\_ORG\_CD.

The following classes are linked incorrectly:

Change from:

ERSSSponsoredProject 🡺 ERSPI 🡺 ERSEMPL 🡺 ERSHOMEDEPT 🡺 ERSORGDEPT 🡺ERSORGANIZATIONFLAT

Change to:

ERSSSponsoredProject 🡺 ERSFundingSource 🡺 ERSORGANIZATIONFLAT (for both Acct\_Org\_Dc and Fund\_Org\_Cd)

Be sure to set the ‘decentralized security’ flag to true in ERSConfig.xml to emulate UC Davis scenario.

When the code is fixed, set up a test scenario in the UC Davis Oracle environment for Debra Henn to test.

**Bug 3819**

Remove periods that have been archived from period prompt in ‘Effort Report Generation’.

When Effort Reports are generated for a period, the drop down list of periods incorrectly contains archived periods.

Archived periods should be removed from this list.

See class ViewIndividualReportGenAction() method makeReportPeriodList

Modify the ‘rpList = rpdao.find to include a qualifier to exclude archived reports.

**Bug 3820**

FUA truncated in Archive Payroll Report.

Widen FAU in Jasper Report ‘archiveDrillDownReport2.jrxml’ to accommodate 30 character FAU.

Also, verify FAU in Archive Late Pay Report ‘archiveLatePayAfterArchive.jrxml’

## 6: Document Revision History

|  |  |  |
| --- | --- | --- |
| **Version** |  **Date** | **Changes** |
| 1.0 |  2014 - May 7 | Original Spec |
| 1.1 |  2014 - May 19 | Additional documentation and refinements |
| 1.2 |  2014 - July 22  | Additional documentation and refinements |
| 1.3 |  2014 – August 16  | Addition of bug section 5 |

**Appendix 1 - Technical Note:**

Notice how columns in the ERSRPTFUNDITEM table tie to the certification of the ‘sponsored’ project.

Notice how columns in the ERSRPTCATITEM table tie to the certification of ‘other’ and ‘non sponsored’ projects.



**Appendix 2 - Technical Note: Menu Option "Generate Effort Reports For Employees"**

class ReportPeriodGeneration / method genEffortReports

{

 ReportBuilder builder = new ReportBuilder();

 builder.categorizeEarnings( errorFaus,reportHeader, matcher,

 qualifiedEarnings, categorized, categorizedWeighted, fsFinder );

 categorized.put(ERSConstants.RPG\_NONSPONSORED\_KEY,nonSponsored);

 categorized.put(ERSConstants.RPG\_OTHERSPONSORED\_KEY, otherSponsored);

 categorized.put(ERSConstants.RPG\_INELIGIBLE\_KEY, ineligibleEarnings);

 categorized.put( ERSConstants.RPG\_EXCLUDED\_KEY, excludedEarnings );

 categorized.put(ERSConstants.RPG\_CERTIFIABLE\_KEY, certifiable);

 categorized.put(ERSConstants.RPG\_INVALID\_KEY, invalidEarnings);

 establishEffectiveEffort(categorized, categorizedWeighted);

\*\*\* boolean mixed = checkMixed(categorized);

 for (Iterator iter = keys.iterator(); iter.hasNext();) {

 Integer key = (Integer) iter.next();

 List<Map<String, Comparable>> weightedProj

 = new ArrayList<Map<String, Comparable>>();

 List earns = (List) certifiable.get(key);

 List origin = new ArrayList();

 if (earns!=null) {

 origin.addAll(earns);

 }

 if (mixed) {

\*\*\* weightedProj = weight(origin, true, wearns);

 } else {

 weightedProj = weight(origin, false, wearns);

 }

 certifiableWeighted.put(key,weightedProj);

**see screenshot categorizedWeighted.put(ERSConstants.RPG\_CERTIFIABLE\_KEY,certifiableWeighted);**

**below**  categorizedWeighted.put(ERSConstants.RPG\_INELIGIBLE\_KEY,ineligibleWeighted);

 categorizedWeighted.put(ERSConstants.RPG\_OTHERSPONSORED\_KEY,otherWeighted);

 categorizedWeighted.put(ERSConstants.RPG\_NONSPONSORED\_KEY,nonWeighted);

 categorizedWeighted.put(ERSConstants.RPG\_INVALID\_KEY,invalidWeighted);

 er = builder.makeEffortReport(categorizedWeighted, categorized, reportHeader, addUser);

 EffortReport er = addVersion1\_0(categorizedWeighted, categorizedEarnings, aer, addUser);

 buildLines(er,categorizedWeighted,true, decisions);

 constructLines( er, categorizedWeighted, decisions, null );

\*\*\*\* ReportFundItem rfi = makeFundLine( er, spid, (List)certifiable.get(spid),projPercent,

 commitTotalHandle, enteredCommitmentMap, commitMap );

 newRptFundItem.setGenPctEffort(projPercent);

saveLines( er, replicateToEntered, adjustedLines, null );

\*\*\*\* rfidao.save(rfi); **// save report fund item**

}

**Appendix 2 - Technical Note: (continued)**

This code creates the ERSRPTFUNDITEM row. (sponsored project row on the Effort Report certification screen)

Class: ReportBuilder Method: constructLines(...)

for (Iterator iterator = keys.iterator(); iterator.hasNext();) {

 BigDecimal projTotal = new BigDecimal("0.0000");

 Integer earnKey = (Integer) iterator.next();

 List projEarn = (List)certifiable.get(earnKey);

 for (Iterator iter = projEarn.iterator(); iter.hasNext();) {

 Map earnMap = (Map) iter.next();

 BigDecimal weightedEffort

= (BigDecimal) earnMap.get(ERSConstants.RPG\_WEIGHTED\_KEY); // weightedEffort = .1878

 erTotal = addEarningToCurrentTotal(weightedEffort, erTotal); // erTotal = 5.0610

 projTotal = addEarningToCurrentTotal(weightedEffort, projTotal);

 }

 projTotals.put(earnKey,projTotal); // earnKey = 12033 projTotal=.1878

}

ReportFundItem rfi =

 makeFundLine( er, spid, (List)certifiable.get(spid),projPercent, commitTotalHandle, enteredCommitmentMap, commitMap );

 makeFundLine newRptFundItem.setGenPctEffort(projPercent);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

When viewing an effort report, press button “view payroll details”.

The following code builds the payroll report:

Class: DrillDownBuilder.java Method: buildfull

 Reportdetails.do

 ReportDetailsAction.java

 DrillDownBuilder builder = new DrillDownBuilder()

 DrillDownStructure ddStruct = builder.buildFull(rptID)

Table note:

ERSEarningsVersion.SP-ID = sponsored project ID

ERSRPTFundItem – rpt\_seq\_nbr + SP\_ID

**NOTE 1:**

