Project: PRJ872695

Platform ***Default***
Database ***PY900SXF***
Version ***85***
Description ***UPD872695***
Comments
Tax Update 12-F for Release 9.0

Records

CAN_TAX_PROV
CAN_TAX_PROV_AB
CAN_TAX_PROV_BC
CAN_TAX_PROV_MB
CAN_TAX_PROV_NB
CAN_TAX_PROV_NF
CAN_TAX_PROV_NN
CAN_TAX_PROV_NS
CAN_TAX_PROV_NT
CAN_TAX_PROV_ON
CAN_TAX_PROV_PE
CAN_TAX_PROV_QC
CAN_TAX_PROV_SK
CAN_TAX_PROV_YT
CAN_TAX_PROV_ZZ

Fields

PST_PCT
TX810_3_BTN1

Components

PY_T4A_PRT_CAN (CAN)

PeopleCode (Record)

CAN_YE_T4_VW2.CAN_YE_BOX_AMT.RowInit
DERIVED_PAY.TX810_3_BTN1.FieldChange
PY_IC_PI_SUM_VW.OPRID.FieldFormula

PeopleCode (Page)

RUNCTL_TAX810NE.Activate

File References

CTX900_SQR
CTX910AP_SQR

CTX910AU_SQR
CTX910LD_SQR
CTX910TP_SQR
CTX910XM_SQR
DDP003_SQR
PAY002_SQR
PAY003_SQR
PAY014_SQR
PAY018_SQR
PAYGL01_SQR
PSCCTXRT_CBL
PSCLCLRT_CBL
PSCSTTRT_CBL
PSCTARRY_CBL
PSCTXARY_CBL
PSCTXLNK_CBL
PSPCNTAX_CBL
PSPCTXRT_CBL
PSPDEDTX_CBL
PSPMFLSA_CBL
PSPPYNET_CBL
PSPPYTIP_CBL
PSPSLSTX_CBL
PSPTARRY_CBL
PSPTAXDT_CBL
PSPTCALC_CBL
PSPUSTAX_CBL
TAX001_SQR
TAX007_SQR
TAX010FD_SQR
TAX011PA_SQR
TAX014_SQR
TAX017_SQR
TAX810NM_SQR
TAX900_SQR
TAX910ER_SQR
TAX910LD_SQR
TAX916PA_SQR
TAX930PM_SQR
TAX950_SQR
TAX960LC_SQR
UPD872695_01_DAT
UPD872695_01_DMS
UPD872695_02_DAT
UPD872695_02_DMS
UPD872695_03_DMS
UPD872695_04_DAT
UPD872695_04_DMS
UPD872695_05_DAT
UPD872695_05_DMS
UPD872695_06_DAT
UPD872695_06_DMS

Object Type: 69 ?

```
COBOL, , ,
MAINT, , ,
SQR, , ,
```

Object Type: 85 ?

```
PYT412N_CO_1, , ,
PYT412N_EE_1, , ,
PYT412N_GVT_1, , ,
PYT412S_EE_1, , ,
PYT4A12N_CO_1, , ,
PYT4A12N_EE_1, , ,
PYT4A12N_GVT_1, , ,
PYT4A12S_EE_1, , ,
```

Object Type: 86 ?

```
PYT412N_CO, , ,
PYT412N_EE, , ,
PYT412N_GVT, , ,
PYT412S_EE, , ,
PYT4A12N_CO, , ,
PYT4A12N_EE, , ,
PYT4A12N_GVT, , ,
PYT4A12S_EE, , ,
```

Object Type: 87 ?

```
1OZUHVEI4HGA4KBN973CQ, , ,
DVYWWDQI4HGPVBUQJYJBEA, , ,
LRJVRNQI4HGPVBUQJYJBEA, , ,
OBUFENQI4HGPVBUQJYJBEA, , ,
S7DKBVII4HGA4KBN973CQ, , ,
SZLJ2NMI4HGPVBUQJYJBEA, , ,
VJQZFSGI4HGKOR9MCUYJSW, , ,
Y1ZD2FII4HGA4KBN973CQ, , ,
```

Object Type: 88 ?

```
XML, PYT412, ,
XML, PYT4A12, ,
```

Record Detail

Record: CAN_TAX_PROV

Description CAN Provincial Tax Parameters
Record Type SQL Table
Parent Record CAN_TAX_TBL
Related Language
DDL Space Name PYAPP.PSHRDMOP
Owner ID Payroll
Comments Calculation formula parameters for income tax surtax and other provi
ncial taxes.

FieldName	Type	Len	Key	Srch	Req	Edit	Prompt	C	Default
EFFDT	Date	10	KD		Req				%date
PROVINCE	Char	6	KALS		Req				
CIT_V	Num	1.5							
CIT_V1A	Num	1.5							
CIT_V1B	Num	7.2							
CIT_V2	Num	1.5							
CIT_V3	Num	1.5							
CIT_V4	Num	5.2							
CIT_S	Num	5.2							
CIT_N1	Num	1.4							
CIT_N2	Num	7.2							
CIT_NCA	Num	7.2							
CIT_TH	Num	5.2							
HEALTH_INS_RT	Num	1.6							
WCB_LIMIT	Num	7.2							
PST_PCT	Num	2.2							
PST_PCT	*Num*	*2.3*							
PAYROLL_TAX_RT	Num	1.4							
PSTI_PCT	Num	2.2							
LCP_MAX	Num	5.2							
LCP_RT	Num	0.5							
HST_PCT	Num	2.2							
CIT_V1A2	Num	1.5							
CIT_V1B2	Num	7.2							
HP_STATS_PROV_CD	Char	5			Req				'99999'
PIT_K1P	Num	1.5							
PIT_TCP	Num	7.2							

Record: CAN_TAX_PROV_AB

Description CAN Provincial Tax Parameters
Record Type SQL View
Parent Record CAN_TAX_TBL
Related Language
Build Seq 1
Owner ID Payroll
Comments Calculation formula parameters for income tax surtax and other provi
ncial taxes.

FieldName	Type	Len	Key	Srch	Req	Edit	Prompt	C	Default
EFFDT	Date	10	KD		Req				%date
PROVINCE	Char	6	KALS		Req				
CIT_V	Num	1.5							
CIT_V1A	Num	1.5							
CIT_V1B	Num	7.2							
CIT_V2	Num	1.5							
CIT_V3	Num	1.5							
CIT_V4	Num	5.2							
CIT_S	Num	5.2							
CIT_N1	Num	1.4							
CIT_N2	Num	7.2							
CIT_NCA	Num	7.2							
CIT_TH	Num	5.2							
HEALTH_INS_RT	Num	1.6							
WCB_LIMIT	Num	7.2							
PST_PCT	Num	2.2							
PST_PCT	*Num*	*2.3*							
PAYROLL_TAX_RT	Num	1.4							
PSTI_PCT	Num	2.2							

LCP_MAX	Num	5.2				
LCP_RT	Num	0.5				
HST_PCT	Num	2.2				
CIT_V1A2	Num	1.5				
CIT_V1B2	Num	7.2				
HP_STATS_PROV_CD	Char	5		Req		'99999'

Record: CAN_TAX_PROV_BC

Description CAN Provincial Tax Parameters
Record Type SQL View
Parent Record CAN_TAX_TBL
Related Language
Build Seq 1
Owner ID Payroll
Comments Calculation formula parameters for income tax surtax and other provi
ncial taxes.

FieldName	Type	Len	Key	Srch	Req	Edit	Prompt	C	Default
EFFDT	Date	10	KD		Req				%date
PROVINCE	Char	6	KALS		Req				
CIT_V	Num	1.5							
CIT_V1A	Num	1.5							
CIT_V1B	Num	7.2							
CIT_V2	Num	1.5							
CIT_V3	Num	1.5							
CIT_V4	Num	5.2							
CIT_S	Num	5.2							
CIT_N1	Num	1.4							
CIT_N2	Num	7.2							
CIT_NCA	Num	7.2							
CIT_TH	Num	5.2							
HEALTH_INS_RT	Num	1.6							
WCB_LIMIT	Num	7.2							
PST_PCT	**Num**	**2.2**							
PST_PCT	***Num***	***2.3***							
PAYROLL_TAX_RT	Num	1.4							
PSTI_PCT	Num	2.2							
LCP_MAX	Num	5.2							
LCP_RT	Num	0.5							
HST_PCT	Num	2.2							
CIT_V1A2	Num	1.5							
CIT_V1B2	Num	7.2							
HP_STATS_PROV_CD	Char	5			Req				'99999'

Record: CAN_TAX_PROV_MB

Description CAN Provincial Tax Parameters
Record Type SQL View
Parent Record CAN_TAX_TBL
Related Language
Build Seq 1
Owner ID Payroll
Comments Calculation formula parameters for income tax surtax and other provi
ncial taxes.

FieldName	Type	Len	Key	Srch	Req	Edit	Prompt	C	Default
EFFDT	Date	10	KD		Req				%date
PROVINCE	Char	6	KALS		Req				
CIT_V	Num	1.5							
CIT_V1A	Num	1.5							
CIT_V1B	Num	7.2							
CIT_V2	Num	1.5							
CIT_V3	Num	1.5							
CIT_V4	Num	5.2							
CIT_S	Num	5.2							
CIT_N1	Num	1.4							
CIT_N2	Num	7.2							

CIT_NCA	Num	7.2				
CIT_TH	Num	5.2				
HEALTH_INS_RT	Num	1.6				
WCB_LIMIT	Num	7.2				
PST_PCT	**Num**	**2.2**				
PST_PCT	***Num***	***2.3***				
PAYROLL_TAX_RT	Num	1.4				
PSTI_PCT	Num	2.2				
LCP_MAX	Num	5.2				
LCP_RT	Num	0.5				
HST_PCT	Num	2.2				
CIT_V1A2	Num	1.5				
CIT_V1B2	Num	7.2				
HP_STATS_PROV_CD	Char	5		Req		'99999'

Record: CAN_TAX_PROV_NB

Description CAN Provincial Tax Parameters
Record Type SQL View
Parent Record CAN_TAX_TBL
Related Language
Build Seq 1
Owner ID Payroll
Comments Calculation formula parameters for income tax surtax and other provi
ncial taxes.

FieldName	Type	Len	Key	Srch	Req	Edit	Prompt	C	Default
EFFDT	Date	10	KD		Req				%date
PROVINCE	Char	6	KALS		Req				
CIT_V	Num	1.5							
CIT_V1A	Num	1.5							
CIT_V1B	Num	7.2							
CIT_V2	Num	1.5							
CIT_V3	Num	1.5							
CIT_V4	Num	5.2							
CIT_S	Num	5.2							
CIT_N1	Num	1.4							
CIT_N2	Num	7.2							
CIT_NCA	Num	7.2							
CIT_TH	Num	5.2							
HEALTH_INS_RT	Num	1.6							
WCB_LIMIT	Num	7.2							
PST_PCT	**Num**	**2.2**							
PST_PCT	***Num***	***2.3***							
PAYROLL_TAX_RT	Num	1.4							
PSTI_PCT	Num	2.2							
LCP_MAX	Num	5.2							
LCP_RT	Num	0.5							
HST_PCT	Num	2.2							
CIT_V1A2	Num	1.5							
CIT_V1B2	Num	7.2							
HP_STATS_PROV_CD	Char	5			Req				'99999'

Record: CAN_TAX_PROV_NF

Description CAN Provincial Tax Parameters
Record Type SQL View
Parent Record CAN_TAX_TBL
Related Language
Build Seq 1
Owner ID Payroll
Comments Calculation formula parameters for income tax surtax and other provi
ncial taxes.

FieldName	Type	Len	Key	Srch	Req	Edit	Prompt	C	Default
EFFDT	Date	10	KD		Req				%date
PROVINCE	Char	6	KALS		Req				
CIT_V	Num	1.5							

CIT_V1A	Num	1.5				
CIT_V1B	Num	7.2				
CIT_V2	Num	1.5				
CIT_V3	Num	1.5				
CIT_V4	Num	5.2				
CIT_S	Num	5.2				
CIT_N1	Num	1.4				
CIT_N2	Num	7.2				
CIT_NCA	Num	7.2				
CIT_TH	Num	5.2				
HEALTH_INS_RT	Num	1.6				
WCB_LIMIT	Num	7.2				
PST_PCT	Num	2.2				
PST_PCT	*Num*	*2.3*				
PAYROLL_TAX_RT	Num	1.4				
PSTI_PCT	Num	2.2				
LCP_MAX	Num	5.2				
LCP_RT	Num	0.5				
HST_PCT	Num	2.2				
CIT_V1A2	Num	1.5				
CIT_V1B2	Num	7.2				
HP_STATS_PROV_CD	Char	5		Req		'99999'

Record: CAN_TAX_PROV_NN

Description CAN Provincial Tax Parameters
Record Type SQL View
Parent Record CAN_TAX_TBL
Related Language
Build Seq 1
Owner ID Payroll
Comments Calculation formula parameters for income tax surtax and other provi ncial taxes.

FieldName	Type	Len	Key	Srch	Req	Edit Prompt	C	Default
EFFDT	Date	10	KD		Req			%date
PROVINCE	Char	6	KALS		Req			
CIT_V	Num	1.5						
CIT_V1A	Num	1.5						
CIT_V1B	Num	7.2						
CIT_V2	Num	1.5						
CIT_V3	Num	1.5						
CIT_V4	Num	5.2						
CIT_S	Num	5.2						
CIT_N1	Num	1.4						
CIT_N2	Num	7.2						
CIT_NCA	Num	7.2						
CIT_TH	Num	5.2						
HEALTH_INS_RT	Num	1.6						
WCB_LIMIT	Num	7.2						
PST_PCT	Num	2.2						
PST_PCT	*Num*	*2.3*						
PAYROLL_TAX_RT	Num	1.4						
PSTI_PCT	Num	2.2						
LCP_MAX	Num	5.2						
LCP_RT	Num	0.5						
HST_PCT	Num	2.2						
CIT_V1A2	Num	1.5						
CIT_V1B2	Num	7.2						
HP_STATS_PROV_CD	Char	5			Req			'99999'

Record: CAN_TAX_PROV_NS

Description CAN Provincial Tax Parameters
Record Type SQL View
Parent Record CAN_TAX_TBL
Related Language
Build Seq 1

Owner ID Payroll
Comments Calculation formula parameters for income tax surtax and other provi
ncial taxes.

FieldName	Type	Len	Key	Srch	Req	Edit	Prompt	C	Default
EFFDT	Date	10	KD		Req				%date
PROVINCE	Char	6	KALS		Req				
CIT_V	Num	1.5							
CIT_V1A	Num	1.5							
CIT_V1B	Num	7.2							
CIT_V2	Num	1.5							
CIT_V3	Num	1.5							
CIT_V4	Num	5.2							
CIT_S	Num	5.2							
CIT_N1	Num	1.4							
CIT_N2	Num	7.2							
CIT_NCA	Num	7.2							
CIT_TH	Num	5.2							
HEALTH_INS_RT	Num	1.6							
WCB_LIMIT	Num	7.2							
PST_PCT	Num	2.2							
PST_PCT	*Num*	*2.3*							
PAYROLL_TAX_RT	Num	1.4							
PSTI_PCT	Num	2.2							
LCP_MAX	Num	5.2							
LCP_RT	Num	0.5							
HST_PCT	Num	2.2							
CIT_V1A2	Num	1.5							
CIT_V1B2	Num	7.2							
HP_STATS_PROV_CD	Char	5			Req				'99999'

Record: CAN_TAX_PROV_NT

Description CAN Provincial Tax Parameters
Record Type SQL View
Parent Record CAN_TAX_TBL
Related Language
Build Seq 1
Owner ID Payroll
Comments Calculation formula parameters for income tax surtax and other provi
ncial taxes.

FieldName	Type	Len	Key	Srch	Req	Edit	Prompt	C	Default
EFFDT	Date	10	KD		Req				%date
PROVINCE	Char	6	KALS		Req				
CIT_V	Num	1.5							
CIT_V1A	Num	1.5							
CIT_V1B	Num	7.2							
CIT_V2	Num	1.5							
CIT_V3	Num	1.5							
CIT_V4	Num	5.2							
CIT_S	Num	5.2							
CIT_N1	Num	1.4							
CIT_N2	Num	7.2							
CIT_NCA	Num	7.2							
CIT_TH	Num	5.2							
HEALTH_INS_RT	Num	1.6							
WCB_LIMIT	Num	7.2							
PST_PCT	Num	2.2							
PST_PCT	*Num*	*2.3*							
PAYROLL_TAX_RT	Num	1.4							
PSTI_PCT	Num	2.2							
LCP_MAX	Num	5.2							
LCP_RT	Num	0.5							
HST_PCT	Num	2.2							
CIT_V1A2	Num	1.5							
CIT_V1B2	Num	7.2							
HP_STATS_PROV_CD	Char	5			Req				'99999'

Record: CAN_TAX_PROV_ON

Description CAN Provincial Tax Parameters
Record Type SQL View
Parent Record CAN_TAX_TBL
Related Language
Build Seq 1
Owner ID Payroll
Comments Calculation formula parameters for income tax surtax and other provi
ncial taxes.

FieldName	Type	Len	Key	Srch	Req	Edit	Prompt	C	Default
EFFDT	Date	10	KD		Req				%date
PROVINCE	Char	6	KALS		Req				
CIT_V	Num	1.5							
CIT_V1A	Num	1.5							
CIT_V1B	Num	7.2							
CIT_V2	Num	1.5							
CIT_V3	Num	1.5							
CIT_V4	Num	5.2							
CIT_S	Num	5.2							
CIT_N1	Num	1.4							
CIT_N2	Num	7.2							
CIT_NCA	Num	7.2							
CIT_TH	Num	5.2							
HEALTH_INS_RT	Num	1.6							
WCB_LIMIT	Num	7.2							
PST_PCT	Num	2.2							
PST_PCT	*Num*	*2.3*							
PAYROLL_TAX_RT	Num	1.4							
PSTI_PCT	Num	2.2							
LCP_MAX	Num	5.2							
LCP_RT	Num	0.5							
HST_PCT	Num	2.2							
CIT_V1A2	Num	1.5							
CIT_V1B2	Num	7.2							
HP_STATS_PROV_CD	Char	5			Req				'99999'

Record: CAN_TAX_PROV_PE

Description CAN Provincial Tax Parameters
Record Type SQL View
Parent Record CAN_TAX_TBL
Related Language
Build Seq 1
Owner ID Payroll
Comments Calculation formula parameters for income tax surtax and other provi
ncial taxes.

FieldName	Type	Len	Key	Srch	Req	Edit	Prompt	C	Default
EFFDT	Date	10	KD		Req				%date
PROVINCE	Char	6	KALS		Req				
CIT_V	Num	1.5							
CIT_V1A	Num	1.5							
CIT_V1B	Num	7.2							
CIT_V2	Num	1.5							
CIT_V3	Num	1.5							
CIT_V4	Num	5.2							
CIT_S	Num	5.2							
CIT_N1	Num	1.4							
CIT_N2	Num	7.2							
CIT_NCA	Num	7.2							
CIT_TH	Num	5.2							
HEALTH_INS_RT	Num	1.6							
WCB_LIMIT	Num	7.2							
PST_PCT	Num	2.2							
PST_PCT	*Num*	*2.3*							
PAYROLL_TAX_RT	Num	1.4							

PSTI_PCT	Num	2.2				
LCP_MAX	Num	5.2				
LCP_RT	Num	0.5				
HST_PCT	Num	2.2				
CIT_V1A2	Num	1.5				
CIT_V1B2	Num	7.2				
HP_STATS_PROV_CD	Char	5		Req		'99999'

Record: CAN_TAX_PROV_QC

Description CAN Provincial Tax Parameters
Record Type SQL View
Parent Record CAN_TAX_TBL
Related Language
Build Seq 1
Owner ID Payroll
Comments Calculation formula parameters for income tax surtax and other provi
ncial taxes.

FieldName	Type	Len	Key	Srch	Req	Edit	Prompt	C	Default
EFFDT	Date	10	KD		Req				%date
PROVINCE	Char	6	KALS		Req				
CIT_V	Num	1.5							
CIT_V1A	Num	1.5							
CIT_V1B	Num	7.2							
CIT_V2	Num	1.5							
CIT_V3	Num	1.5							
CIT_V4	Num	5.2							
CIT_S	Num	5.2							
CIT_N1	Num	1.4							
CIT_N2	Num	7.2							
CIT_NCA	Num	7.2							
CIT_TH	Num	5.2							
HEALTH_INS_RT	Num	1.6							
WCB_LIMIT	Num	7.2							
PST_PCT	Num	2.2							
PST_PCT	*Num*	*2.3*							
PAYROLL_TAX_RT	Num	1.4							
PSTI_PCT	Num	2.2							
LCP_MAX	Num	5.2							
LCP_RT	Num	0.5							
HST_PCT	Num	2.2							
CIT_V1A2	Num	1.5							
CIT_V1B2	Num	7.2							
HP_STATS_PROV_CD	Char	5			Req				'99999'

Record: CAN_TAX_PROV_SK

Description CAN Provincial Tax Parameters
Record Type SQL View
Parent Record CAN_TAX_TBL
Related Language
Build Seq 1
Owner ID Payroll
Comments Calculation formula parameters for income tax surtax and other provi
ncial taxes.

FieldName	Type	Len	Key	Srch	Req	Edit	Prompt	C	Default
EFFDT	Date	10	KD		Req				%date
PROVINCE	Char	6	KALS		Req				
CIT_V	Num	1.5							
CIT_V1A	Num	1.5							
CIT_V1B	Num	7.2							
CIT_V2	Num	1.5							
CIT_V3	Num	1.5							
CIT_V4	Num	5.2							
CIT_S	Num	5.2							
CIT_N1	Num	1.4							

CIT_N2	Num	7.2					
CIT_NCA	Num	7.2					
CIT_TH	Num	5.2					
HEALTH_INS_RT	Num	1.6					
WCB_LIMIT	Num	7.2					
PST_PCT	**Num**	**2.2**					
PST_PCT	***Num***	***2.3***					
PAYROLL_TAX_RT	Num	1.4					
PSTI_PCT	Num	2.2					
LCP_MAX	Num	5.2					
LCP_RT	Num	0.5					
HST_PCT	Num	2.2					
CIT_V1A2	Num	1.5					
CIT_V1B2	Num	7.2					
HP_STATS_PROV_CD	Char	5		Req			'99999'

Record: CAN_TAX_PROV_YT

Description CAN Provincial Tax Parameters
Record Type SQL View
Parent Record CAN_TAX_TBL
Related Language
Build Seq 1
Owner ID Payroll
Comments Calculation formula parameters for income tax surtax and other provi
ncial taxes.

FieldName	Type	Len	Key	Srch	Req	Edit	Prompt	C	Default
EFFDT	Date	10	KD		Req				%date
PROVINCE	Char	6	KALS		Req				
CIT_V	Num	1.5							
CIT_V1A	Num	1.5							
CIT_V1B	Num	7.2							
CIT_V2	Num	1.5							
CIT_V3	Num	1.5							
CIT_V4	Num	5.2							
CIT_S	Num	5.2							
CIT_N1	Num	1.4							
CIT_N2	Num	7.2							
CIT_NCA	Num	7.2							
CIT_TH	Num	5.2							
HEALTH_INS_RT	Num	1.6							
WCB_LIMIT	Num	7.2							
PST_PCT	**Num**	**2.2**							
PST_PCT	***Num***	***2.3***							
PAYROLL_TAX_RT	Num	1.4							
PSTI_PCT	Num	2.2							
LCP_MAX	Num	5.2							
LCP_RT	Num	0.5							
HST_PCT	Num	2.2							
CIT_V1A2	Num	1.5							
CIT_V1B2	Num	7.2							
HP_STATS_PROV_CD	Char	5			Req				'99999'

Record: CAN_TAX_PROV_ZZ

Description CAN Provincial Tax Parameters
Record Type SQL View
Parent Record CAN_TAX_TBL
Related Language
Build Seq 1
Owner ID Payroll
Comments Calculation formula parameters for income tax surtax and other provi
ncial taxes.

FieldName	Type	Len	Key	Srch	Req	Edit	Prompt	C	Default
EFFDT	Date	10	KD		Req				%date
PROVINCE	Char	6	KALS		Req				

CIT_V	Num	1.5			
CIT_V1A	Num	1.5			
CIT_V1B	Num	7.2			
CIT_V2	Num	1.5			
CIT_V3	Num	1.5			
CIT_V4	Num	5.2			
CIT_S	Num	5.2			
CIT_N1	Num	1.4			
CIT_N2	Num	7.2			
CIT_NCA	Num	7.2			
CIT_TH	Num	5.2			
HEALTH_INS_RT	Num	1.6			
WCB_LIMIT	Num	7.2			
PST_PCT	Num	2.2			
PST_PCT	*Num*	*2.3*			
PAYROLL_TAX_RT	Num	1.4			
PSTI_PCT	Num	2.2			
LCP_MAX	Num	5.2			
LCP_RT	Num	0.5			
HST_PCT	Num	2.2			
CIT_V1A2	Num	1.5			
CIT_V1B2	Num	7.2			
HP_STATS_PROV_CD	Char	5	Req		'99999'

Field Detail

Field: PST_PCT

Field Type Num
Length *6* 5
Decimal Positions *3* 2
Format UpperCase
Family Name
Display Name
Owner ID Payroll

Label ID	Short Name	Long Name	Def
PROVINCIALSALESTAX	Provincial Sale	Provincial Sales Tax	
PST_PCT	PST %	Provincial Sales Tax Percent	Yes

Field: TX810_3_BTN1

Field Type Char
Length 1
Format UpperCase
Family Name
Display Name
Owner ID Payroll

Label ID	Short Name	Long Name	Def
TX810_3_BTN1	NE/NV/NH/NJ	NE/NV/NH/NJ	Yes
TX810_3_BTN1	*NE/NV/NH/NJ/NM*	*NE/NV/NH/NJ/NM*	*Yes*

Component Detail

Component: PY_T4A_PRT_CAN (CAN)

Description T4A PDF Print
Search Record PRCSRUNCNTL
Add Search Record
Force Search No
Disable Saving Page No
Include in Navigator Yes
Mandatory SpellCheck No
Detail Page PRCSRUNCNTL
Actions Add Update
Comments Print using XML Publisher
Owner ID Payroll

Internet
Primary Action Search
Default Action Update
Default Lookup Type Basic
Allow Action Mode Sel. Yes
Link to App Page Msg 124
Nbr 62
Link to Search Page Msg 124
Nbr 63
Instructional Text Msg 124
Nbr 50
Disp Folder Tab (top) Yes
Disp Hiperlinks (bottom) Yes
Processing Mode ***Interactive*** **Deffered**
Allow Expert Entry No
Show ToolBar Yes
Save Yes
Cancel Yes
SpellCheck No
Return to List Yes
Next in List Yes
Prev in List Yes
Next Page No
Prev Page No
Refresh No
Notify Yes
View Worklist Yes
Next Worklist Yes
Prov Worklist Yes
Add Yes
Update Yes
UpdateAll No
Correction No
Show Pagebar Yes
Help Link Yes
Copy URL Link Yes
New Window Link Yes
Customize Page Link Yes

Panel Name	Item Name	Hid	Item Label	Folder Tab Label
PY_RC_PRTT4A_CAN	PY_RC_PRTT4A_CAN		Create T4A PDF Slips	

PeopleCode Detail

PeopleCode (Record): CAN_YE_T4_VW2.CAN_YE_BOX_AMT.RowInit

```
If %PanelGroup = PanelGroup.YEAR_END_SLIPS Then
   If CAN_YE_T4_VW2.BOX = "50" Then
      If CAN_YE_T4_VW2.CAN_YE_BOX_AMT = 0 Then
         CAN_YE_T4_VW2.CAN_YE_BOX_AMT.DisplayZero = False;
      Else
         CAN_YE_T4_VW2.CAN_YE_BOX_AMT.DisplayZero = True;
      End-If;
   Else
      CAN_YE_T4_VW2.CAN_YE_BOX_AMT.DisplayZero = True;
   End-If;
End-If;
```

PeopleCode (Record): DERIVED_PAY.TX810_3_BTN1.FieldChange

```
Page.RUNCTL_TAX810NE.Visible = True;
Page.RUNCTL_TAX810NV.Visible = True;
Page.RUNCTL_TAX810NH.Visible = True;
Page.RUNCTL_TAX810NJ.Visible = True;
Page.RUNCTL_TAX810NM.Visible = False;
Page.RUNCTL_TAX810NM.Visible = True;
TransferPanel(Panel.RUNCTL_TAX810NE);
```

PeopleCode (Record): PY_IC_PI_SUM_VW.OPRID.FieldFormula

```
Global boolean &GBL_SHOW_OTH_CHKS;
Global string &CURRENT_CHK;
Global number &CURR_PAYCHECK_NBR;
Global number &PAYCHECK_NBR;
Component string &CheckCountry;
Component string &Prior_Earns_Found;
Component string &Pay_Begin_Dt;
Component string &P_Earns_End;

Declare Function Get_Translate_Value PeopleCode PY_IC_FUNCLIB.FUNCLIB_01 FieldFormula;
Declare Function Populate_Ytd_Check PeopleCode PY_IC_PI_WRK.FUNCLIB_01 FieldFormula;
Declare Function Populate_Pay_Distribution PeopleCode PY_IC_PI_WRK.FUNCLIB_01 FieldFormula;
```

```
Local Rowset &RSLEAVE_VW;
Local string &Upd_Earn_Begin;
Local string &Upd_Earn_End;

/* Determine if this is the current check or not */
Function Get_Current_Chk();

   &SQL = "SELECT MAX(A.PAYCHECK_NBR), %DateOut(MAX(PAY_END_DT)) FROM PS_PAY_CHECK A WHERE A.EMPLID = '"
| PY_IC_PI_SUM_VW.EMPLID | "' AND A.COMPANY = '" | PY_IC_PI_SUM_VW.COMPANY | "' AND A.PAYCHECK_STATUS =
'F' AND A.PAYCHECK_OPTION <> 'R' AND A.CHECK_DT = (SELECT MAX(A1.CHECK_DT) FROM PS_PAY_CHECK A1 WHERE
A1.EMPLID = A.EMPLID AND A1.COMPANY = A.COMPANY AND A1.PAYCHECK_STATUS = 'F' AND A1.PAYCHECK_OPTION <>
'R') AND A.SEPCHK = (SELECT MIN(A2.SEPCHK) FROM PS_PAY_CHECK A2 WHERE A2.EMPLID = A.EMPLID AND
A2.COMPANY = A.COMPANY AND A2.PAYCHECK_STATUS = 'F' AND A2.PAYCHECK_OPTION <> 'R' AND A2.CHECK_DT =
A.CHECK_DT) AND A.FORM_ID = (SELECT MIN(A3.FORM_ID) FROM PS_PAY_CHECK A3 WHERE A3.EMPLID = A.EMPLID AND
A3.COMPANY = A.COMPANY AND A3.PAYCHECK_STATUS = 'F' AND A3.PAYCHECK_OPTION <> 'R' AND A3.CHECK_DT =
A.CHECK_DT AND A3.SEPCHK = A.SEPCHK) AND A.PAYCHECK_NBR = (SELECT MAX(A4.PAYCHECK_NBR) FROM PS_PAY_CHECK
A4 WHERE A4.EMPLID = A.EMPLID AND A4.COMPANY = A.COMPANY AND A4.PAYCHECK_STATUS = 'F' AND
A4.PAYCHECK_OPTION <> 'R' AND A4.CHECK_DT = A.CHECK_DT AND A4.SEPCHK = A.SEPCHK AND A4.FORM_ID =
A.FORM_ID)";

   SQLExec(&SQL, &PAYCHECK_NBR, &CURRENT_PAY_END_DT);

   &CURRENT_CHK = "N";
   If &PAYCHECK_NBR = PY_IC_PI_SUM_VW.PAYCHECK_NBR Then

      &CURRENT_CHK = "Y";
   End-If;

   SQLExec("SELECT A.COUNTRY FROM PS_PAYGROUP_TBL A WHERE A.COMPANY =:1 AND A.PAYGROUP =:2 AND A.EFFDT =
(SELECT MAX(A1.EFFDT) FROM PS_PAYGROUP_TBL A1 WHERE A.COMPANY = A1.COMPANY AND A.PAYGROUP = A1.PAYGROUP
AND A1.EFFDT <= %DATEIN(:3))", PY_IC_PI_SUM_VW.COMPANY, PY_IC_PI_SUM_VW.PAYGROUP,
PY_IC_PI_SUM_VW.PAY_END_DT_ALT, &CheckCountry);

   /* Determine if there is adjustment to prior period earnings */
   SQLExec("SELECT %DATEOUT(C.PAY_BEGIN_DT) FROM PS_PAY_CALENDAR C WHERE C.COMPANY = :1 AND C.PAYGROUP
=:2 AND C.PAY_END_DT = %DATEIN(:3)", PY_IC_PI_SUM_VW.COMPANY, PY_IC_PI_SUM_VW.PAYGROUP,
PY_IC_PI_SUM_VW.PAY_END_DT_ALT, &Pay_Begin_Dt);
   &Count = 0;
   SQLExec("SELECT %DATEOUT(MAX(E.EARNS_END_DT)) FROM PS_PAY_EARNINGS E WHERE E.COMPANY = :1 AND
E.PAYGROUP = :2 AND E.PAY_END_DT = %DATEIN(:3) AND E.OFF_CYCLE = :4 AND E.PAGE_NUM = :5 AND E.LINE_NUM
= :6 AND E.SEPCHK = :7 AND E.EARNS_END_DT < %DATEIN(:8) AND STATE IN ('CA','NY','CO') ",
PY_IC_PI_SUM_VW.COMPANY, PY_IC_PI_SUM_VW.PAYGROUP, PY_IC_PI_SUM_VW.PAY_END_DT_ALT,
PY_IC_PI_SUM_VW.OFF_CYCLE, PY_IC_PI_SUM_VW.PAGE_ALT, PY_IC_PI_SUM_VW.LINE_NUM, PY_IC_PI_SUM_VW.SEPCHK,
&Pay_Begin_Dt, &P_Earns_End);
   If None(&P_Earns_End) Then
      &Prior_Earns_Found = "N";
   Else
      &Prior_Earns_Found = "Y";
   End-If;

End-Function;


/*  Determine the employee benefit record number and empl record for the check  */

Function Get_Current_Bnbr();

   SQLExec("SELECT OFF_CYCLE, PAGE_NUM, LINE_NUM, %DateOut(PAY_END_DT) FROM PS_PAY_CHECK WHERE EMPLID
= :1 AND PAYCHECK_NBR = :2", PY_IC_PI_SUM_VW.EMPLID, &CURR_PAYCHECK_NBR, &OFF_CYCLE, &PAGE_NUM,
&LINE_NUM, &ALT_PAY_END_DT);

   SQLExec("SELECT DISTINCT BENEFIT_RCD_NBR, EMPL_RCD FROM PS_PAY_EARNINGS WHERE COMPANY = :1 AND
PAYGROUP = :2 AND PAY_END_DT = %DateIn(:3) AND OFF_CYCLE = :4 AND PAGE_NUM = :5 AND LINE_NUM = :6",
PY_IC_PI_SUM_VW.COMPANY, PY_IC_PI_SUM_VW.PAYGROUP, PY_IC_PI_SUM_VW.PAY_END_DT_ALT, &OFF_CYCLE,
&PAGE_NUM, &LINE_NUM, &BENEFIT_RCD_NBR, &EMPL_RCD_NBR);

End-Function;


/*---------------------------------------------------------------------*/
/*---Function - bld_gen_where_clause()---------------------------------*/
/*--This function creates the generic where clause that can be used ----*/
```

```
/*--for scroll selects of current data----------------------------------*/
/*--------------------------------------------------------------------*/
Function bld_gen_where_clause();
   remark builds the where clause and assigns it to a variable;
   &QUOTE = "'";
   &SPACE = " ";
   &GEN_CUR_WHERE = "  where COMPANY = " | &QUOTE | PY_IC_PI_SUM_VW.COMPANY_ALT | &QUOTE | " AND
PAYGROUP = " | &QUOTE | PY_IC_PI_SUM_VW.PAYGROUP | &QUOTE | " and PAY_END_DT = %datein(" | &QUOTE |
PY_IC_PI_SUM_VW.PAY_END_DT_ALT | &QUOTE | ") and OFF_CYCLE = " | &QUOTE | PY_IC_PI_SUM_VW.OFF_CYCLE |
&QUOTE | " and PAGE_NUM = " | PY_IC_PI_SUM_VW.PAGE_ALT | " and LINE_NUM = " | PY_IC_PI_SUM_VW.LINE_NUM |
" and SEPCHK = " | PY_IC_PI_SUM_VW.SEPCHK | &SPACE;
End-Function;

/*--------------------------------------------------------------------------*/
/*--This function builds a generic where clause that can be used for
scroll selects of YTD data---*/
/*--------------------------------------------------------------------------*/
Function bld_gen_ytd_where_clause();
   &QUOTE = "'";
   &SPACE = " ";
   &GEN_YTD_WHERE = "  B1 where B1.EMPLID =  " | &QUOTE | PY_IC_PI_SUM_VW.EMPLID | &QUOTE | " and
B1.COMPANY = " | &QUOTE | PY_IC_PI_SUM_VW.COMPANY_ALT | &QUOTE | " AND B1.BALANCE_ID = " | &QUOTE |
INSTALLATION.BAL_ID_FOR_CAL_YR | &QUOTE | " and B1.BALANCE_YEAR = " | PY_IC_PI_WRK.BALANCE_YEAR | " and
B1.BALANCE_PERIOD  = " | &SPACE;
End-Function;

/*--------------------------------------------------------------------------*/
/*---This function finds the balance_year,balance_qtr and balance_period to be used
to read the ytd balance data---*/
/*--------------------------------------------------------------------------*/
Function get_balance_data();
   SQLExec("select BALANCE_YEAR,BALANCE_QTR,BALANCE_PERIOD from PS_PAY_CAL_BAL_ID where COMPANY = :1 and
PAYGROUP = :2 and PAY_END_DT = %DATEIN(:3) and BALANCE_ID = :4", PY_IC_PI_SUM_VW.COMPANY_ALT,
PY_IC_PI_SUM_VW.PAYGROUP, PY_IC_PI_SUM_VW.PAY_END_DT_ALT, INSTALLATION.BAL_ID_FOR_CAL_YR,
PY_IC_PI_WRK.BALANCE_YEAR, PY_IC_PI_WRK.BALANCE_QTR, PY_IC_PI_WRK.BALANCE_PERIOD);
End-Function;

/*--------------------------------------------------------------------------*/
/*--This function prepares the rec-field names that will be used for ---*/
/*--processing the target scroll. -*/
/*--------------------------------------------------------------------------*/
Function Decide_Tgt_Fields();
   &TGT_RECORD = "RECORD" | "." | &TGT_RECORD_SCROLL;
   &TGT_CODE_FLD = &TGT_RECORD_SCROLL | "." | "pay_web_code";
   &TGT_DESCR_FLD = &TGT_RECORD_SCROLL | "." | "pay_web_descr";
   &TGT_CURAMT_FLD = &TGT_RECORD_SCROLL | "." | "pay_web_amt";
   &TGT_YTDAMT_FLD = &TGT_RECORD_SCROLL | "." | "pay_web_ytd_amt";

   If &TGT_RECORD_SCROLL = "PY_IC_PI_ERN" Or
         &TGT_RECORD_SCROLL = "PY_IC_PI_ERN_MX" Then
      &TGT_RATE_FLD = &TGT_RECORD_SCROLL | "." | "pay_web_rate";
      &TGT_CURHRS_FLD = &TGT_RECORD_SCROLL | "." | "pay_web_hrs";
      &TGT_YTDHRS_FLD = &TGT_RECORD_SCROLL | "." | "pay_web_ytd_hrs";
      &TGT_COMPRATECD_FLD = &TGT_RECORD_SCROLL | "." | "pay_web_compratecd";
      &TGT_EARN_BEGIN_FLD = &TGT_RECORD_SCROLL | "." | "earns_begin_dt";
      &TGT_EARN_END_FLD = &TGT_RECORD_SCROLL | "." | "earns_end_dt";
      &TGT_EARNS_TYPE_FLD = &TGT_RECORD_SCROLL | "." | "earns_type";
   Else
      &TGT_SORT_FLD = &TGT_RECORD_SCROLL | "." | "pay_web_sort";
   End-If;
   If &TGT_RECORD_SCROLL = "PY_IC_PI_TAX" Or
         &TGT_RECORD_SCROLL = "PY_IC_PI_TAX_MX" Then
      &TGT_CURTXGRS_FLD = &TGT_RECORD_SCROLL | "." | "pay_web_txgrs_cur";
      &TGT_YTDTXGRS_FLD = &TGT_RECORD_SCROLL | "." | "pay_web_txgrs_ytd";

   End-If;

   If &TGT_RECORD_SCROLL = "PY_IC_PI_TAX_MX" Or
         &TGT_RECORD_SCROLL = "PY_IC_PI_ERN_MX" Or
         &TGT_RECORD_SCROLL = "PY_IC_PI_BT_MX" Or
         &TGT_RECORD_SCROLL = "PY_IC_PI_AT_MX" Or
         &TGT_RECORD_SCROLL = "PY_IC_PI_ER_MX" Then
```

```
      &TGT_SEQNO = &TGT_RECORD_SCROLL | "." | "SEQNO";

   End-If;

End-Function;

/*---------------------------------------------------------------*/
/*--Function to decide fieldnames for the source records---*/
/*---------------------------------------------------------------*/
Function Decide_Src_Fields();
   &SRC_RECORD = "record." | &SRC_RECORD_SCROLL;
   Evaluate &TGT_RECORD_SCROLL
   When = "PY_IC_PI_ERN"
      &SRC_YTDHRS_FLD = &SRC_RECORD_SCROLL | ".hrs_ytd";
      &SRC_CODE_FLD = &SRC_RECORD_SCROLL | ".erncd";
      &SRC_YTDAMT_FLD = &SRC_RECORD_SCROLL | ".grs_ytd";
      Break;
   When = "PY_IC_PI_DEDN"
      &SRC_DEDCLASS_FLD = &SRC_RECORD_SCROLL | ".ded_class";
      &SRC_GARNPRS_FLD = &SRC_RECORD_SCROLL | ".spcl_process";
      &SRC_PLANTYPE_FLD = &SRC_RECORD_SCROLL | ".plan_type";
      &SRC_CODE_FLD = &SRC_RECORD_SCROLL | ".dedcd";
      &SRC_YTDAMT_FLD = &SRC_RECORD_SCROLL | ".ded_ytd";
      &SRC_CURAMT_FLD = &SRC_RECORD_SCROLL | ".ded_cur";
      &SRC_GARNID_FLD = &GARN_RECORD_SCROLL | ".garnid";
      &SRC_GARNYTD_FLD = &GARN_RECORD_SCROLL | ".ded_ytd";
      &SRC_GARNYTD_DED_FLD = &GARN_RECORD_SCROLL | ".DED_GARN_YTD";
      &SRC_GARNYTD_CFEE_FLD = &GARN_RECORD_SCROLL | ".DED_CFEE_YTD";
      &SRC_GARNYTD_PFEE_FLD = &GARN_RECORD_SCROLL | ".DED_PFEE_YTD";
      &GARN_RECORD = "record." | &GARN_RECORD_SCROLL;
      Break;
   When = "PY_IC_PI_TAX"
      &SRC_STATE_FLD = &SRC_RECORD_SCROLL | ".state";
      &SRC_LOCALITY_FLD = &SRC_RECORD_SCROLL | ".locality";
      &SRC_TAXCLASS_FLD = &SRC_RECORD_SCROLL | ".tax_class";
      &SRC_CURAMT_FLD = &SRC_RECORD_SCROLL | ".tax_cur";
      &SRC_YTDAMT_FLD = &SRC_RECORD_SCROLL | ".tax_ytd";
      Break;
   When-Other
      Break;
   End-Evaluate;
End-Function;

/*---------------------------------------------------------------------*/
/*This function updates a row in the target scroll. This only updates
some fields.Fields like descr, code,rate are one-time updates only;
but fields like hours and amount may be updated again when the same
code/rate is found. This function facilitates update of these fields--*/
/*---------------------------------------------------------------------*/
Function Update_Partial(&ROW);

   If &TYPE = "C" Then

      UpdateValue(@&TGT_CURAMT_FLD, &ROW, &CUR_AMT);
      If &TGT_RECORD_SCROLL = "PY_IC_PI_ERN" Then
         UpdateValue(@&TGT_CURHRS_FLD, &ROW, &CUR_HRS);
         If &Upd_Earn_Begin = "Y" Then
            UpdateValue(@&TGT_EARN_BEGIN_FLD, &ROW, &EARNS_BEGIN_DT);
         End-If;
         If &Upd_Earn_End = "Y" Then
            UpdateValue(@&TGT_EARN_END_FLD, &ROW, &EARNS_END_DT);
         End-If;
      Else
         If &TGT_RECORD_SCROLL = "PY_IC_PI_TAX" Then
            UpdateValue(@&TGT_CURTXGRS_FLD, &ROW, &CUR_TAXABLE_GROSS);
         End-If;
      End-If;
   Else
      If &TYPE = "Y" Then
         UpdateValue(@&TGT_YTDAMT_FLD, &ROW, &YTD_AMT);
         If &TGT_RECORD_SCROLL = "PY_IC_PI_ERN" Then
            UpdateValue(@&TGT YTDHRS FLD, &ROW, &YTD HRS);
```

```
         Else

            If &TGT_RECORD_SCROLL = "PY_IC_PI_TAX" Then
               UpdateValue(@&TGT_YTDTXGRS_FLD, &ROW, &YTD_TAXABLE_GROSS);
            End-If;
         End-If;
      End-If;

   End-If;


End-Function;

/*---------------------------------------------------------------------*/
/*This function updates a row in the target scroll. This only updates
ONE field, seqno.--*/
/*---------------------------------------------------------------------*/
Function Update_Partial_MX(&ROW);
   If &TGT_RECORD_SCROLL = "PY_IC_PI_TAX_MX" Or
         &TGT_RECORD_SCROLL = "PY_IC_PI_ERN_MX" Or
         &TGT_RECORD_SCROLL = "PY_IC_PI_BT_MX" Or
         &TGT_RECORD_SCROLL = "PY_IC_PI_AT_MX" Or
         &TGT_RECORD_SCROLL = "PY_IC_PI_ER_MX" Then
      UpdateValue(@&TGT_SEQNO, &ROW, &ROW);
   End-If;

End-Function;


/*---------------------------------------------------------------------*/
/*--This function updates values in the target scroll. The target ------*/
/*--scroll can be either PY_IC_PI_ERN (Earnings), PY_IC_PI_TAX (Taxes)--*/
/*--or PY_IC_PI_DEDN (Deductions). All values are updated. The row no. is
row no. that is passed to this function. -- */
/*---------------------------------------------------------------------*/
Function update_all(&ROW);
   UpdateValue(@&TGT_CODE_FLD, &ROW, &CODE);
   If &TGT_RECORD_SCROLL = "PY_IC_PI_ERN" Or
         &TGT_RECORD_SCROLL = "PY_IC_PI_ERN_MX" Then
      UpdateValue(@&TGT_RATE_FLD, &ROW, &RATE);
      UpdateValue(@&TGT_COMPRATECD_FLD, &ROW, &COMPRATECD);
      UpdateValue(@&TGT_EARN_BEGIN_FLD, &ROW, &EARNS_BEGIN_DT);
      UpdateValue(@&TGT_EARN_END_FLD, &ROW, &EARNS_END_DT);
   Else
      UpdateValue(@&TGT_DESCR_FLD, &ROW, &DESCR);
      UpdateValue(@&TGT_SORT_FLD, &ROW, &SORT);
   End-If;
   Update_Partial(&ROW);
End-Function;

Function update_ytd_ern_values(&ROW);
   UpdateValue(@&TGT_CODE_FLD, &ROW, &CODE);
   UpdateValue(@&TGT_YTDAMT_FLD, &ROW, &YTD_AMT);
   UpdateValue(@&TGT_YTDHRS_FLD, &ROW, &YTD_HRS);
End-Function;

Function Test_For_Rate_Recalc(&CODE, &CUR_HRS, &CUR_AMT, &RATE, &RateUsed);
   If &CUR_AMT <> 0 And
         &CUR_HRS <> 0 And
         &CUR_AMT <> 0 Then
      &W_CUR_AMT = &RATE * &CUR_HRS;
      &W_CUR_AMT = &W_CUR_AMT - &CUR_AMT;

      SQLExec("SELECT ET.FACTOR_HRS_ADJ, ET.FACTOR_ERN_ADJ FROM PS_EARNINGS_TBL ET WHERE ET.ERNCD  = :1
AND ET.EFFDT  = (SELECT MAX(EFFDT) FROM PS_EARNINGS_TBL ET1 WHERE  ET1.ERNCD  = ET.ERNCD AND ET1.EFFDT
<= %DATEIN(:2))", &CODE, %Date, &FactorHrsAdj, &FactorErnAdj);

      If &W_CUR_AMT > 0.01 Or
            &W_CUR_AMT < - 0.01 Then

         If &RateUsed = "F" Or
               &RateUsed = "A" Or
```

```
               &FactorHrsAdj <> 0 Or
               &FactorErnAdj <> 0 Then
            &NeedXfootMsg = "Y";
         End-If;
      End-If;
   End-If;
End-Function;

Function fetch_reg_hr_earnings();
   &CODE = FetchValue(PY_IC_PI_ERN_VW.ERNCD_REG_HRS, &I);
   &ERNCD_REG_HRS = &CODE;
   If &Prior_Earns_Found = "Y" Then
      &EARNS_BEGIN_DT = FetchValue(PY_IC_PI_ERN_VW.EARNS_BEGIN_DT, &I);
      &EARNS_END_DT = FetchValue(PY_IC_PI_ERN_VW.EARNS_END_DT, &I);
      If &EARNS_END_DT >= &Pay_Begin_Dt Then
         &EARNS_BEGIN_DT = "";
         &EARNS_END_DT = "";
      End-If;
   Else
      &EARNS_BEGIN_DT = "";
      &EARNS_END_DT = "";
   End-If;
   &CUR_HRS = FetchValue(PY_IC_PI_ERN_VW.REG_HRS, &I);
   &CUR_AMT = FetchValue(PY_IC_PI_ERN_VW.REG_HRLY_EARNS, &I);
   &COMPRATECD = FetchValue(PY_IC_PI_ERN_VW.COMP_RATECD_REG, &I);
   &RATE = FetchValue(DERIVED_PAY.COMPRATE_USED_REG, &I);
   &Total_Current_Hrs = &Total_Current_Hrs + &CUR_HRS;

   &RateUsed = " ";
   Test_For_Rate_Recalc(&CODE, &CUR_HRS, &CUR_AMT, &RATE, &RateUsed);
End-Function;

Function fetch_ot_hr_earnings();
   &CODE = FetchValue(PY_IC_PI_ERN_VW.ERNCD_OT_HRS, &I);
   &ERNCD_OT_HRS = &CODE;
   If &Prior_Earns_Found = "Y" Then
      &EARNS_BEGIN_DT = FetchValue(PY_IC_PI_ERN_VW.EARNS_BEGIN_DT, &I);
      &EARNS_END_DT = FetchValue(PY_IC_PI_ERN_VW.EARNS_END_DT, &I);
      If &EARNS_END_DT >= &Pay_Begin_Dt Then
         &EARNS_BEGIN_DT = "";
         &EARNS_END_DT = "";
      End-If;
   Else
      &EARNS_BEGIN_DT = "";
      &EARNS_END_DT = "";
   End-If;
   &CUR_HRS = FetchValue(PY_IC_PI_ERN_VW.OT_HRS, &I);
   &CUR_AMT = FetchValue(PY_IC_PI_ERN_VW.OT_HRLY_EARNS, &I);
   &COMPRATECD = FetchValue(PY_IC_PI_ERN_VW.COMP_RATECD_OT, &I);
   &RATE = FetchValue(DERIVED_PAY.COMPRATE_USED_OT, &I);
   &Total_Current_Hrs = &Total_Current_Hrs + &CUR_HRS;

   &RateUsed = FetchValue(PY_IC_PI_ERN_VW.RATE_USED, &I);
   Test_For_Rate_Recalc(&CODE, &CUR_HRS, &CUR_AMT, &RATE, &RateUsed);
End-Function;

Function fetch_reg_sal_earnings();
   &CODE = FetchValue(PY_IC_PI_ERN_VW.ERNCD_REG_EARNS, &I);
   &FLSA_STATUS = FetchValue(PY_IC_PI_ERN_VW.FLSA_STATUS, &I);
   &ERN_STATE = FetchValue(PY_IC_PI_ERN_VW.STATE, &I);
   &ERNCD_REG_EARNS = &CODE;
   If &Prior_Earns_Found = "Y" Then
      &EARNS_BEGIN_DT = FetchValue(PY_IC_PI_ERN_VW.EARNS_BEGIN_DT, &I);
      &EARNS_END_DT = FetchValue(PY_IC_PI_ERN_VW.EARNS_END_DT, &I);
      If &EARNS_END_DT >= &Pay_Begin_Dt Then
         &EARNS_BEGIN_DT = "";
         &EARNS_END_DT = "";
      End-If;
   Else
      &EARNS_BEGIN_DT = "";
      &EARNS_END_DT = "";
   End-If;
```

```
   If (&FLSA_STATUS = "N" Or
            &FLSA_STATUS = "V") And
          (&ERN_STATE = "NY" Or
            &ERN_STATE = "CA") Then
      &CUR_HRS = FetchValue(PY_IC_PI_ERN_VW.REG_EARN_HRS, &I);
      &RATE = FetchValue(PY_IC_PI_ERN_VW.HOURLY_RT, &I);
   Else
      &CUR_HRS = 0;
      &RATE = 0;
   End-If;
   &CUR_AMT = FetchValue(PY_IC_PI_ERN_VW.REG_EARNS, &I);
   &COMPRATECD = " ";
   &Total_Current_Hrs = &Total_Current_Hrs + &CUR_HRS;
End-Function;

Function fetch_other_earnings();
   &CODE = FetchValue(PY_IC_PI_ERN_VW.ERNCD_REG_EARNS, &I, PY_IC_PI_OE_VW.ERNCD, &K);
   If &Prior_Earns_Found = "Y" Then
      &EARNS_BEGIN_DT = FetchValue(PY_IC_PI_ERN_VW.EARNS_BEGIN_DT, &I);
      &EARNS_END_DT = FetchValue(PY_IC_PI_ERN_VW.EARNS_END_DT, &I);
      If &EARNS_END_DT >= &Pay_Begin_Dt Then
         &EARNS_BEGIN_DT = "";
         &EARNS_END_DT = "";
      End-If;
   Else
      &EARNS_BEGIN_DT = "";
      &EARNS_END_DT = "";
   End-If;
   &RATE = FetchValue(PY_IC_PI_ERN_VW.HOURLY_RT, &I);
   &CUR_HRS = FetchValue(PY_IC_PI_ERN_VW.OT_HRS, &I, DERIVED_PAY.OTH_HRS, &K);
   &CUR_AMT = FetchValue(PY_IC_PI_ERN_VW.REG_EARNS, &I, PY_IC_PI_OE_VW.OTH_EARNS, &K);
   &PAY = FetchValue(PY_IC_PI_ERN_VW.REG_EARNS, &I, PY_IC_PI_OE_VW.OTH_PAY, &K);
   &COMPRATECD = FetchValue(PY_IC_PI_ERN_VW.REG_EARNS, &I, PY_IC_PI_OE_VW.COMP_RATECD, &K);
   &RATE = FetchValue(PY_IC_PI_ERN_VW.REG_EARNS, &I, DERIVED_PAY.COMPRATE_USED, &K);
   &Total_Current_Hrs = &Total_Current_Hrs + &CUR_HRS;

   &RateUsed = FetchValue(PY_IC_PI_ERN_VW.REG_EARNS, &I, PY_IC_PI_OE_VW.RATE_USED, &K);
   Test_For_Rate_Recalc(&CODE, &CUR_HRS, &CUR_AMT, &RATE, &RateUsed);
End-Function;

/*---------------------------------------------------------------------*/
/*--This function updates the target earnings scroll (PY_IC_PI_ERN).
It compares the code ,comprate code and rate field from PY_IC_PI_ERN
with that read from the source earnings scrolls (PY_IC_PI_ERN_VW and
PY_IC_PI_OE_VW). If a match is found, it updates the amount and
hours for that row. If a match is not found, it inserts a row in
the target earnings scroll (PY_IC_PI_ERN).--*/
/*---------------------------------------------------------------------*/
Function update_earnings();
   &Upd_Earn_Begin = "N";
   &Upd_Earn_End = "N";

   If &CUR_HRS <> 0 Or
         &CUR_AMT <> 0 Then
      &NO_OF_TGT_EARNS = ActiveRowCount(@&TGT_RECORD);
      For &L = 1 To &NO_OF_TGT_EARNS
         &FOUND = False;
         &STORED_CODE = FetchValue(@&TGT_CODE_FLD, &L);
         &STORED_EARN_BEGIN = FetchValue(@&TGT_EARN_BEGIN_FLD, &L);
         &STORED_EARN_END = FetchValue(@&TGT_EARN_END_FLD, &L);
         &STORED_CUR_HRS = FetchValue(@&TGT_CURHRS_FLD, &L);
         &STORED_RATE = FetchValue(@&TGT_RATE_FLD, &L);
         &STORED_CUR_AMT = FetchValue(@&TGT_CURAMT_FLD, &L);
         &STORED_COMPRATECD = FetchValue(@&TGT_COMPRATECD_FLD, &L);
         If None(&STORED_CODE) Then
            &FOUND = True;
            update_all(&L);
            Break;
         Else
            If &STORED_CODE = &CODE And
                  &STORED_RATE = &RATE And
                  &STORED COMPRATECD = &COMPRATECD And
```

```
                  ((&STORED_EARN_BEGIN = &EARNS_BEGIN_DT And
                        &STORED_EARN_END = &EARNS_END_DT) Or
                     (&STORED_EARN_END <> "" And
                        &EARNS_END_DT <> "" And
                        &EARNS_END_DT < &Pay_Begin_Dt)) Then

               If &STORED_EARN_END <> "" And
                     &EARNS_END_DT <> "" And
                     &EARNS_END_DT < &Pay_Begin_Dt Then
                  If &EARNS_BEGIN_DT <> "" And
                        &EARNS_BEGIN_DT < &STORED_EARN_BEGIN Then
                     &Upd_Earn_Begin = "Y";
                  End-If;

                  If &EARNS_END_DT <> "" And
                        &EARNS_END_DT > &STORED_EARN_END Then
                     &Upd_Earn_End = "Y";
                  End-If;
               End-If;
               &CUR_HRS = &CUR_HRS + &STORED_CUR_HRS;
               &CUR_AMT = &CUR_AMT + &STORED_CUR_AMT;
               &FOUND = True;
               Update_Partial(&L);
               Break;
            End-If;
         End-If;
      End-For;
      If &FOUND = False Then
         InsertRow(@&TGT_RECORD, &NO_OF_TGT_EARNS);
         update_all((&NO_OF_TGT_EARNS + 1));
      End-If;
      &CUR_HRS = 0;
      &CUR_AMT = 0;
   End-If;
End-Function;

/*---------------------------------------------------------------------*/
/*--This function reads the source earnings scrolls (PY_IC_PI_ERN_VW
and PY_IC_PI_OE_VW). For every row read, it then calls the function
update_earnings() to update the target scroll (PY_IC_PI_ERN) with
the value read. ----------------------------------------------------*/
/*---------------------------------------------------------------------*/
Function get_earnings();
   &NO_OF_SRC_ERNS = ActiveRowCount(Record.PY_IC_PI_ERN_VW);
   For &I = 1 To &NO_OF_SRC_ERNS
      For &J = 1 To 3
         Evaluate &J
         When = 1
            fetch_reg_hr_earnings();
            update_earnings();
            Break;
         When = 2
            fetch_ot_hr_earnings();
            update_earnings();
            Break;
         When = 3
            fetch_reg_sal_earnings();
            update_earnings();
            Break;
         End-Evaluate;
      End-For;
      &NO_OF_SRC_OTHERNS = ActiveRowCount(Record.PY_IC_PI_ERN_VW, &I, Record.PY_IC_PI_OE_VW);
      For &K = 1 To &NO_OF_SRC_OTHERNS
         fetch_other_earnings();
         update_earnings();
      End-For;
   End-For;
End-Function;

/*---------------------------------------------------------------------------*/
/*--This function updates the target earnings scroll (PY_IC_PI_ERN).
It compares the code field from PY IC PI ERN
```

```
with that read from the source earnings scrolls. If a match is found, it updates the amount and hours
for that row. If a match is not found, it inserts a row in
the target earnings scroll (PY_IC_PI_ERN).--*/
/*-------------------------------------------------------------------------------*/
Function update_ytd_earnings();
   &RATE = 0;
   &COMPRATECD = "";
   &NO_OF_TGT_EARNS = ActiveRowCount(@&TGT_RECORD);
   For &TGT_COUNTER = 1 To &NO_OF_TGT_EARNS
      &FOUND = False;
      &STORED_CODE = FetchValue(@&TGT_CODE_FLD, &TGT_COUNTER);
      &STORED_YTD_HRS = FetchValue(@&TGT_YTDHRS_FLD, &TGT_COUNTER);
      &STORED_YTD_AMT = FetchValue(@&TGT_YTDAMT_FLD, &TGT_COUNTER);
      If None(&STORED_CODE) Then
         &FOUND = True;
         update_all(&TGT_COUNTER);
         Break;
      Else
         If &STORED_CODE = &CODE Then
            &YTD_HRS = &YTD_HRS + &STORED_YTD_HRS;
            &YTD_AMT = &YTD_AMT + &STORED_YTD_AMT;
            &FOUND = True;
            Update_Partial(&TGT_COUNTER);
            Break;
         End-If;
      End-If;
   End-For;
   If &FOUND = False Then
      InsertRow(@&TGT_RECORD, &NO_OF_TGT_EARNS);
      update_all((&NO_OF_TGT_EARNS + 1));
   End-If;
   &YTD_AMT = 0;
   &YTD_HRS = 0;
End-Function;

/*-------------------------------------------------------------------------*/
/*---This function reads the ytd earnings from the source balance record.
It then calls the function to update the target scroll - PY_IC_PI_ERN
with the value read.----*/
/*-------------------------------------------------------------------------*/
Function get_ytd_earnings();
   &YTD_ROWS = ActiveRowCount(@&SRC_RECORD);
   For &YTD_PTR = 1 To &YTD_ROWS
      &CODE = FetchValue(@&SRC_CODE_FLD, &YTD_PTR);
      &YTD_HRS = FetchValue(@&SRC_YTDHRS_FLD, &YTD_PTR);
      &YTD_AMT = FetchValue(@&SRC_YTDAMT_FLD, &YTD_PTR);
      update_ytd_earnings();
   End-For;
End-Function;

/*-------------------------------------------------------------------------*/
/*--This function puts a value of zero in the rate field for those
earnings codes that do not have hours associated with them. ---------*/
/*-------------------------------------------------------------------------*/
Function decide_rate();
   &NO_OF_TGT_EARNS = ActiveRowCount(Record.PY_IC_PI_ERN);
   For &I = 1 To &NO_OF_TGT_EARNS
      &HRS = FetchValue(PY_IC_PI_ERN.PAY_WEB_HRS, &I);
      If &HRS = 0 Then
         UpdateValue(PY_IC_PI_ERN.PAY_WEB_RATE, &I, 0);
      End-If;
   End-For;
End-Function;

/*-------------------------------------------------------------------------*/
/*--This function updates the intended target scroll. It could be
PY_IC_PI_DEDN (Deductions) or PY_IC_PI_TAX (Taxes). It reads code
from the target scroll and compares it with the code obtained from
the source scroll. If there is a match, it updates that row, else
it inserts a new row.-----------------------------------------------*/
/*-------------------------------------------------------------------------*/
Function update array();
```

```
   &NO_OF_TGT_ROWS = ActiveRowCount(@&TGT_RECORD);
   &FOUND = False;
   For &K = 1 To &NO_OF_TGT_ROWS
      &STORED_CODE = FetchValue(@&TGT_CODE_FLD, &K);
      If None(&STORED_CODE) Then
         &FOUND = True;
         update_all(&K);
      Else
         If &STORED_CODE = &CODE Then
            &FOUND = True;
            &STORED_CUR_AMT = FetchValue(@&TGT_CURAMT_FLD, &K);
            &CUR_AMT = &STORED_CUR_AMT + &CUR_AMT;
            &STORED_YTD_AMT = FetchValue(@&TGT_YTDAMT_FLD, &K);
            &YTD_AMT = &STORED_YTD_AMT + &YTD_AMT;
            If &TGT_RECORD_SCROLL = "PY_IC_PI_TAX" Then
               &STORED_CUR_GROSS = FetchValue(@&TGT_CURTXGRS_FLD, &K);
               &CUR_TAXABLE_GROSS = &STORED_CUR_GROSS + &CUR_TAXABLE_GROSS;
               &STORED_YTD_GROSS = FetchValue(@&TGT_YTDTXGRS_FLD, &K);
               &YTD_TAXABLE_GROSS = &STORED_YTD_GROSS + &YTD_TAXABLE_GROSS;
            End-If;
            Update_Partial(&K);
            Break;
         End-If;
      End-If;
   End-For;
   If &FOUND = False Then
      InsertRow(@&TGT_RECORD, &NO_OF_TGT_ROWS);
      update_all((&NO_OF_TGT_ROWS + 1));
   End-If;
   /*      If Not (&TGT_RECORD_SCROLL = "PY_IC_PI_TAX" And
            PAYGROUP_TBL.COUNTRY = "CAN") Then */
   &CUR_AMT = 0;
   &YTD_AMT = 0;
   &CUR_TAXABLE_GROSS = 0;
   &YTD_TAXABLE_GROSS = 0;
   /*      End-If; */
End-Function;


/*---------------------------------------------------------------------------*/
/*---This function sets a value for the tax descr and sort variables.--*/
/*---------------------------------------------------------------------------*/
Function Get_Tax_Descr_and_Sort();
   If &STATE = "$U" Then
      &DESCR = "Fed";
      &SORT = "E";
   Else
      &SORT = "F";
      &DESCR = &STATE;
      If All(&LOCALITY) Then
         &LOCALITY_ABBRV = FetchValue(LOCAL_TAX_TBL2.LOCALITY_ABBRV, &SRC_PTR);
         &DESCR = &DESCR | " " | &LOCALITY_ABBRV;
      End-If;
   End-If;

   &DESCR = &DESCR | " " | &TAX_CLASS_XLAT;

End-Function;

/*---------------------------------------------------------------------------*/
/*--This function reads the source taxes scroll for US taxes.
For every row read, it then calls the function
update_array() to update the target scroll (PY_IC_PI_TAX) . The source scroll
could either be for the current taxes or for ytd taxes. The variable &TYPE
determines this. A value of 'C' indicates current, while a value of 'Y'
indicates YTD.-------*/
/*---------------------------------------------------------------------------*/
Function get_us_taxes();
   For &SRC_PTR = 1 To ActiveRowCount(@&SRC_RECORD)
      &STATE = RTrim(FetchValue(@&SRC_STATE_FLD, &SRC_PTR));
      &LOCALITY = RTrim(FetchValue(@&SRC_LOCALITY_FLD, &SRC_PTR));
      &TAX_CLASS = FetchValue(@&SRC_TAXCLASS_FLD, &SRC_PTR);
```

```
      &CODE = &STATE | &LOCALITY | &TAX_CLASS;
      &TAX_CLASS_XLAT = Get_Translate_Value("S", &TAX_CLASS_FLD, &TAX_CLASS, &OPR_LANG_CD,
PY_IC_PI_WRK.EFFDT);
      If &TYPE = "Y" Then
         If &STATE = "PA" And
               None(&LOCALITY) And
               &TAX_CLASS = "P" Then
            /* Do not display memo row for State of 'PA' */
         Else
            &YTD_AMT = FetchValue(@&SRC_YTDAMT_FLD, &SRC_PTR);
            Get_Tax_Descr_and_Sort();
            /*         &YTD_TAXABLE_GROSS = 0;*/
            &YTD_TAXABLE_GROSS = FetchValue(TAX_BALANCE.TXGRS_YTD, &SRC_PTR);
            If &STATE = "$U" And
                  &TAX_CLASS = "H" Then
               PY_IC_PI_WRK.PAY_WEB_TXGRS_YTD = PY_IC_PI_WRK.PAY_WEB_TXGRS_YTD + &YTD_TAXABLE_GROSS;
            End-If;
            update_array();
         End-If;
      Else
         If &TYPE = "C" Then
            &CUR_AMT = FetchValue(@&SRC_CURAMT_FLD, &SRC_PTR);
            /*            &CUR_TAXABLE_GROSS = 0;*/
            &CUR_TAXABLE_GROSS = FetchValue(PAY_TAX.TXGRS_CUR, &SRC_PTR);
            If &STATE = "$U" And
                  &TAX_CLASS = "H" Then
               PY_IC_PI_WRK.PAY_WEB_TXGRS_CUR = PY_IC_PI_WRK.PAY_WEB_TXGRS_CUR + &CUR_TAXABLE_GROSS;
            End-If;
            update_array();
         End-If;
      End-If;
   End-For;
End-Function;

/*---------------------------------------------------------------------*/
/*---This function gets the current tax amounts for Canadian taxes---*/
/*---------------------------------------------------------------------*/
Function Get_Can_Cur();
   For &CUR_PTR = 1 To &CUR_ROWS
      If &TAX_CLASS_CAN = FetchValue(PAY_TAX_CAN.TAX_CLASS_CAN, &CUR_PTR) Then
         &CUR_AMT = &CUR_AMT + FetchValue(PAY_TAX_CAN.TAX_CUR, &CUR_PTR);
         &CUR_TAXABLE_GROSS = &CUR_TAXABLE_GROSS + FetchValue(PAY_TAX_CAN.TXGRS_CUR, &CUR_PTR);
         If &TAX_CLASS_CAN = "CIT" Then
            PY_IC_PI_WRK.PAY_WEB_TXGRS_CUR = PY_IC_PI_WRK.PAY_WEB_TXGRS_CUR + FetchValue
(PAY_TAX_CAN.TXGRS_CUR, &CUR_PTR);
         End-If;
      End-If;
   End-For;
End-Function;

/*---------------------------------------------------------------------*/
/*---This function gets the YTD tax amounts for Canadian taxes---*/
/*---------------------------------------------------------------------*/
Function Get_Can_Ytd();
   For &YTD_PTR = 1 To &YTD_ROWS
      If &TAX_CLASS_CAN = FetchValue(CAN_TAX_BALANCE.TAX_CLASS_CAN, &YTD_PTR) Then
         &YTD_AMT = &YTD_AMT + FetchValue(CAN_TAX_BALANCE.TAX_YTD, &YTD_PTR);
         &YTD_TAXABLE_GROSS = &YTD_TAXABLE_GROSS + FetchValue(CAN_TAX_BALANCE.TXGRS_YTD, &YTD_PTR);
         If &TAX_CLASS_CAN = "CIT" Then
            PY_IC_PI_WRK.PAY_WEB_TXGRS_YTD = PY_IC_PI_WRK.PAY_WEB_TXGRS_YTD + FetchValue
(CAN_TAX_BALANCE.TXGRS_YTD, &YTD_PTR);;
         End-If;
      End-If;
   End-For;
End-Function;

/*---------------------------------------------------------------------*/
/*--This function decides the value of &TAX_CLASS_CAN. There can be 9 different
values. For every value, it then calls the functions to get current and YTD
canadian taxes. After this, it calls the function update_array() to update
the target scroll (PY_IC_PI_TAX) ----------*/
/*---------------------------------------------------------------------*/
```

```
Function get_can_taxes();
   &SORT = "Z";
   &CUR_ROWS = ActiveRowCount(Record.PAY_TAX_CAN);
   &YTD_ROWS = ActiveRowCount(Record.CAN_TAX_BALANCE);
   &CIT_DESCR = MsgGetText(2001, 1, "CIT");
   &CPP_DESCR = MsgGetText(2001, 2, "CPP");
   &EI_DESCR = MsgGetText(2001, 3, "EI");
   &QIT_DESCR = MsgGetText(2001, 4, "QIT");
   &QPP_DESCR = MsgGetText(2001, 5, "QPP");
   &PYT_DESCR = MsgGetText(2001, 6, "NWT Payroll");
   /* &QPIP_DESCR = MsgGetText(2001, 7, "QPIP"); */
   &QPIP_DESCR = "QPIP";
   For &TAXTYPE = 1 To 9
      Evaluate &TAXTYPE
      When = 1
         &TAX_CLASS_CAN = "CIT";
         &DESCR = &CIT_DESCR;
         Break;
      When = 2
         &TAX_CLASS_CAN = "T4A";
         /*      &DESCR = &T4A_DESCR;*/
         Break;
      When = 3
         &TAX_CLASS_CAN = "CPP";
         &DESCR = &CPP_DESCR;
         Break;
      When = 4
         &TAX_CLASS_CAN = "EIE";
         &DESCR = &EI_DESCR;
         Break;
      When = 5
         &TAX_CLASS_CAN = "PYT";
         &DESCR = &PYT_DESCR;
         Break;
      When = 6
         &TAX_CLASS_CAN = "QIT";
         &DESCR = &QIT_DESCR;
         Break;
      When = 7
         &TAX_CLASS_CAN = "RV2";
         /*         &DESCR = &RV2_DESCR;*/
         Break;
      When = 8
         &TAX_CLASS_CAN = "QPP";
         &DESCR = &QPP_DESCR;
         Break;
      When = 9
         &TAX_CLASS_CAN = "QIE";
         &DESCR = &QPIP_DESCR;
         Break;
      End-Evaluate;
      &YTD_AMT = 0;
      &CUR_AMT = 0;
      &YTD_TAXABLE_GROSS = 0;
      &CUR_TAXABLE_GROSS = 0;
      If &TAX_CLASS_CAN = "T4A" Then
         &CODE = "CIT";
      Else
         If &TAX_CLASS_CAN = "RV2" Then
            &CODE = "QIT";
         Else
            &CODE = &TAX_CLASS_CAN;
         End-If;
      End-If;
      Get_Can_Ytd();
      If &YTD_AMT <> 0 Then
         &TYPE = "Y";
         update_array();
      End-If;
      Get_Can_Cur();
      If &CUR_AMT <> 0 Then
         &TYPE = "C";
```

```
         update_array();
      End-If;
   End-For;
End-Function;

/*---------------------------------------------------------------------*/
/*--This function sets a value for the Garnishment code---*/
/*---------------------------------------------------------------------*/
Function Get_Garn_Code(&AMT_TYPE);
   &CODE = "GG" | &GARNID | &DED_CLASS1 | &AMT_TYPE;
End-Function;

/*---------------------------------------------------------------------*/
/*--This function reads the source garnishments scroll. It could either
be current or YTD data. This depends on the value of &TYPE. A value of
'C' indicates current while 'Y' indicates YTD data.
For every row read, it then calls the function update_array() to
update the target scroll (PY_IC_PI_DEDN) ---------------------------*/
/*---------------------------------------------------------------------*/
Function get_garnishments();
   &DESCR_SAVE = &DESCR;
   For &SRC_GARN_PTR = 1 To ActiveRowCount(@&SRC_RECORD, &SRC_PTR, @&GARN_RECORD);
      &GARNID = RTrim(FetchValue(@&SRC_RECORD, &SRC_PTR, @&SRC_GARNID_FLD, &SRC_GARN_PTR));
      If &TYPE = "Y" Then
         &YTD_AMT = FetchValue(@&SRC_RECORD, &SRC_PTR, @&SRC_GARNYTD_FLD, &SRC_GARN_PTR);
         &YTD_DED_AMT = FetchValue(@&SRC_RECORD, &SRC_PTR, @&SRC_GARNYTD_DED_FLD, &SRC_GARN_PTR);
         &YTD_CFEE_AMT = FetchValue(@&SRC_RECORD, &SRC_PTR, @&SRC_GARNYTD_CFEE_FLD, &SRC_GARN_PTR);
         &YTD_PFEE_AMT = FetchValue(@&SRC_RECORD, &SRC_PTR, @&SRC_GARNYTD_PFEE_FLD, &SRC_GARN_PTR);
         &GARN_TYPE = FetchValue(@&SRC_RECORD, &SRC_PTR, GARN_SPEC.GARN_TYPE, &SRC_GARN_PTR);
         &GARN_TYPE_XLAT = Get_Translate_Value("S", &GARN_TYPE_FLD, &GARN_TYPE, &OPR_LANG_CD,
PY_IC_PI_WRK.EFFDT);
         /*-- garnishment description --*/
         If &YTD_AMT = &YTD_DED_AMT Then
            If &CheckCountry = "CAN" Then
               &DESCR = &DEDCD | " " | &GARNID | " " | &GARN_TYPE_XLAT;
            Else
               &DESCR = &DESCR_SAVE | " - " | &GARN_TYPE_XLAT;
            End-If;
            Get_Garn_Code("");
            &SORT = "B";
            update_array();
         Else
            /* Garnishment Amount */
            &Garn_Amt_Txt = MsgGetExplainText(2001, 844, "(Amount)");
            If &CheckCountry = "CAN" Then
               &DESCR = &DEDCD | " " | &GARNID | " " | &GARN_TYPE_XLAT | " " | &Garn_Amt_Txt;
            Else
               &DESCR = &DESCR_SAVE | " - " | &GARN_TYPE_XLAT | " " | &Garn_Amt_Txt;
            End-If;
            &YTD_AMT = &YTD_DED_AMT;
            Get_Garn_Code("");
            &SORT = "B";
            update_array();

            /* Garnishment Company Fee */
            &Garn_CFee_Txt = MsgGetExplainText(2001, 845, "(Co. Fee)");
            If &CheckCountry = "CAN" Then
               &DESCR = &DEDCD | " " | &GARNID | " " | &GARN_TYPE_XLAT | " " | &Garn_CFee_Txt;
            Else
               &DESCR = &DESCR_SAVE | " - " | &GARN_TYPE_XLAT | " " | &Garn_CFee_Txt;
            End-If;
            &YTD_AMT = &YTD_CFEE_AMT;
            Get_Garn_Code("C");
            &SORT = "B";
            update_array();

            /* Garnishment Payee Fee */
            &Garn_PFee_Txt = MsgGetExplainText(2001, 846, "(Payee Fee)");
            If &CheckCountry = "CAN" Then
               &DESCR = &DEDCD | " " | &GARNID | " " | &GARN_TYPE_XLAT | " " | &Garn_PFee_Txt;
            Else
               &DESCR = &DESCR_SAVE | " - " | &GARN_TYPE_XLAT | " " | &Garn_PFee_Txt;
```

```
            End-If;
            &YTD_AMT = &YTD_PFEE_AMT;
            Get_Garn_Code("P");
            &SORT = "B";
            update_array();
         End-If;
      Else
         If &TYPE = "C" Then
            &CUR_AMT = FetchValue(@&SRC_RECORD, &SRC_PTR, PAY_GARNISH.DEDUCT_AMT, &SRC_GARN_PTR);
            &CUR_GARN_AMT = FetchValue(@&SRC_RECORD, &SRC_PTR, PAY_GARNISH.DEDUCT_GARN_AMT,
&SRC_GARN_PTR);
            &CUR_CFEE_AMT = FetchValue(@&SRC_RECORD, &SRC_PTR, PAY_GARNISH.DEDUCT_CMPNY_FEE,
&SRC_GARN_PTR);
            &CUR_PFEE_AMT = FetchValue(@&SRC_RECORD, &SRC_PTR, PAY_GARNISH.DEDUCT_PAYEE_FEE,
&SRC_GARN_PTR);
         End-If;
         &CUR_AMT = &CUR_GARN_AMT;
         Get_Garn_Code("");
         &SORT = "B";
         update_array();

         &CUR_AMT = &CUR_CFEE_AMT;
         Get_Garn_Code("C");
         &SORT = "B";
         update_array();

         &CUR_AMT = &CUR_PFEE_AMT;
         Get_Garn_Code("P");
         &SORT = "B";
         update_array();
      End-If;
   End-For;
End-Function;

/*---------------------------------------------------------------------------*/
/*---This function is used to set a value for the deduction code and sort
variables---*/
/*---------------------------------------------------------------------------*/
Function Get_Dedn_Code_And_Sort();
   Evaluate &DED_CLASS1
   When = "B"
      &SORT = "A";
      &DED_CLASS2 = "B";
      Break;
   When = "A"
      &SORT = "B";
      &DED_CLASS2 = "A";
      Break;
   When = "N"
      &DED_CLASS2 = "N";
      &SORT = "C";
      Break;
   When = "P"
      If &CheckCountry = "USA" Then
         &DED_CLASS2 = "P";
         &SORT = "C";
      Else
         If &CheckCountry = "CAN" Then
            &DED_CLASS2 = "N";
            &SORT = "C";
         End-If;
      End-If;
      Break;
   When = "L"
   When = "T"
      &DED_CLASS2 = "T";
      &SORT = "D";
      Break;
   When-Other
      &DED_CLASS2 = &DED_CLASS1;
      &SORT = "Z";
      Break;
```

```
   End-Evaluate;
   rem form the code by combining plan_type, dedcd and ded_class;
   &CODE = &PLAN_TYPE | &DEDCD | &DED_CLASS2;
End-Function;

/*---------------------------------------------------------------------*/
/*--This function reads the source deductions scroll. This could either be
the current or YTD scroll. The value of &TYPE decides if current or YTD data
is being read. A value of 'C' indicates current while 'Y' indicates YTD data.
For every row read, it checks if this deduction has garnishments. If it does, it calls the function
get_garnishments() to get the
corresponding garnishments. Deductions are grouped by the Deduction-
class. It then calls the function update_array() to update the
target scroll (PY_IC_PI_DEDN) ----------------------------------------*/
/*---------------------------------------------------------------------*/
Function get_deductions();
   Decide_Src_Fields();
   For &SRC_PTR = 1 To ActiveRowCount(@&SRC_RECORD);
      &DED_CLASS1 = FetchValue(@&SRC_DEDCLASS_FLD, &SRC_PTR);
      &GARN_PROCESS = FetchValue(@&SRC_RECORD, &SRC_PTR, DEDUCTION_TBL.SPCL_PROCESS);
      &PLAN_TYPE = FetchValue(@&SRC_PLANTYPE_FLD, &SRC_PTR);
      &DEDCD = FetchValue(@&SRC_CODE_FLD, &SRC_PTR);
      &DESCR = FetchValue(@&SRC_RECORD, &SRC_PTR, DEDUCTION_TBL.DESCR);
      If &TYPE = "Y" Then
         &YTD_AMT = FetchValue(@&SRC_YTDAMT_FLD, &SRC_PTR);
      Else
         If &TYPE = "C" Then
            &CUR_AMT = FetchValue(PAY_DEDUCTION.DED_CUR, &SRC_PTR);
         End-If;
      End-If;
      If &GARN_PROCESS = "G" Then
         get_garnishments();
      Else
         Get_Dedn_Code_And_Sort();
         update_array();
      End-If;
   End-For;
End-Function;


Function get_at_deductions();
   Decide_Src_Fields();
   For &SRC_PTR = 1 To ActiveRowCount(@&SRC_RECORD);
      &DED_CLASS1 = FetchValue(@&SRC_DEDCLASS_FLD, &SRC_PTR);
      If &DED_CLASS1 = "A" Then
         &GARN_PROCESS = FetchValue(@&SRC_RECORD, &SRC_PTR, DEDUCTION_TBL.SPCL_PROCESS);
         &PLAN_TYPE = FetchValue(@&SRC_PLANTYPE_FLD, &SRC_PTR);
         &DEDCD = FetchValue(@&SRC_CODE_FLD, &SRC_PTR);
         &DESCR = FetchValue(@&SRC_RECORD, &SRC_PTR, DEDUCTION_TBL.DESCRSHORT);
         If &TYPE = "Y" Then
            &YTD_AMT = FetchValue(@&SRC_YTDAMT_FLD, &SRC_PTR);
         Else
            If &TYPE = "C" Then
               &CUR_AMT = FetchValue(PAY_DEDUCTION.DED_CUR, &SRC_PTR);
            End-If;
         End-If;
         If &GARN_PROCESS = "G" Then
            get_garnishments();
         Else

            Get_Dedn_Code_And_Sort();
            update_array();
         End-If;
      End-If;
   End-For;
End-Function;


Function get_bt_deductions();
   Decide_Src_Fields();
   For &SRC_PTR = 1 To ActiveRowCount(@&SRC_RECORD);
      &DED_CLASS1 = FetchValue(@&SRC_DEDCLASS_FLD, &SRC_PTR);
```

```
      If &DED_CLASS1 = "B" Then
         &GARN_PROCESS = FetchValue(@&SRC_RECORD, &SRC_PTR, DEDUCTION_TBL.SPCL_PROCESS);
         &PLAN_TYPE = FetchValue(@&SRC_PLANTYPE_FLD, &SRC_PTR);
         &DEDCD = FetchValue(@&SRC_CODE_FLD, &SRC_PTR);
         &DESCR = FetchValue(@&SRC_RECORD, &SRC_PTR, DEDUCTION_TBL.DESCRSHORT);
         If &TYPE = "Y" Then
            &YTD_AMT = FetchValue(@&SRC_YTDAMT_FLD, &SRC_PTR);
         Else
            If &TYPE = "C" Then
               &CUR_AMT = FetchValue(PAY_DEDUCTION.DED_CUR, &SRC_PTR);
            End-If;
         End-If;
         If &GARN_PROCESS = "G" Then
            get_garnishments();
         Else
            Get_Dedn_Code_And_Sort();
            update_array();
         End-If;
      End-If;
   End-For;
End-Function;

Function get_er_deductions();
   Decide_Src_Fields();
   For &SRC_PTR = 1 To ActiveRowCount(@&SRC_RECORD);
      &DED_CLASS1 = FetchValue(@&SRC_DEDCLASS_FLD, &SRC_PTR);
      If &DED_CLASS1 = "L" Or
            &DED_CLASS1 = "N" Or
            &DED_CLASS1 = "P" Or
            &DED_CLASS1 = "T" Then
         &GARN_PROCESS = FetchValue(@&SRC_RECORD, &SRC_PTR, DEDUCTION_TBL.SPCL_PROCESS);
         &PLAN_TYPE = FetchValue(@&SRC_PLANTYPE_FLD, &SRC_PTR);
         &DEDCD = FetchValue(@&SRC_CODE_FLD, &SRC_PTR);
         &DESCR = FetchValue(@&SRC_RECORD, &SRC_PTR, DEDUCTION_TBL.DESCRSHORT);
         If &DED_CLASS1 = "L" Or
               &DED_CLASS1 = "T" Then
            &DESCR = &DESCR | "*"
         End-If;
         If &TYPE = "Y" Then
            &YTD_AMT = FetchValue(@&SRC_YTDAMT_FLD, &SRC_PTR);
         Else
            If &TYPE = "C" Then
               &CUR_AMT = FetchValue(PAY_DEDUCTION.DED_CUR, &SRC_PTR);
            End-If;
         End-If;
         If &GARN_PROCESS = "G" Then
            get_garnishments();
         Else

            Get_Dedn_Code_And_Sort();
            update_array();
         End-If;
      End-If;
   End-For;
End-Function;

/*---------------------------------------------------------------------*/
/*---Function - check_for_no_data()-------------------------------------*/
/*--This function works on the target scoll. The target scroll could ---*/
/*--either be PY_IC_PI_ERN (Earnings) or PY_IC_PI_DEDN(Deductions) -----*/
/*--or PY_IC_PI_TAX (Taxes).
/*--If the scroll does not have data, this function displaye a message-*/
/*--to that effect. In case, the scroll has more than one row, it sorts-*/
/*--the scroll appropriately.
/*---------------------------------------------------------------------*/
Function check_for_no_data();
   &NO_OF_ROWS = ActiveRowCount(@&TGT_RECORD);
   If &NO_OF_ROWS = 1 Then
      /* code removed - these grids are not displayed anywhere in the transaction */
   Else
      Evaluate &TGT_RECORD_SCROLL
      When = "PY IC PI ERN"
```

```
         If &Prior_Earns_Found = "N" Then
            SortScroll(1, @&TGT_RECORD, @&TGT_CURAMT_FLD, "D");
         End-If;
         Break;
      When = "PY_IC_PI_DEDN"
         SortScroll(1, @&TGT_RECORD, @&TGT_SORT_FLD, "A");
         Break;
      When = "PY_IC_PI_TAX"
         SortScroll(1, @&TGT_RECORD, @&TGT_SORT_FLD, "A");
         Break;
      When-Other
         Break;
      End-Evaluate;
   End-If;
End-Function;

/*---------------------------------------------------------------------*/
/*--This function gets data from the Payroll Deductions and Garnishment
tables. Both current and YTD data is read. It then calls functions that use
this data to populate the target scroll - PY_IC_PI_DEDN. ----------*/
/*---------------------------------------------------------------------*/
Function populate_deductions();
   /*--Initialize the target record - PY_IC_PI_DEDN----*/
   &TGT_RECORD_SCROLL = "PY_IC_PI_DEDN";
   Decide_Tgt_Fields();
   ScrollFlush(@&TGT_RECORD);

   /* set up garnishment exist clause. */

   &GARN_EXIST = "  AND EXISTS (SELECT " | &QUOTE | "X" | &QUOTE | "FROM PS_PAY_DEDUCTION D WHERE
D.COMPANY = PS_PAY_GARNISH.COMPANY AND D.PAYGROUP = PS_PAY_GARNISH.PAYGROUP AND D.PAY_END_DT =
PS_PAY_GARNISH.PAY_END_DT AND D.OFF_CYCLE = PS_PAY_GARNISH.OFF_CYCLE AND D.PAGE_NUM =
PS_PAY_GARNISH.PAGE_NUM AND D.LINE_NUM = PS_PAY_GARNISH.LINE_NUM and D.SEPCHK = PS_PAY_GARNISH.SEPCHK
AND D.BENEFIT_PLAN = PS_PAY_GARNISH.BENEFIT_PLAN AND D.PLAN_TYPE = PS_PAY_GARNISH.PLAN_TYPE AND D.DEDCD
= PS_PAY_GARNISH.DEDCD AND D.DED_CLASS = PS_PAY_GARNISH.DED_CLASS) ";


   /*---------------------------------------------------------------------*/
   /*---Now read in data for current deductions                  ------*/
   &NO_OF_CUR_ROWS = ActiveRowCount(Record.PAY_DEDUCTION);
   For &I = 1 To &NO_OF_CUR_ROWS
      ScrollFlush(Record.PAY_DEDUCTION, &I, Record.PAY_GARNISH);
   End-For;
   ScrollFlush(Record.PAY_DEDUCTION);
   &ORDER_CLAUSE = "order by PLAN_TYPE, BENEFIT_PLAN, DEDCD, DED_CLASS";
   &WHERE_CLAUSE = &GEN_CUR_WHERE | &SPACE | &ORDER_CLAUSE;
   ScrollSelect(1, Record.PAY_DEDUCTION, Record.PAY_DEDUCTION, &WHERE_CLAUSE, True);
   &ORDER_CLAUSE = "ORDER BY GARNID ASC";
   &WHERE_CLAUSE = &GEN_CUR_WHERE | &SPACE | &GARN_EXIST | &SPACE | &ORDER_CLAUSE;
   ScrollSelect(2, Record.PAY_DEDUCTION, Record.PAY_GARNISH, Record.PAY_GARNISH, &WHERE_CLAUSE, True);
   /*---------------------------------------------------------------------*/
   /*-----Now read in data for YTD deductions.--------------------------*/
   /*--Note: For YTD, separate tables have to be read for USA and Canada--*/

   &GEN_YTD_WHERE2 = "  B1 WHERE B1.EMPLID =  " | &QUOTE | PY_IC_PI_SUM_VW.EMPLID | &QUOTE | " AND
B1.COMPANY = " | &QUOTE | PY_IC_PI_SUM_VW.COMPANY_ALT | &QUOTE | " AND B1.BALANCE_ID = " | &QUOTE |
INSTALLATION.BAL_ID_FOR_CAL_YR | &QUOTE | " AND B1.BALANCE_YEAR = " | PY_IC_PI_WRK.BALANCE_YEAR | " AND
( B1.BALANCE_PERIOD  = " | &SPACE;
   If &CheckCountry = "USA" Then
      &WHERE_CLAUSE = &GEN_YTD_WHERE2 | "( SELECT MAX(DB2.BALANCE_PERIOD) FROM PS_DEDUCTION_BAL DB2
WHERE DB2.EMPLID = B1.EMPLID AND DB2.COMPANY = B1.COMPANY AND DB2.BALANCE_ID = B1.BALANCE_ID AND
DB2.BALANCE_YEAR = B1.BALANCE_YEAR AND DB2.PLAN_TYPE = B1.PLAN_TYPE AND DB2.BENEFIT_PLAN =
B1.BENEFIT_PLAN AND DB2.DEDCD = B1.DEDCD AND DB2.DED_CLASS = B1.DED_CLASS AND DB2.BENEFIT_RCD_NBR =
B1.BENEFIT_RCD_NBR AND DB2.BALANCE_PERIOD <= " | PY_IC_PI_WRK.BALANCE_PERIOD | " ) OR B1.BALANCE_PERIOD
IN (SELECT DISTINCT(G.BALANCE_PERIOD) FROM PS_GARN_BALANCE G WHERE G.EMPLID = B1.EMPLID AND G.COMPANY =
B1.COMPANY AND G.BALANCE_ID = B1.BALANCE_ID AND G.BALANCE_YEAR  = B1.BALANCE_YEAR AND G.PLAN_TYPE =
B1.PLAN_TYPE AND G.BENEFIT_PLAN = B1.BENEFIT_PLAN AND G.DEDCD = B1.DEDCD AND G.DED_CLASS = B1.DED_CLASS
AND G.BALANCE_PERIOD = (SELECT MAX(G3.BALANCE_PERIOD) FROM PS_GARN_BALANCE G3 WHERE G3.EMPLID = G.EMPLID
AND G3.COMPANY = G.COMPANY AND G3.BALANCE_ID = G.BALANCE_ID AND G3.BALANCE_YEAR = G.BALANCE_YEAR AND
G3.BALANCE_QTR = G.BALANCE_QTR AND G3.PLAN_TYPE = G.PLAN_TYPE AND G3.BENEFIT_PLAN = G.BENEFIT_PLAN AND
G3.DEDCD = G.DEDCD AND G3.DED_CLASS = G.DED_CLASS AND G3.GARNID = G.GARNID) ) ) AND B1.DED_YTD <> 0";
      &ORDER CLAUSE = "order by PLAN TYPE, BENEFIT PLAN, DEDCD, DED CLASS";
```

```
      &WHERE_CLAUSE = &WHERE_CLAUSE | &SPACE | &ORDER_CLAUSE;
      &NO_OF_BAL_ROWS = ActiveRowCount(Record.DEDUCTION_BAL);
      For &I = 1 To &NO_OF_BAL_ROWS
         ScrollFlush(Record.DEDUCTION_BAL, &I, Record.GARN_BALANCE);
      End-For;
      ScrollFlush(Record.DEDUCTION_BAL);
      ScrollSelect(1, Record.DEDUCTION_BAL, Record.DEDUCTION_BAL, &WHERE_CLAUSE, True);
      &WHERE_CLAUSE = &GEN_YTD_WHERE | "( SELECT MAX(GB2.BALANCE_PERIOD) FROM PS_GARN_BALANCE GB2 WHERE
GB2.EMPLID = B1.EMPLID AND GB2.COMPANY = B1.COMPANY AND GB2.BALANCE_ID = B1.BALANCE_ID AND
GB2.BALANCE_YEAR = B1.BALANCE_YEAR AND GB2.PLAN_TYPE = B1.PLAN_TYPE AND GB2.BENEFIT_PLAN =
B1.BENEFIT_PLAN AND GB2.DEDCD = B1.DEDCD AND GB2.DED_CLASS = B1.DED_CLASS AND GB2.GARNID = B1.GARNID AND
GB2.BALANCE_PERIOD <= " | PY_IC_PI_WRK.BALANCE_PERIOD | " ) AND B1.DED_YTD <> 0 AND EXISTS (SELECT
DED_YTD FROM PS_DEDUCTION_BAL CDB WHERE CDB.EMPLID = B1.EMPLID AND CDB.COMPANY = B1.COMPANY AND
CDB.BALANCE_ID = B1.BALANCE_ID AND CDB.BALANCE_YEAR = B1.BALANCE_YEAR AND CDB.BALANCE_QTR =
B1.BALANCE_QTR AND CDB.BALANCE_PERIOD = B1.BALANCE_PERIOD AND CDB.PLAN_TYPE = B1.PLAN_TYPE AND
CDB.BENEFIT_PLAN = B1.BENEFIT_PLAN AND CDB.DEDCD = B1.DEDCD AND CDB.DED_CLASS = B1.DED_CLASS AND
CDB.DED_YTD <> 0) ";
      &ORDER_CLAUSE = "ORDER BY GARNID ASC";
      &WHERE_CLAUSE = &WHERE_CLAUSE | &SPACE | &ORDER_CLAUSE;
      ScrollSelect(2, Record.DEDUCTION_BAL, Record.GARN_BALANCE, Record.GARN_BALANCE, &WHERE_CLAUSE,
True);
      &SRC_RECORD_SCROLL = "DEDUCTION_BAL";
      &GARN_RECORD_SCROLL = "GARN_BALANCE";
   Else
      If &CheckCountry = "CAN" Then
         &WHERE_CLAUSE = &GEN_YTD_WHERE2 | "( SELECT MAX(DB2.BALANCE_PERIOD) FROM PS_CAN_DED_BALANCE DB2
WHERE DB2.EMPLID = B1.EMPLID AND DB2.COMPANY = B1.COMPANY AND DB2.BALANCE_ID = B1.BALANCE_ID AND
DB2.BALANCE_YEAR = B1.BALANCE_YEAR AND DB2.WAGE_LOSS_PLAN = B1.WAGE_LOSS_PLAN AND DB2.PROVINCE =
B1.PROVINCE AND DB2.PLAN_TYPE = B1.PLAN_TYPE AND DB2.DEDCD = B1.DEDCD AND DB2.DED_CLASS = B1.DED_CLASS
AND DB2.DED_SLSTX_CLASS = B1.DED_SLSTX_CLASS AND DB2.BENEFIT_RCD_NBR = B1.BENEFIT_RCD_NBR AND
DB2.BALANCE_PERIOD <= " | PY_IC_PI_WRK.BALANCE_PERIOD | " ) OR B1.BALANCE_PERIOD IN (SELECT DISTINCT
(G.BALANCE_PERIOD) FROM PS_GARN_BALANCE G WHERE G.EMPLID = B1.EMPLID AND G.COMPANY = B1.COMPANY AND
G.BALANCE_ID = B1.BALANCE_ID AND G.BALANCE_YEAR  = B1.BALANCE_YEAR AND G.PLAN_TYPE = B1.PLAN_TYPE AND
G.BENEFIT_PLAN = B1.BENEFIT_PLAN AND G.DEDCD = B1.DEDCD AND G.DED_CLASS = B1.DED_CLASS AND
G.BALANCE_PERIOD = (SELECT MAX(G3.BALANCE_PERIOD) FROM PS_GARN_BALANCE G3 WHERE G3.EMPLID = G.EMPLID AND
G3.COMPANY = G.COMPANY AND G3.BALANCE_ID = G.BALANCE_ID AND G3.BALANCE_YEAR = G.BALANCE_YEAR AND
G3.BALANCE_QTR = G.BALANCE_QTR AND G3.PLAN_TYPE = G.PLAN_TYPE AND G3.BENEFIT_PLAN = G.BENEFIT_PLAN AND
G3.DEDCD = G.DEDCD AND G3.DED_CLASS = G.DED_CLASS AND G3.GARNID = G.GARNID) ) ) AND B1.DED_YTD <> 0";
         &ORDER_CLAUSE = "order by PLAN_TYPE, BENEFIT_PLAN, DEDCD, DED_CLASS,DED_SLSTX_CLASS";
         &WHERE_CLAUSE = &WHERE_CLAUSE | &SPACE | &ORDER_CLAUSE;
         ScrollFlush(Record.CAN_DED_BALANCE);
         ScrollSelect(1, Record.CAN_DED_BALANCE, Record.CAN_DED_BALANCE, &WHERE_CLAUSE, True);
         &NO_OF_BAL_ROWS = ActiveRowCount(Record.CAN_DED_BALANCE);
         For &I = 1 To &NO_OF_BAL_ROWS
            ScrollFlush(Record.CAN_DED_BALANCE, &I, Record.PY_IC_PI_GBL_VW);
         End-For;
         &WHERE_CLAUSE = &GEN_YTD_WHERE | "( SELECT MAX(GB2.BALANCE_PERIOD) FROM PS_PY_IC_PI_GBL_VW GB2
WHERE GB2.EMPLID = B1.EMPLID AND GB2.COMPANY = B1.COMPANY AND GB2.BALANCE_ID = B1.BALANCE_ID AND
GB2.BALANCE_YEAR = B1.BALANCE_YEAR AND GB2.PLAN_TYPE = B1.PLAN_TYPE AND GB2.BENEFIT_PLAN =
B1.BENEFIT_PLAN AND GB2.DEDCD = B1.DEDCD AND GB2.DED_CLASS = B1.DED_CLASS AND GB2.GARNID = B1.GARNID AND
GB2.BALANCE_PERIOD <= " | PY_IC_PI_WRK.BALANCE_PERIOD | " ) AND B1.DED_YTD <> 0 AND EXISTS (SELECT
DED_YTD FROM PS_CAN_DED_BALANCE CDB WHERE CDB.EMPLID = B1.EMPLID AND CDB.COMPANY = B1.COMPANY AND
CDB.BALANCE_ID = B1.BALANCE_ID AND CDB.BALANCE_YEAR = B1.BALANCE_YEAR AND CDB.BALANCE_QTR =
B1.BALANCE_QTR AND CDB.BALANCE_PERIOD = B1.BALANCE_PERIOD AND CDB.PLAN_TYPE = B1.PLAN_TYPE AND
CDB.BENEFIT_PLAN = B1.BENEFIT_PLAN AND CDB.DEDCD = B1.DEDCD AND CDB.DED_CLASS = B1.DED_CLASS AND
CDB.DED_YTD <> 0)";
         &ORDER_CLAUSE = "ORDER BY GARNID ASC";
         &WHERE_CLAUSE = &WHERE_CLAUSE | &SPACE | &ORDER_CLAUSE;
         ScrollSelect(2, Record.CAN_DED_BALANCE, Record.PY_IC_PI_GBL_VW, Record.PY_IC_PI_GBL_VW,
&WHERE_CLAUSE, True);
         &SRC_RECORD_SCROLL = "CAN_DED_BALANCE";
         &GARN_RECORD_SCROLL = "PY_IC_PI_GBL_VW";
      End-If;
   End-If;
   Decide_Src_Fields();
   &TYPE = "Y";
   get_deductions();



   /* Populate the AT - After Tax Dedn, BT - Before Tax Dedn, ER - Employer Paid Benefits, ERN -
Eranings, and the TAX -Taxes tables. */
```

```
   /*--Initialize the target record - PY_IC_PI_AT_DED----*/
   &TGT_RECORD_SCROLL = "PY_IC_PI_AT_DED";
   Decide_Tgt_Fields();
   ScrollFlush(@&TGT_RECORD);
   get_at_deductions();

   /*--Initialize the target record - PY_IC_PI_BT_DED----*/
   &TGT_RECORD_SCROLL = "PY_IC_PI_BT_DED";
   Decide_Tgt_Fields();
   ScrollFlush(@&TGT_RECORD);
   get_bt_deductions();

   /*--Initialize the target record - PY_IC_PI_ER_DED----  */
   &TGT_RECORD_SCROLL = "PY_IC_PI_ER_DED";
   Decide_Tgt_Fields();
   ScrollFlush(@&TGT_RECORD);
   get_er_deductions();

   &TGT_RECORD_SCROLL = "PY_IC_PI_DEDN";
   Decide_Tgt_Fields();


   &SRC_RECORD_SCROLL = "PAY_DEDUCTION";
   &GARN_RECORD_SCROLL = "PAY_GARNISH";
   Decide_Src_Fields();
   &TYPE = "C";
   get_deductions();

   /*--Initialize the target record - PY_IC_PI_AT_DED----*/
   &TGT_RECORD_SCROLL = "PY_IC_PI_AT_DED";
   Decide_Tgt_Fields();
   get_at_deductions();

   /*--Initialize the target record - PY_IC_PI_BT_DED---- */
   &TGT_RECORD_SCROLL = "PY_IC_PI_BT_DED";
   Decide_Tgt_Fields();
   get_bt_deductions();

   /*--Initialize the target record - PY_IC_PI_ER_DED---- */
   &TGT_RECORD_SCROLL = "PY_IC_PI_ER_DED";
   Decide_Tgt_Fields();
   get_er_deductions();
   check_for_no_data();
End-Function;




Function Determine_max_ded_rows();

   /* check for max rows when this is a prior cheque and YTD should not be shown on grid   */
   If &CURRENT_CHK = "N" Then
      If &Row_Num_bt >= &Row_Num_at And
            &Row_Num_bt >= &Row_Num_er Then
         &Max_rows = &Row_Num_bt;
      End-If;
      If &Row_Num_at >= &Row_Num_bt And
            &Row_Num_at >= &Row_Num_er Then
         &Max_rows = &Row_Num_at;
      End-If;

      If &Row_Num_er >= &Row_Num_at And
            &Row_Num_er >= &Row_Num_bt Then
         &Max_rows = &Row_Num_er;
      End-If;


   End-If;
   /* check for max rows when this is a current cheque and YTD should be shown on grid   */
```

```
   If &CURRENT_CHK = "Y" Then
      If ActiveRowCount(Record.PY_IC_PI_BT_DED) >= ActiveRowCount(Record.PY_IC_PI_AT_DED) And
            ActiveRowCount(Record.PY_IC_PI_BT_DED) >= ActiveRowCount(Record.PY_IC_PI_ER_DED) Then
         &Max_rows = ActiveRowCount(Record.PY_IC_PI_BT_DED);
      End-If;


      If ActiveRowCount(Record.PY_IC_PI_AT_DED) >= ActiveRowCount(Record.PY_IC_PI_BT_DED) And
            ActiveRowCount(Record.PY_IC_PI_AT_DED) >= ActiveRowCount(Record.PY_IC_PI_ER_DED) Then
         &Max_rows = ActiveRowCount(Record.PY_IC_PI_AT_DED);
      End-If;

      If ActiveRowCount(Record.PY_IC_PI_ER_DED) >= ActiveRowCount(Record.PY_IC_PI_BT_DED) And
            ActiveRowCount(Record.PY_IC_PI_ER_DED) >= ActiveRowCount(Record.PY_IC_PI_AT_DED) Then
         &Max_rows = ActiveRowCount(Record.PY_IC_PI_ER_DED);
      End-If;
   End-If;
End-Function;


Function Determine_max_ERN_TAX_rows();

   /* check for max rows when this is a prior cheque and YTD should not be shown on grid   */
   If &CURRENT_CHK = "N" Then
      If &Row_Num_ern >= &Row_Num_tax Then
         &Max_rows = &Row_Num_ern;
      Else
         &Max_rows = &Row_Num_tax;
      End-If;
   End-If;


   /* check for max rows when this is a current cheque and YTD should be shown on grid   */
   If &CURRENT_CHK = "Y" Then
      If ActiveRowCount(Record.PY_IC_PI_ERN) >= ActiveRowCount(Record.PY_IC_PI_TAX) Then
         &Max_rows = ActiveRowCount(Record.PY_IC_PI_ERN);
      Else
         &Max_rows = ActiveRowCount(Record.PY_IC_PI_TAX);
      End-If;
   End-If;
End-Function;




Function Pad_to_max_rows(&Row_Num, &Max_Rows, &Total_Current_Amt, &Total_Ytd_Amt, &Total_Current_Hrs);
   &TGT_SEQNO_FLD = &TGT_RECORD_SCROLL | "." | "SEQNO";
   If (&Row_Num < &Max_Rows) Then
      For &J = (&Row_Num + 1) To &Max_Rows
         InsertRow(@&TGT_RECORD, &J);
         UpdateValue(@&TGT_SEQNO_FLD, &J, &J);
      End-For;

   End-If;

   /* insert one blank row before the total Row in the grid */
   &Total_Rows = &Max_Rows + 1;
   InsertRow(@&TGT_RECORD, &Total_Rows);
   UpdateValue(@&TGT_SEQNO_FLD, &Total_Rows, &Total_Rows);

   /* If 'Employer Paid Benefits' column, add a line that shows '* Taxable' */
   If &TGT_RECORD_SCROLL = "PY_IC_PI_ER_MX" Then
      &Total_Rows = &Max_Rows + 2;
      InsertRow(@&TGT_RECORD, &Total_Rows);
      UpdateValue(@&TGT_SEQNO_FLD, &Total_Rows, &Total_Rows);

      &TGT_DESCR_FLD = &TGT_RECORD_SCROLL | "." | "PAY_WEB_DESCR";
      &Taxable_Literal = MsgGetText(2001, 557, "Taxable");
      UpdateValue(@&TGT_DESCR_FLD, &Total_Rows, &Taxable_Literal);
   Else
      &Total_Rows = &Max_Rows + 2;
      InsertRow(@&TGT RECORD, &Total Rows);
```

```
      UpdateValue(@&TGT_SEQNO_FLD, &Total_Rows, &Total_Rows);
   End-If;

   &Total_Rows = &Max_Rows + 3;
   InsertRow(@&TGT_RECORD, &Total_Rows);
   UpdateValue(@&TGT_SEQNO_FLD, &Total_Rows, &Total_Rows);

   &TGT_DESCR_FLD = &TGT_RECORD_SCROLL | "." | "PAY_WEB_DESCR";
   &Total_Literal = MsgGetText(2001, 109, "Total");
   UpdateValue(@&TGT_DESCR_FLD, &Total_Rows, &Total_Literal);

   If &Total_Current_Hrs <> 0 Then
      &TGT_CURHRS_FLD = &TGT_RECORD_SCROLL | "." | "PAY_WEB_HRS";
      UpdateValue(@&TGT_CURHRS_FLD, &Total_Rows, &Total_Current_Hrs);
   End-If;

   &TGT_CUR_AMT_FLD = &TGT_RECORD_SCROLL | "." | "PAY_WEB_AMT";
   UpdateValue(@&TGT_CUR_AMT_FLD, &Total_Rows, &Total_Current_Amt);

   &TGT_YTD_AMT_FLD = &TGT_RECORD_SCROLL | "." | "PAY_WEB_YTD_AMT";
   UpdateValue(@&TGT_YTD_AMT_FLD, &Total_Rows, &Total_Ytd_Amt);

End-Function;


Function populate_ded_btmx();

   &TGT_RECORD_SCROLL = "PY_IC_PI_BT_MX";
   REM ScrollFlush(@&TGT_RECORD);
   Decide_Tgt_Fields();


   &Actual_Rows = ActiveRowCount(Record.PY_IC_PI_BT_DED);

   /* &Row_Num is used to track the number of rows that are inserted where the current amount is not
zero. */

   &Row_Num = 0;
   &Total_Cur_Amt_bt = 0;
   &Total_Ytd_Amt_bt = 0;



   For &J = 1 To &Actual_Rows
      &CODE = FetchValue(PY_IC_PI_BT_DED.PAY_WEB_CODE, &J);
      &DESCR = FetchValue(PY_IC_PI_BT_DED.PAY_WEB_DESCR, &J);
      &AMT = FetchValue(PY_IC_PI_BT_DED.PAY_WEB_AMT, &J);
      &SORT = FetchValue(PY_IC_PI_BT_DED.PAY_WEB_SORT, &J);
      &YTD_AMT = FetchValue(PY_IC_PI_BT_DED.PAY_WEB_YTD_AMT, &J);

      If (&CURRENT_CHK = "N" And
               &AMT <> 0) Or
            (&CURRENT_CHK = "Y" And
               Not None(&AMT, &YTD_AMT)) Then

         &Row_Num = &Row_Num + 1;
         InsertRow(@&TGT_RECORD, &Row_Num);
         UpdateValue(PY_IC_PI_BT_MX.SEQNO, &Row_Num, &Row_Num);
         UpdateValue(PY_IC_PI_BT_MX.PAY_WEB_CODE, &Row_Num, &CODE);
         UpdateValue(PY_IC_PI_BT_MX.PAY_WEB_DESCR, &Row_Num, &DESCR);
         UpdateValue(PY_IC_PI_BT_MX.PAY_WEB_AMT, &Row_Num, &AMT);
         UpdateValue(PY_IC_PI_BT_MX.PAY_WEB_SORT, &Row_Num, &SORT);
         UpdateValue(PY_IC_PI_BT_MX.PAY_WEB_YTD_AMT, &Row_Num, &YTD_AMT);
         &Total_Cur_Amt_bt = &Total_Cur_Amt_bt + &AMT;
         &Total_Ytd_Amt_bt = &Total_Ytd_Amt_bt + &YTD_AMT;
      End-If;
      &Row_Num_bt = &Row_Num;
   End-For;



End-Function;
```

```
Function populate_ded_atmx();

   &TGT_RECORD_SCROLL = "PY_IC_PI_AT_MX";
   REM ScrollFlush(@&TGT_RECORD);
   Decide_Tgt_Fields();

   /* For &J = 1 To ActiveRowCount(Record.PY_IC_PI_AT_DED) */
   &Actual_Rows = ActiveRowCount(Record.PY_IC_PI_AT_DED);

   /* &Row_Num is used to track the number of rows that are inserted where the current amount is not
zero. */

   &Row_Num = 0;
   &Total_Cur_Amt_at = 0;
   &Total_Ytd_Amt_at = 0;


   For &J = 1 To &Actual_Rows
      &CODE = FetchValue(PY_IC_PI_AT_DED.PAY_WEB_CODE, &J);
      &DESCR = FetchValue(PY_IC_PI_AT_DED.PAY_WEB_DESCR, &J);
      &AMT = FetchValue(PY_IC_PI_AT_DED.PAY_WEB_AMT, &J);
      &SORT = FetchValue(PY_IC_PI_AT_DED.PAY_WEB_SORT, &J);
      &YTD_AMT = FetchValue(PY_IC_PI_AT_DED.PAY_WEB_YTD_AMT, &J);

      If (&CURRENT_CHK = "N" And
                &AMT <> 0) Or
             (&CURRENT_CHK = "Y" And
                Not None(&AMT, &YTD_AMT)) Then
         &Row_Num = &Row_Num + 1;
         InsertRow(@&TGT_RECORD, &Row_Num);
         UpdateValue(PY_IC_PI_AT_MX.SEQNO, &Row_Num, &Row_Num);
         UpdateValue(PY_IC_PI_AT_MX.PAY_WEB_CODE, &Row_Num, &CODE);
         UpdateValue(PY_IC_PI_AT_MX.PAY_WEB_DESCR, &Row_Num, &DESCR);
         UpdateValue(PY_IC_PI_AT_MX.PAY_WEB_AMT, &Row_Num, &AMT);
         UpdateValue(PY_IC_PI_AT_MX.PAY_WEB_SORT, &Row_Num, &SORT);
         UpdateValue(PY_IC_PI_AT_MX.PAY_WEB_YTD_AMT, &Row_Num, &YTD_AMT);
         &Total_Cur_Amt_at = &Total_Cur_Amt_at + &AMT;
         &Total_Ytd_Amt_at = &Total_Ytd_Amt_at + &YTD_AMT;
      End-If;
      &Row_Num_at = &Row_Num;

   End-For;




End-Function;


Function populate_ded_ermx();

   &TGT_RECORD_SCROLL = "PY_IC_PI_ER_MX";
   REM ScrollFlush(@&TGT_RECORD);
   Decide_Tgt_Fields();


   &Actual_Rows = ActiveRowCount(Record.PY_IC_PI_ER_DED);

   /* &Row_Num is used to track the number of rows that are inserted where the current amount is not
zero. */

   &Row_Num = 0;
   &Total_Cur_Amt_er = 0;
   &Total_Ytd_Amt_er = 0;

   For &J = 1 To &Actual_Rows
      &CODE = FetchValue(PY_IC_PI_ER_DED.PAY_WEB_CODE, &J);
      &DESCR = FetchValue(PY_IC_PI_ER_DED.PAY_WEB_DESCR, &J);
      &AMT = FetchValue(PY_IC_PI_ER_DED.PAY_WEB_AMT, &J);
      &SORT = FetchValue(PY_IC_PI_ER_DED.PAY_WEB_SORT, &J);
      &YTD_AMT = FetchValue(PY_IC_PI_ER_DED.PAY_WEB_YTD_AMT, &J);
```

```
      If (&CURRENT_CHK = "N" And
            &AMT <> 0) Or
          (&CURRENT_CHK = "Y" And
             Not None(&AMT, &YTD_AMT)) Then

         &Row_Num = &Row_Num + 1;

         InsertRow(@&TGT_RECORD, &Row_Num);
         UpdateValue(PY_IC_PI_ER_MX.SEQNO, &Row_Num, &Row_Num);
         UpdateValue(PY_IC_PI_ER_MX.PAY_WEB_CODE, &Row_Num, &CODE);
         UpdateValue(PY_IC_PI_ER_MX.PAY_WEB_DESCR, &Row_Num, &DESCR);
         UpdateValue(PY_IC_PI_ER_MX.PAY_WEB_AMT, &Row_Num, &AMT);
         UpdateValue(PY_IC_PI_ER_MX.PAY_WEB_SORT, &Row_Num, &SORT);
         UpdateValue(PY_IC_PI_ER_MX.PAY_WEB_YTD_AMT, &Row_Num, &YTD_AMT);
         &Total_Cur_Amt_er = &Total_Cur_Amt_er + &AMT;
         &Total_Ytd_Amt_er = &Total_Ytd_Amt_er + &YTD_AMT;
      End-If;
      &Row_Num_er = &Row_Num;

   End-For;



End-Function;



/* LEAVE BALANCE LOGIC FOR MULTIPLE JOB EMPLOYEE RECEIVING MULTIPLE CHECKS */


Function populate_ded_lveMJmx();


   &RSLEAVE_VW = CreateRowset(Record.PY_IC_PI_LVE_VW);

   &RSLEAVE_VW.Flush();


   &TGT_RECORD_SCROLL = "PY_IC_PI_LVE_M";
   &TGT_RECORD = "RECORD" | "." | &TGT_RECORD_SCROLL;
   ScrollFlush(@&TGT_RECORD);



   &EMPLID = PY_IC_PI_SUM_VW.EMPLID;
   &EMPL_RCD = &EMPL_RCD_NBR;
   &BENEFIT_NBR = &BENEFIT_RCD_NBR;
   &PAY_END_DT = PY_IC_PI_SUM_VW.PAY_END_DT_ALT;
   &CHECK_DT = PY_IC_PI_SUM_VW.CHECK_DT;
   &FIRST_TIME = "Y";
   &TOTAL_LVE_ROWS = 0;



   &RSLEAVE_VW.Fill("WHERE FILL.PLAN_TYPE LIKE '5%'  AND FILL.PLAN_TYPE <> '5A'  AND FILL.EMPLID = :1
AND FILL.BENEFIT_PLAN <> ' ' AND FILL.EFFDT = (SELECT MAX(A1.EFFDT) FROM PS_LEAVE_PLAN A1 WHERE
FILL.EMPLID = A1.EMPLID AND FILL.BENEFIT_NBR = A1.BENEFIT_NBR AND FILL.EMPL_RCD = A1.EMPL_RCD AND
A1.PLAN_TYPE = FILL.PLAN_TYPE AND A1.PLAN_TYPE <> '5A' AND A1.EFFDT <= %DATEIN(:2)) AND FILL.PLAN_TYPE
IN (SELECT B.PLAN_TYPE FROM PS_LEAVE_PLAN_TBL B WHERE FILL.PLAN_TYPE = B.PLAN_TYPE AND FILL.BENEFIT_PLAN
= B.BENEFIT_PLAN AND B.BALANCE_VISIBLE = 'Y' AND B.EFFDT = (SELECT MAX(C.EFFDT) FROM PS_LEAVE_PLAN_TBL C
WHERE B.PLAN_TYPE = C.PLAN_TYPE AND B.BENEFIT_PLAN = C.BENEFIT_PLAN AND C.EFFDT <= %DATEIN(:3))) ORDER
BY FILL.PLAN_TYPE", &EMPLID, &PAY_END_DT, &PAY_END_DT);



   &COMPANY = PY_IC_PI_SUM_VW.COMPANY;
   &PLAN_TYPE_PREV = " ";
```

```
   If All(&RSLEAVE_VW(1).PY_IC_PI_LVE_VW.EMPLID.Value) = True Then

      &ARCLVE = &RSLEAVE_VW.ActiveRowCount;
      For &I = 1 To &ARCLVE
         &PLAN_TYPE = &RSLEAVE_VW(&I).PY_IC_PI_LVE_VW.PLAN_TYPE.Value;
         &EMPL_RCD = &RSLEAVE_VW(&I).PY_IC_PI_LVE_VW.EMPL_RCD.Value;

         &STARTBAL = 0;
         &EARNED = 0;
         &BOUGHT = 0;
         &TAKEN = 0;
         &SOLD = 0;
         &ADJUST = 0;
         &ENDBAL = 0;

         If &PLAN_TYPE <> &PLAN_TYPE_PREV And
               &FIRST_TIME = "N" Then

            &TOTAL_LVE_ROWS = &TOTAL_LVE_ROWS + 1;

            &DESCR = Get_Translate_Value("S", "PLAN_TYPE", &PLAN_TYPE_PREV, "ENG", &CHECK_DT);
            InsertRow(@&TGT_RECORD, &TOTAL_LVE_ROWS);
            UpdateValue(PY_IC_PI_LVE_M.PAY_WEB_DESCR, &TOTAL_LVE_ROWS, &DESCR);
            UpdateValue(PY_IC_PI_LVE_M.PAY_WEB_YTD_AMT, &TOTAL_LVE_ROWS, &TOTAL_END_BAL);

            &TOTAL_YTD_AMT_LVE = &TOTAL_YTD_AMT_LVE + &TOTAL_END_BAL;
            &TOTAL_END_BAL = 0;

         End-If;

         SQLExec("SELECT
HRS_CARRYOVER,HRS_EARNED_YTD,HRS_BOUGHT_YTD,HRS_BOUGHT_UNPROC,HRS_TAKEN_YTD,HRS_TAKEN_UNPROC,HRS_SOLD_YT
D,HRS_SOLD_UNPROC,HRS_ADJUST_YTD,HRS_ADJUST_UNPROC,%DateOut(ACCRUAL_PROC_DT) FROM PS_LEAVE_ACCRUAL WHERE
EMPLID = :1 AND EMPL_RCD = :2 AND COMPANY = :3 AND PLAN_TYPE = :4 AND ACCRUAL_PROC_DT = (SELECT MAX
(B.ACCRUAL_PROC_DT) FROM PS_LEAVE_ACCRUAL B WHERE EMPLID = :1 AND EMPL_RCD = :2 AND COMPANY = :3 AND
PLAN_TYPE = :4 AND(ACCRUAL_PROC_DT <= %DATEIN(:5) OR ACCRUAL_PROC_DT IS NULL)) ORDER BY ACCRUAL_PROC_DT
DESC", &EMPLID, &EMPL_RCD, &COMPANY, &PLAN_TYPE, &CHECK_DT, &HRS_CARRYOVER, &HRS_EARNED_YTD,
&HRS_BOUGHT_YTD, &HRS_BOUGHT_UNPROC, &HRS_TAKEN_YTD, &HRS_TAKEN_UNPROC, &HRS_SOLD_YTD, &HRS_SOLD_UNPROC,
&HRS_ADJUST_YTD, &HRS_ADJUST_UNPROC, &ACCRUAL_PROC_DT);

         &STARTBAL = &HRS_CARRYOVER;
         &EARNED = &HRS_EARNED_YTD;
         &BOUGHT = &HRS_BOUGHT_YTD + &HRS_BOUGHT_UNPROC;
         &TAKEN = &HRS_TAKEN_YTD + &HRS_TAKEN_UNPROC;
         &SOLD = &HRS_SOLD_YTD + &HRS_SOLD_UNPROC;
         &ADJUST = &HRS_ADJUST_YTD + &HRS_ADJUST_UNPROC;
         &ENDBAL = &STARTBAL + &EARNED + &BOUGHT - &TAKEN - &SOLD + &ADJUST;

         &TOTAL_END_BAL = &TOTAL_END_BAL + &ENDBAL;

         &PLAN_TYPE_PREV = &PLAN_TYPE;
         &FIRST_TIME = "N";

         If &TOTAL_END_BAL <> 0 Then
            &DATA_EXIST = "Y";
         End-If;




      End-For;

   Else
      &DATA_EXIST = "N";

   End-If;


   /****  LAST RECORD FROM LEAVE ACCURAL  ****/
```

```
   If &DATA_EXIST = "Y" Then
      &TOTAL_LVE_ROWS = &TOTAL_LVE_ROWS + 1;

      &DESCR = Get_Translate_Value("S", "PLAN_TYPE", &PLAN_TYPE_PREV, "ENG", &CHECK_DT);
      InsertRow(@&TGT_RECORD, &TOTAL_LVE_ROWS);
      UpdateValue(PY_IC_PI_LVE_M.PAY_WEB_DESCR, &TOTAL_LVE_ROWS, &DESCR);
      UpdateValue(PY_IC_PI_LVE_M.PAY_WEB_YTD_AMT, &TOTAL_LVE_ROWS, &TOTAL_END_BAL);

      &TOTAL_YTD_AMT_LVE = &TOTAL_YTD_AMT_LVE + &TOTAL_END_BAL;

   End-If;

   /*** INSERT A BLANK LINE  ****/

   If &DATA_EXIST = "Y" Then
      &TOTAL_LVE_ROWS = &TOTAL_LVE_ROWS + 1;
      InsertRow(@&TGT_RECORD, &TOTAL_LVE_ROWS);
      UpdateValue(PY_IC_PI_LVE_M.PAY_WEB_DESCR, &TOTAL_LVE_ROWS, " ");

   End-If;


   /*** INSERT THE TOTAL LINE  ****/

   If &DATA_EXIST = "Y" Then
      &TOTAL_LVE_ROWS = &TOTAL_LVE_ROWS + 1;
      InsertRow(@&TGT_RECORD, &TOTAL_LVE_ROWS);
      &Total_YTD_Literal = MsgGetText(2001, 824, "Total YTD Amount:");
      UpdateValue(PY_IC_PI_LVE_M.PAY_WEB_DESCR, &TOTAL_LVE_ROWS, &Total_YTD_Literal);
      UpdateValue(PY_IC_PI_LVE_M.PAY_WEB_YTD_AMT, &TOTAL_LVE_ROWS, &TOTAL_YTD_AMT_LVE);

   End-If;



End-Function;



/* This function populates the employee PTO leave balances based on the employee receiving a single
check */

Function populate_ded_lveymx();


   /*** SINGLE CHECK  YES  ****/

   &RSLEAVE_VW = CreateRowset(Record.PY_IC_PI_LVE_VW);
   &RSLEAVE_VW.Flush();


   &TGT_RECORD_SCROLL = "PY_IC_PI_LVE_M";
   &TGT_RECORD = "RECORD" | "." | &TGT_RECORD_SCROLL;
   ScrollFlush(@&TGT_RECORD);

   &EMPLID = PY_IC_PI_SUM_VW.EMPLID;
   &PAY_END_DT = PY_IC_PI_SUM_VW.PAY_END_DT_ALT;
   &CHECK_DT = PY_IC_PI_SUM_VW.CHECK_DT;
   &FIRST_TIME = "Y";
   &TOTAL_LVE_ROWS = 0;

   &RSLEAVE_VW.Fill("WHERE FILL.PLAN_TYPE LIKE '5%'  AND FILL.PLAN_TYPE <> '5A'  AND FILL.EMPLID = :1
AND FILL.BENEFIT_PLAN <> ' ' AND FILL.EFFDT = (SELECT MAX(A1.EFFDT) FROM PS_LEAVE_PLAN A1 WHERE
FILL.EMPLID = A1.EMPLID AND A1.EMPL_RCD = FILL.EMPL_RCD AND A1.PLAN_TYPE = FILL.PLAN_TYPE AND
A1.PLAN_TYPE <> '5A' AND A1.EFFDT <= %DATEIN(:2)) AND FILL.PLAN_TYPE IN (SELECT B.PLAN_TYPE FROM
PS_LEAVE_PLAN_TBL B WHERE FILL.PLAN_TYPE = B.PLAN_TYPE AND FILL.BENEFIT_PLAN = B.BENEFIT_PLAN AND
B.BALANCE_VISIBLE = 'Y' AND B.EFFDT = (SELECT MAX(C.EFFDT) FROM PS_LEAVE_PLAN_TBL C WHERE B.PLAN_TYPE =
C.PLAN_TYPE AND B.BENEFIT_PLAN = C.BENEFIT_PLAN AND C.EFFDT <= %DATEIN(:3))) ORDER BY FILL.PLAN_TYPE",
&EMPLID, &PAY_END_DT, &PAY_END_DT);


   &COMPANY = PY IC PI SUM VW.COMPANY;
```

```
&PLAN_TYPE_PREV = " ";

If All(&RSLEAVE_VW(1).PY_IC_PI_LVE_VW.EMPLID.Value) = True Then

   &ARCLVE = &RSLEAVE_VW.ActiveRowCount;
   For &I = 1 To &ARCLVE
      &PLAN_TYPE = &RSLEAVE_VW(&I).PY_IC_PI_LVE_VW.PLAN_TYPE.Value;
      &EMPL_RCD = &RSLEAVE_VW(&I).PY_IC_PI_LVE_VW.EMPL_RCD.Value;


      &STARTBAL = 0;
      &EARNED = 0;
      &BOUGHT = 0;
      &TAKEN = 0;
      &SOLD = 0;
      &ADJUST = 0;
      &ENDBAL = 0;

      If &PLAN_TYPE <> &PLAN_TYPE_PREV And
            &FIRST_TIME = "N" Then

         &TOTAL_LVE_ROWS = &TOTAL_LVE_ROWS + 1;

         &DESCR = Get_Translate_Value("S", "PLAN_TYPE", &PLAN_TYPE_PREV, "ENG", &CHECK_DT);
         InsertRow(@&TGT_RECORD, &TOTAL_LVE_ROWS);
         UpdateValue(PY_IC_PI_LVE_M.PAY_WEB_DESCR, &TOTAL_LVE_ROWS, &DESCR);
         UpdateValue(PY_IC_PI_LVE_M.PAY_WEB_YTD_AMT, &TOTAL_LVE_ROWS, &TOTAL_END_BAL);

         &TOTAL_YTD_AMT_LVE = &TOTAL_YTD_AMT_LVE + &TOTAL_END_BAL;
         &TOTAL_END_BAL = 0;

      End-If;

      SQLExec("SELECT
HRS_CARRYOVER,HRS_EARNED_YTD,HRS_BOUGHT_YTD,HRS_BOUGHT_UNPROC,HRS_TAKEN_YTD,HRS_TAKEN_UNPROC,HRS_SOLD_YT
D,HRS_SOLD_UNPROC,HRS_ADJUST_YTD,HRS_ADJUST_UNPROC,%DateOut(ACCRUAL_PROC_DT)  FROM PS_LEAVE_ACCRUAL
WHERE EMPLID = :1 AND EMPL_RCD = :2 AND COMPANY = :3 AND PLAN_TYPE = :4 AND ACCRUAL_PROC_DT = (SELECT
MAX(B.ACCRUAL_PROC_DT) FROM PS_LEAVE_ACCRUAL B WHERE EMPLID = :1 AND EMPL_RCD = :2 AND COMPANY = :3 AND
PLAN_TYPE = :4 AND(ACCRUAL_PROC_DT <= %DATEIN(:5) OR ACCRUAL_PROC_DT IS NULL)) ORDER BY ACCRUAL_PROC_DT
DESC", &EMPLID, &EMPL_RCD, &COMPANY, &PLAN_TYPE, &CHECK_DT, &HRS_CARRYOVER, &HRS_EARNED_YTD,
&HRS_BOUGHT_YTD, &HRS_BOUGHT_UNPROC, &HRS_TAKEN_YTD, &HRS_TAKEN_UNPROC, &HRS_SOLD_YTD, &HRS_SOLD_UNPROC,
&HRS_ADJUST_YTD, &HRS_ADJUST_UNPROC, &ACCRUAL_PROC_DT);

      &STARTBAL = &HRS_CARRYOVER;
      &EARNED = &HRS_EARNED_YTD;
      &BOUGHT = &HRS_BOUGHT_YTD + &HRS_BOUGHT_UNPROC;
      &TAKEN = &HRS_TAKEN_YTD + &HRS_TAKEN_UNPROC;
      &SOLD = &HRS_SOLD_YTD + &HRS_SOLD_UNPROC;
      &ADJUST = &HRS_ADJUST_YTD + &HRS_ADJUST_UNPROC;
      &ENDBAL = &STARTBAL + &EARNED + &BOUGHT - &TAKEN - &SOLD + &ADJUST;
      &TOTAL_END_BAL = &TOTAL_END_BAL + &ENDBAL;

      &PLAN_TYPE_PREV = &PLAN_TYPE;

      &FIRST_TIME = "N";


      If &TOTAL_END_BAL <> 0 Then
         &DATA_EXIST = "Y";
      End-If;



   End-For;

Else
   &DATA_EXIST = "N";

End-If;


/****  LAST RECORD FROM LEAVE ACCURAL  ****/
```

```
   If &DATA_EXIST = "Y" Then
      &TOTAL_LVE_ROWS = &TOTAL_LVE_ROWS + 1;

      &DESCR = Get_Translate_Value("S", "PLAN_TYPE", &PLAN_TYPE_PREV, "ENG", &CHECK_DT);
      InsertRow(@&TGT_RECORD, &TOTAL_LVE_ROWS);
      UpdateValue(PY_IC_PI_LVE_M.PAY_WEB_DESCR, &TOTAL_LVE_ROWS, &DESCR);
      UpdateValue(PY_IC_PI_LVE_M.PAY_WEB_YTD_AMT, &TOTAL_LVE_ROWS, &TOTAL_END_BAL);

      &TOTAL_YTD_AMT_LVE = &TOTAL_YTD_AMT_LVE + &TOTAL_END_BAL;

   End-If;

   /*** BLANK LINE  ****/

   If &DATA_EXIST = "Y" Then
      &TOTAL_LVE_ROWS = &TOTAL_LVE_ROWS + 1;
      InsertRow(@&TGT_RECORD, &TOTAL_LVE_ROWS);
      UpdateValue(PY_IC_PI_LVE_M.PAY_WEB_DESCR, &TOTAL_LVE_ROWS, " ");

   End-If;


   /*** TOTAL LINE  ****/

   If &DATA_EXIST = "Y" Then
      &TOTAL_LVE_ROWS = &TOTAL_LVE_ROWS + 1;
      InsertRow(@&TGT_RECORD, &TOTAL_LVE_ROWS);
      &Total_YTD_Literal = MsgGetText(2001, 824, "Total YTD Amount:");
      UpdateValue(PY_IC_PI_LVE_M.PAY_WEB_DESCR, &TOTAL_LVE_ROWS, &Total_YTD_Literal);
      UpdateValue(PY_IC_PI_LVE_M.PAY_WEB_YTD_AMT, &TOTAL_LVE_ROWS, &TOTAL_YTD_AMT_LVE);

   End-If;


End-Function;



Function populate_ded_ern_mx();

   &TGT_RECORD_SCROLL = "PY_IC_PI_ERN_MX";
   REM ScrollFlush(@&TGT_RECORD);
   Decide_Tgt_Fields();


   &Actual_Rows = ActiveRowCount(Record.PY_IC_PI_ERN);

   /* &Row_Num is used to track the number of rows that are inserted where the current amount is not
zero. */

   &Row_Num = 0;

   For &J = 1 To &Actual_Rows

      &CODE = FetchValue(PY_IC_PI_ERN.PAY_WEB_CODE, &J);
      If %Language = %Language_Base Then
         SQLExec("SELECT A.DESCRSHORT FROM PS_EARNINGS_TBL A WHERE A.ERNCD = :1 AND A.EFFDT = (SELECT
MAX(A1.EFFDT) FROM PS_EARNINGS_TBL A1 WHERE A1.ERNCD = A.ERNCD AND A1.EFFDT <= %DateIn(:2))", &CODE,
PY_IC_PI_SUM_VW.PAY_END_DT_ALT, &DESCR_ERN);
      Else
         SQLExec("SELECT A.DESCRSHORT FROM PS_EARNINGS_LANG A WHERE A.ERNCD = :1 AND A.LANGUAGE_CD = :2
AND A.EFFDT = (SELECT MAX(A1.EFFDT) FROM PS_EARNINGS_LANG A1 WHERE A1.ERNCD = A.ERNCD AND A1.LANGUAGE_CD
= A.LANGUAGE_CD AND A1.EFFDT <= %DateIn(:3))", &CODE, %Language, PY_IC_PI_SUM_VW.PAY_END_DT_ALT,
&DESCR_ERN);
         If None(&DESCR_ERN) Then
            SQLExec("SELECT A.DESCRSHORT FROM PS_EARNINGS_TBL A WHERE A.ERNCD = :1 AND A.EFFDT = (SELECT
MAX(A1.EFFDT) FROM PS_EARNINGS_TBL A1 WHERE A1.ERNCD = A.ERNCD AND A1.EFFDT <= %DateIn(:2))", &CODE,
PY_IC_PI_SUM_VW.PAY_END_DT_ALT, &DESCR_ERN);
         End-If;
      End-If;
      &RATE = FetchValue(PY_IC_PI_ERN.PAY_WEB_RATE, &J);
```

```
      &AMT = FetchValue(PY_IC_PI_ERN.PAY_WEB_AMT, &J);
      &HRS = FetchValue(PY_IC_PI_ERN.PAY_WEB_HRS, &J);
      &YTD_HRS = FetchValue(PY_IC_PI_ERN.PAY_WEB_YTD_HRS, &J);
      &YTD_AMT = FetchValue(PY_IC_PI_ERN.PAY_WEB_YTD_AMT, &J);
      &COMP_RATECD = FetchValue(PY_IC_PI_ERN.PAY_WEB_COMPRATECD, &J);
      &EARNS_TYPE = FetchValue(PY_IC_PI_ERN.EARNS_TYPE, &J);
      &EARNS_BEGIN_DT = FetchValue(PY_IC_PI_ERN.EARNS_BEGIN_DT, &J);
      &EARNS_END_DT = FetchValue(PY_IC_PI_ERN.EARNS_END_DT, &J);

      If (&CURRENT_CHK = "N" And
               (&AMT <> 0 Or
                  &HRS <> 0)) Or
            (&CURRENT_CHK = "Y" And
               Not None(&AMT, &HRS, &YTD_HRS, &YTD_AMT)) Then
         &Row_Num = &Row_Num + 1;

         InsertRow(@&TGT_RECORD, &Row_Num);
         UpdateValue(PY_IC_PI_ERN_MX.SEQNO, &Row_Num, &Row_Num);
         UpdateValue(PY_IC_PI_ERN_MX.EARNS_TYPE, &Row_Num, &EARNS_TYPE);
         UpdateValue(PY_IC_PI_ERN_MX.EARNS_BEGIN_DT, &Row_Num, &EARNS_BEGIN_DT);
         UpdateValue(PY_IC_PI_ERN_MX.EARNS_END_DT, &Row_Num, &EARNS_END_DT);
         UpdateValue(PY_IC_PI_ERN_MX.PAY_WEB_CODE, &Row_Num, &CODE);
         UpdateValue(PY_IC_PI_ERN_MX.PAY_WEB_DESCR, &Row_Num, &DESCR_ERN);
         UpdateValue(PY_IC_PI_ERN_MX.PAY_WEB_RATE, &Row_Num, &RATE);
         UpdateValue(PY_IC_PI_ERN_MX.PAY_WEB_AMT, &Row_Num, &AMT);
         UpdateValue(PY_IC_PI_ERN_MX.PAY_WEB_HRS, &Row_Num, &HRS);
         UpdateValue(PY_IC_PI_ERN_MX.PAY_WEB_YTD_HRS, &Row_Num, &YTD_HRS);
         UpdateValue(PY_IC_PI_ERN_MX.PAY_WEB_YTD_AMT, &Row_Num, &YTD_AMT);
         UpdateValue(PY_IC_PI_ERN_MX.PAY_WEB_COMPRATECD, &Row_Num, &COMP_RATECD);


      End-If;

   End-For;
   &Row_Num_ern = &Row_Num;

End-Function;

Function populate_ded_tax_mx();

   &TGT_RECORD_SCROLL = "PY_IC_PI_TAX_MX";
   REM ScrollFlush(@&TGT_RECORD);
   Decide_Tgt_Fields();


   &Actual_Rows = ActiveRowCount(Record.PY_IC_PI_TAX);

   /* &Row_Num is used to track the number of rows that are inserted where the current amount is not
zero. */

   &Row_Num = 0;


   For &J = 1 To &Actual_Rows
      &CODE = FetchValue(PY_IC_PI_TAX.PAY_WEB_CODE, &J);
      &DESCR = FetchValue(PY_IC_PI_TAX.PAY_WEB_DESCR, &J);
      &AMT = FetchValue(PY_IC_PI_TAX.PAY_WEB_AMT, &J);
      &SORT = FetchValue(PY_IC_PI_TAX.PAY_WEB_SORT, &J);
      &TAXGRS_CUR = FetchValue(PY_IC_PI_TAX.PAY_WEB_TXGRS_CUR, &J);
      &YTD_AMT = FetchValue(PY_IC_PI_TAX.PAY_WEB_YTD_AMT, &J);
      &TXGRS_YTD = FetchValue(PY_IC_PI_TAX.PAY_WEB_TXGRS_YTD, &J);

      /* We will display zero rows for taxes, to make it consistent with the printed and PDF paychecks
       If (&CURRENT_CHK = "N" And
            &AMT <> 0) Or
            (&CURRENT_CHK = "Y" And
               Not None(&AMT, &YTD_AMT)) Then */
      &Row_Num = &Row_Num + 1;



      InsertRow(@&TGT RECORD, &Row Num);
```

```
      UpdateValue(PY_IC_PI_TAX_MX.SEQNO, &Row_Num, &Row_Num);
      UpdateValue(PY_IC_PI_TAX_MX.PAY_WEB_CODE, &Row_Num, &CODE);
      UpdateValue(PY_IC_PI_TAX_MX.PAY_WEB_DESCR, &Row_Num, &DESCR);
      UpdateValue(PY_IC_PI_TAX_MX.PAY_WEB_AMT, &Row_Num, &AMT);
      UpdateValue(PY_IC_PI_TAX_MX.PAY_WEB_SORT, &Row_Num, &SORT);
      UpdateValue(PY_IC_PI_TAX_MX.PAY_WEB_TXGRS_CUR, &Row_Num, &TAXGRS_CUR);
      UpdateValue(PY_IC_PI_TAX_MX.PAY_WEB_YTD_AMT, &Row_Num, &YTD_AMT);
      UpdateValue(PY_IC_PI_TAX_MX.PAY_WEB_TXGRS_YTD, &Row_Num, &TXGRS_YTD);
      /*End-If;*/
   End-For;


   &Row_Num_tax = &Row_Num;



End-Function;


/*---------------------------------------------------------------------------*/
/*--This function arrange earnigns by group regular hourly, regular salary, */
/*--overtime earning, followed by all other earnings                        */
/*---------------------------------------------------------------------------*/
Function arrange_earnings();

   For &J = 1 To ActiveRowCount(Record.PY_IC_PI_ERN)
      &CODE = FetchValue(PY_IC_PI_ERN.PAY_WEB_CODE, &J);
      If &Prior_Earns_Found = "Y" Then
         Evaluate &CODE
         When = &ERNCD_REG_HRS
            &EARNS_TYPE = "1"; /* Regular Hourly */
            Break;
         When = &ERNCD_REG_EARNS
            &EARNS_TYPE = "2"; /* Regular Salary */
            Break;
         When = &ERNCD_OT_HRS
            &EARNS_TYPE = "3"; /* Overtime */
            Break;
         When-Other
            &EARNS_TYPE = "4";
            Break;
         End-Evaluate;
      Else
         &EARNS_TYPE = " ";
      End-If;
      UpdateValue(PY_IC_PI_ERN.EARNS_TYPE, &J, &EARNS_TYPE);
      &EARNS_BEGIN_DT = FetchValue(PY_IC_PI_ERN.EARNS_BEGIN_DT, &J);
      &EARNS_END_DT = FetchValue(PY_IC_PI_ERN.EARNS_END_DT, &J);
      If None(&EARNS_BEGIN_DT) And
            None(&EARNS_END_DT) Then
         UpdateValue(PY_IC_PI_ERN.EARNS_BEGIN_DT, &J, &Pay_Begin_Dt);
         UpdateValue(PY_IC_PI_ERN.EARNS_END_DT, &J, PY_IC_PI_SUM_VW.PAY_END_DT_ALT);
      End-If;
   End-For;

   SortScroll(1, Record.PY_IC_PI_ERN, PY_IC_PI_ERN.EARNS_TYPE, "A", PY_IC_PI_ERN.PAY_WEB_CODE, "A",
PY_IC_PI_ERN.EARNS_END_DT, "D", PY_IC_PI_ERN.EARNS_BEGIN_DT, "D", PY_IC_PI_ERN.PAY_WEB_RATE, "A");

   For &J = 1 To ActiveRowCount(Record.PY_IC_PI_ERN)
      &EARNS_BEGIN_DT = FetchValue(PY_IC_PI_ERN.EARNS_BEGIN_DT, &J);
      &EARNS_END_DT = FetchValue(PY_IC_PI_ERN.EARNS_END_DT, &J);
      If &EARNS_BEGIN_DT = &Pay_Begin_Dt And
            &EARNS_END_DT = PY_IC_PI_SUM_VW.PAY_END_DT_ALT Then
         UpdateValue(PY_IC_PI_ERN.EARNS_BEGIN_DT, &J, "");
         UpdateValue(PY_IC_PI_ERN.EARNS_END_DT, &J, "");
      End-If;
   End-For;

   &EARNS_BEGIN_DT = "";
   &EARNS_END_DT = "";

End-Function;
```

```
/*-------------------------------------------------------------------------*/
/*--This function gets data from the Payroll Earnings and Other Earnings*/
/*--tables. Both current and YTD earnings are read. It then calls functions that use this data to
populate the target scroll - PY_IC_PI_ERN.---------*/
/*-------------------------------------------------------------------------*/
Function populate_earnings();
   /*--Initialize the target record - PY_IC_PI_ERN----*/
   &TGT_RECORD_SCROLL = "PY_IC_PI_ERN";
   Decide_Tgt_Fields();
   ScrollFlush(@&TGT_RECORD);
   /*-------------------------------------------------------------------------*/
   /*---Now read in data for current earnings and other earnings ------*/
   For &J = 1 To ActiveRowCount(Record.PY_IC_PI_ERN_VW)
      ScrollFlush(Record.PY_IC_PI_ERN_VW, &J, Record.PY_IC_PI_OE_VW);
   End-For;
   ScrollFlush(Record.PY_IC_PI_ERN_VW);
   If &Prior_Earns_Found = "Y" Then
      &ORDER_CLAUSE = "ORDER BY EARNS_END_DT DESC, EARNS_BEGIN_DT DESC";
   Else
      &ORDER_CLAUSE = "ORDER BY ADDL_NBR ASC";
   End-If;
   &WHERE_CLAUSE = &GEN_CUR_WHERE | &SPACE | &ORDER_CLAUSE;
   ScrollSelect(1, Record.PY_IC_PI_ERN_VW, Record.PY_IC_PI_ERN_VW, &WHERE_CLAUSE, True);
   &ORDER_CLAUSE = "ORDER BY ERNCD ASC";
   &WHERE_CLAUSE = &GEN_CUR_WHERE | &SPACE | &ORDER_CLAUSE;
   ScrollSelect(2, Record.PY_IC_PI_ERN_VW, Record.PY_IC_PI_OE_VW, Record.PY_IC_PI_OE_VW, &WHERE_CLAUSE,
True);
   /*-------------------------------------------------------------------------*/
   /*-----Now prepare to read YTD earnings.----------------------------*/
   /*--Note: For YTD, separate tables have to be read for USA and Canada--*/
   If &CheckCountry = "USA" Then
      &WHERE_CLAUSE = &GEN_YTD_WHERE | "(SELECT MAX(EB2.BALANCE_PERIOD) FROM PS_EARNINGS_BAL EB2 WHERE
EB2.EMPLID = B1.EMPLID AND EB2.COMPANY = B1.COMPANY AND EB2.BALANCE_ID = B1.BALANCE_ID AND
EB2.BALANCE_YEAR = B1.BALANCE_YEAR AND EB2.SPCL_BALANCE = B1.SPCL_BALANCE AND EB2.ERNCD = B1.ERNCD AND
EB2.EMPL_RCD = B1.EMPL_RCD AND EB2.BALANCE_PERIOD <= " | PY_IC_PI_WRK.BALANCE_PERIOD | ")";
      &SRC_RECORD_SCROLL = "EARNINGS_BAL";
   Else
      /*-------------------------------------------------------------------------*/
      /*-- Prepare to read Canadian balances--------------------------------------*/
      If &CheckCountry = "CAN" Then
         &WHERE_CLAUSE = &GEN_YTD_WHERE | "(SELECT MAX(EB2.BALANCE_PERIOD) FROM PS_CAN_ERN_BALANCE EB2
WHERE EB2.EMPLID = B1.EMPLID AND EB2.COMPANY = B1.COMPANY AND EB2.BALANCE_ID = B1.BALANCE_ID AND
EB2.BALANCE_YEAR = B1.BALANCE_YEAR AND EB2.WAGE_LOSS_PLAN = B1.WAGE_LOSS_PLAN AND EB2.PROVINCE =
B1.PROVINCE AND EB2.SPCL_BALANCE = B1.SPCL_BALANCE AND EB2.ERNCD = B1.ERNCD AND EB2.EMPL_RCD =
B1.EMPL_RCD AND EB2.BALANCE_PERIOD <= " | PY_IC_PI_WRK.BALANCE_PERIOD | ")";
         &SRC_RECORD_SCROLL = "CAN_ERN_BALANCE";
      End-If;
   End-If;
   /*-------------------------------------------------------------------------*/
   &TYPE = "C";
   get_earnings();
   decide_rate();
   If &Prior_Earns_Found = "Y" Then
      arrange_earnings();
   End-If;
   Decide_Src_Fields();
   /*----Read YTD data-------------------------------------------------------*/
   &WHERE_CLAUSE = &WHERE_CLAUSE | " AND B1.SPCL_BALANCE = " | &QUOTE | "N" | &QUOTE;
   ScrollFlush(@&SRC_RECORD);
   ScrollSelect(1, @&SRC_RECORD, @&SRC_RECORD, &WHERE_CLAUSE, True);
   &TYPE = "Y";
   get_ytd_earnings();
   check_for_no_data();
End-Function;

/*-------------------------------------------------------------------------*/
/*--This function gets data from the Payroll Tax tables. Depending on
the country field, either US or Canadian taxes are fetched. Both Current and
YTD Taxes are read. This data is loaded into the source tax scrolls.It then calls functions that use
this data to populate the target scroll(PY IC PI TAX)--*/
```

```
/*---------------------------------------------------------------------*/
Function populate_taxes();
   /*--Initialize the target scroll - PY_IC_PI_TAX----------------*/
   &TGT_RECORD_SCROLL = "PY_IC_PI_TAX";
   Decide_Tgt_Fields();
   ScrollFlush(@&TGT_RECORD);
   If &CheckCountry = "USA" Then
      &STATE_SEL = " AND STATE NOT IN ('$UAS','$UGU','$UPR','$UVI')";
      &TAX_CLASS_SEL = " ('A','B','C','D','F','G','H','I','L','M','N','O','P','T','V','W')";
      &TAX_CLASS_SEL = " ('A','B','C','D','F','G','H','I','L','M','N','O','P','T','V','W','7')";
      &ORDER_CLAUSE = "ORDER BY STATE ASC, LOCALITY ASC,  TAX_CLASS DESC";
      /*---Read current data ----------------------------------------*/
      &WHERE_CLAUSE = &GEN_CUR_WHERE | &STATE_SEL | &SPACE | "AND TAX_CLASS IN " | &TAX_CLASS_SEL |
&SPACE | &ORDER_CLAUSE;
      ScrollFlush(Record.PAY_TAX);
      ScrollSelect(1, Record.PAY_TAX, Record.PAY_TAX, &WHERE_CLAUSE, True);
      /*      SortScroll(1, RECORD.PAY_TAX, PAY_TAX.STATE, "A", PAY_TAX.LOCALITY, "A",
PAY_TAX.TAX_CLASS, "D");*/
      /*-----Read YTD data  -----------------------------------------*/
      &WHERE_CLAUSE = &GEN_YTD_WHERE | "( SELECT MAX(TB2.BALANCE_PERIOD) FROM PS_TAX_BALANCE TB2 WHERE
TB2.EMPLID = B1.EMPLID AND TB2.COMPANY = B1.COMPANY AND TB2.BALANCE_ID = B1.BALANCE_ID AND
TB2.BALANCE_YEAR = B1.BALANCE_YEAR AND TB2.STATE = B1.STATE AND TB2.LOCALITY = B1.LOCALITY AND
TB2.TAX_CLASS = B1.TAX_CLASS AND TB2.WORK_PSD_CD = B1.WORK_PSD_CD AND TB2.RES_PSD_CD = B1.RES_PSD_CD AND
TB2.BALANCE_PERIOD <= " | PY_IC_PI_WRK.BALANCE_PERIOD | &STATE_SEL | " ) AND TAX_CLASS IN" |
&TAX_CLASS_SEL;
      &WHERE_CLAUSE = &WHERE_CLAUSE | &SPACE | &ORDER_CLAUSE;
      ScrollFlush(Record.TAX_BALANCE);
      ScrollSelect(1, Record.TAX_BALANCE, Record.TAX_BALANCE, &WHERE_CLAUSE, True);
      &SRC_RECORD_SCROLL = "TAX_BALANCE";
      Decide_Src_Fields();
      /*---------------------------------------------------------------*/
      /*--&TYPE will be 'Y' for YTD and 'C' for current data rows
First get the YTD data. After that we get the current data.----*/
      /*---------------------------------------------------------------*/
      &TYPE = "Y";
      get_us_taxes();
      &SRC_RECORD_SCROLL = "PAY_TAX";
      Decide_Src_Fields();
      &TYPE = "C";
      get_us_taxes();
   Else
      If PAYGROUP_TBL.COUNTRY = "CAN" Then
         &WHERE_CLAUSE = &GEN_CUR_WHERE;
         /*---Read current data-------------------------------------------*/
         ScrollFlush(Record.PAY_TAX_CAN);
         ScrollSelect(1, Record.PAY_TAX_CAN, Record.PAY_TAX_CAN, &WHERE_CLAUSE, True);
         /*---Read YTD data -----------------------------------------------*/
         &WHERE_CLAUSE = &GEN_YTD_WHERE | "( SELECT MAX(TB2.BALANCE_PERIOD) FROM PS_CAN_TAX_BALANCE TB2
WHERE TB2.EMPLID = B1.EMPLID AND TB2.COMPANY = B1.COMPANY AND TB2.BALANCE_ID = B1.BALANCE_ID AND
TB2.BALANCE_YEAR = B1.BALANCE_YEAR AND TB2.WAGE_LOSS_PLAN = B1.WAGE_LOSS_PLAN AND TB2.PROVINCE =
B1.PROVINCE AND TB2.TAX_CLASS_CAN = B1.TAX_CLASS_CAN AND TB2.BALANCE_PERIOD <= " |
PY_IC_PI_WRK.BALANCE_PERIOD | " )";
         ScrollFlush(Record.CAN_TAX_BALANCE);
         ScrollSelect(1, Record.CAN_TAX_BALANCE, Record.CAN_TAX_BALANCE, &WHERE_CLAUSE, True);
         get_can_taxes();
      End-If;
   End-If;
   check_for_no_data();
End-Function;


Function Pay_Inquiry_Main();
   /*---------------------------------------------------------------------*/
   /*---------------------Main Code starts here---------------------------*/
   /*---------------------------------------------------------------------*/
   PY_IC_PI_WRK.EFFDT = PY_IC_PI_SUM_VW.PAY_END_DT_ALT;
   &OPR_LANG_CD = %Language;
   &TAX_CLASS_FLD = "TAX_CLASS";
   &GARN_TYPE_FLD = "GARN_TYPE";
   PY_IC_PI_WRK.PAY_WEB_TXGRS_CUR = 0;
   PY_IC_PI_WRK.PAY_WEB_TXGRS_YTD = 0;
   &NeedXfootMsg = "N";
```

```
Get_Current_Chk();

If All(PY_IC_PI_SUM_VW.PAY_END_DT_ALT) Then

   bld_gen_where_clause();
   get_balance_data();
   bld_gen_ytd_where_clause();
   populate_earnings();
   populate_deductions();
   populate_taxes();
   Populate_Ytd_Check(&GEN_YTD_WHERE);

   populate_ded_btmx();
   populate_ded_atmx();
   populate_ded_ermx();
   Determine_max_ded_rows();
   &TGT_RECORD_SCROLL = "PY_IC_PI_BT_MX";
   Decide_Tgt_Fields();
   &Total_Current_Amt = &Total_Cur_Amt_bt;
   &Total_Ytd_Amt = &Total_Ytd_Amt_bt;
   Pad_to_max_rows(&Row_Num_bt, &Max_rows, &Total_Current_Amt, &Total_Ytd_Amt, 0);
   &TGT_RECORD_SCROLL = "PY_IC_PI_AT_MX";
   Decide_Tgt_Fields();
   &Total_Current_Amt = &Total_Cur_Amt_at;
   &Total_Ytd_Amt = &Total_Ytd_Amt_at;
   Pad_to_max_rows(&Row_Num_at, &Max_rows, &Total_Current_Amt, &Total_Ytd_Amt, 0);
   &TGT_RECORD_SCROLL = "PY_IC_PI_ER_MX";
   Decide_Tgt_Fields();
   &Total_Current_Amt = &Total_Cur_Amt_er;
   &Total_Ytd_Amt = &Total_Ytd_Amt_er;
   Pad_to_max_rows(&Row_Num_er, &Max_rows, &Total_Current_Amt, &Total_Ytd_Amt, 0);

   populate_ded_ern_mx();

   populate_ded_tax_mx();
   Determine_max_ERN_TAX_rows();
   &TGT_RECORD_SCROLL = "PY_IC_PI_ERN_MX";
   Decide_Tgt_Fields();
   &Total_Current_Amt = PY_IC_PI_SUM_VW.TOTAL_EARNINGS;
   &Total_Ytd_Amt = PY_IC_PI_WRK.TOTAL_GROSS_YTD;
   Pad_to_max_rows(&Row_Num_ern, &Max_rows, &Total_Current_Amt, &Total_Ytd_Amt, &Total_Current_Hrs);

   &TGT_RECORD_SCROLL = "PY_IC_PI_TAX_MX";
   Decide_Tgt_Fields();
   &Total_Current_Amt = PY_IC_PI_SUM_VW.TOTAL_TAXES;
   &Total_Ytd_Amt = PY_IC_PI_WRK.TOTAL_TAXES_YTD;
   Pad_to_max_rows(&Row_Num_tax, &Max_rows, &Total_Current_Amt, &Total_Ytd_Amt, 0);

   &COUNTDBROWS = ActiveRowCount(Record.PY_IC_PI_DEDN);

   Populate_Pay_Distribution(&GEN_CUR_WHERE);

   /****  HIDE THE LEAVE GRID IF THIS IS NOT THE CURRENT CHECK  ****/

   &DISPLAY_LVE = "N";

   /*** If &CURRENT_PAY_END_DT = PAY_END_DT_ALT Then ***/
   If &CURRENT_CHK = "Y" Then
      &DISPLAY_LVE = "Y";
   End-If;

   If &DISPLAY_LVE = "N" Then
      PY_IC_WRK0.HIDE_FIELDS.Visible = False;
   Else
      PY_IC_WRK0.HIDE_FIELDS.Visible = True;
   End-If;

   &COMPANY = PY_IC_PI_SUM_VW.COMPANY;

   /***  DOES THE COMPANY ISSUE A SINGLE CHECK OR MULTIPLE CHECKS  ****/
```

```
      If &DISPLAY_LVE = "Y" Then
         SQLExec("SELECT A.SINGLE_CHECK FROM PS_COMPANY_TBL A WHERE A.COMPANY = :1", &COMPANY,
&SINGLE_CHECK);
      End-If;

      &DATA_EXIST = "N";

      /***  LEAVE BALANCE LOGIC FOR MULTIPLE JOB EMPLOYEE RECEIVING MULTIPLE CHECKS  ****/

      If &DISPLAY_LVE = "Y" And
            &SINGLE_CHECK = "N" Then
         &CURR_PAYCHECK_NBR = PY_IC_PI_SUM_VW.PAYCHECK_NBR;
         If &CURRENT_CHK = "Y" Then
            SQLExec("SELECT PAYCHECK_NBR FROM PS_PAY_CHECK WHERE EMPLID = :1 AND PAYCHECK_NBR <> :2 AND
PAYCHECK_STATUS = 'F' AND PAYCHECK_OPTION <> 'R' AND PAY_END_DT = %DateIn(:3) AND PAYCHECK_NBR <> :4",
PY_IC_PI_SUM_VW.EMPLID, &CURR_PAYCHECK_NBR, &CURRENT_PAY_END_DT, &CURR_PAYCHECK_NBR, &MJ_PAYCHECK_NBR);
            &CURR_PAYCHECK_NBR = &MJ_PAYCHECK_NBR;
            populate_ded_lveMJmx();
         End-If;

      End-If;

      /***   LEAVE BALANCE LOGIC FOR AN EMPLOYEE RECEIVING A SINGLE CHECK   *****/

      If &DISPLAY_LVE = "Y" And
            &SINGLE_CHECK = "Y" Then
         &CURR_PAYCHECK_NBR = PY_IC_PI_SUM_VW.PAYCHECK_NBR;
         Get_Current_Bnbr();
         populate_ded_lveymx();
      End-If;

      /***  HIDE THE LEAVE BALANCE GRID IF LEAVE DATA DOES NOT EXIST  ****/

      If &DATA_EXIST = "N" Then
         PY_IC_WRK0.HIDE_FIELDS.Visible = False;
      End-If;
   End-If;

   /* FLSA cross foot message */
   PY_IC_PI_WRK.EPAY_INST_TEXT.Value = "";
   If &CheckCountry = "USA" Then
      If &NeedXfootMsg = "Y" Then
         PY_IC_PI_WRK.EPAY_INST_TEXT.Value = MsgGetExplainText(2001, 727, "FLSA cross foot message");
         PY_IC_PI_WRK.EPAY_INST_TEXT.Visible = True;
      Else
         PY_IC_PI_WRK.EPAY_INST_TEXT.Visible = False;
      End-If;
   Else
      PY_IC_PI_WRK.EPAY_INST_TEXT.Visible = False;
   End-If;

End-Function;
```

PeopleCode (Page): RUNCTL_TAX810NE.Activate

```
Page.RUNCTL_TAX810NM.Visible = False;
Page.RUNCTL_TAX810NY.Visible = False;
Page.RUNCTL_TAX810ND.Visible = False;
Page.RUNCTL_TAX810OH.Visible = False;
Page.RUNCTL_TAX810OK.Visible = False;
Page.RUNCTL_TAX810OR.Visible = False;
Page.RUNCTL_TAX810PA.Visible = False;
Page.RUNCTL_TAX810PR.Visible = False;
Page.RUNCTL_TAX810RI.Visible = False;
Page.RUNCTL_TAX810XX3.Visible = False;

Gray(DERIVED_PAY.TX810_3_BTN1);
UnGray(DERIVED_PAY.TX810_3_BTN2);
```

```
UnGray(DERIVED_PAY.TX810_3_BTN3);

RC_QTR_UI.REPORTING_MEDIUM = "C";
RC_QTR_UI.DISKETTE_TYPE = "0";
```