VŠEOBECNÁ ZDRAVOTNÍ POJIŠŤOVNA ČESKÉ REPUBLIKY

Integrace aplikace s TS

Příloha standardů a podmínek dodávek informačního systému VZP ČR

Verze: 1.0
Datum: 1.12.2011

Obsah

Historie dokumentu

Verze	Datum	Autor	Popis
1.00	1.12.2011	ÚICT VZP ČR	Vytvoření dokumentu

1. Úvod

Dokument obsahuje sadu standardů pro vybudování integračních vazeb nově dodávaných komponent informačního systému se stávajícími komponentami prostřednictvím integrační platformy v souladu se Standardy ICT VZP ČR. Vytvořené standardy jsou základem pro další rozšiřování systému zaváděním nových komponent a to jak „standardních“, tak i vytvářených dle specifických požadavků VZP ČR. Tento dokument je součástí výše uvedených Standardů ICT.

V případě specifikace rozšíření informačního systému zaváděním nových komponent ve smlouvě s dodavatelem, má specifikace uváděná v této smlouvě přednost před Standardy.

Jeden z hlavních důvodů vzniku společného tiskového subsystému (TS) je hospodaření nad zdroji tiskových serverů. Druhým majoritním důvodem, který je nutné zdůraznit, je jednotná správa nad úložištěm tiskových sestav a jednotný způsob práce s tiskovými subsystémy všech aplikací.

2. FUNKCIONALITA

2.1 GENEROVÁNÍ SESTAV

Základní funkcionalitou tiskového systému je generování tiskových sestav. Aplikace předávají tiskovému subsystému identifikace sestavy s parametry.Tiskový subsystém parametry nekontroluje. Kontrola parametrů je v kompetenci klientské aplikace.

Formáty tisku

Tiskový subsystém je postaven nad technologií Oracle Reports, poskytuje tedy formáty tisky dostupné z tohoto prostředí. Jimi jsou: PDF.

Asynchronní způsob volání

Dosud bylo možné spustit sestavu buď asynchronně nebo synchronně. To znamená buď je sestava generována na pozadí (uživatel může dále pracovat s aplikací) nebo je aplikace do doby vygenerování sestavy blokována.

Nový tiskový subsystém je do značné míry omezen architekturou v tom smyslu, že nelze zajistit (a ani není vhodné) pro každého uživatele vlastní stroj sestav. Z toho plyne nutnost předávat požadavky na tisk výhradně asynchronně.

API tiskového subsystému však disponuje nástroji, které umožní synchronizaci, resp. blokování aplikace do doby, než je požadavek vyřízen.

Možnost plánování tisků

Na přepážkových pracovištích je téměř vždy nutné sestavu vygenerovat co nejdříve. Na ostatních pracovištích se může stát, že uživateli stačí aby sestava byla „zařazena“ do tiskové fronty nebo dokonce vygenerována i do několika hodin nebo i do druhého dne.

Tiskový subsystém muže tyto „odložené“ sestavy vytisknout například, když jsou volné zdroje nebo i v době, kdy jsou přepážková pracoviště uzavřena.

Prioritizace tisků

Při provozu tiskového subsystému může nastat situace, kde je požadováno více tisků než je zdrojů (tiskových serverů). V tomto případě je vhodné aby sestavy požadované z přepážkových pracovišť byly tištěny s nejvyšší prioritou.

Tisk alternativními nástroji

Tiskový subsystém bude mít pod kontrolou veškerou produkci tiskových výstupů, tedy i sestav produkovaných jinými cestami než využitím Report Services. U těchto typů sestav nebudou požadavky předávány Report Serveru, nýbrž tiskový subsystém sám zařídí spuštění příslušného aplikačního modulu na databázovém serveru a dopravení výstupu do zvoleného úložiště sestav.

2.2 ULOŽENÍ SESTAV

Jen na obrazovku, k archivaci, k opětovnému přístupu

Pří zadání požadavků na generování sestavy je jeden z parametrů forma uložení sestavy.

Soubor tiskové sestavy je po vytvoření uložen na centrálním úložišti.

V některých případech je ale dostačující sestavu pouze zobrazit **na obrazovce** (popřípadě umožnit její tisk) s tím, že soubor tiskové sestavy je posléze automaticky zrušen. To znamená není přístupný pro pozdější prohlížení nebo tisk. Vytvoření sestavy se přesto zaeviduje do seznamu vytvořených sestav.

Dalším typem sestav jsou sestavy ukládané **k opětovnému přístupu.** Tyto sestavy nebudou po prvním zobrazení na obrazovce rušeny.

Důležité tiskové výstupy budou určeny k archivaci. Tento fakt bude uveden v evidenci a podle něj bude řízena archivace (odklad) tiskových výstupů.

Způsob uložení výstupu bude definován na dvou úrovních. Vyšší úroveň je definována autorem aplikace. Způsob uložení výstupu je definován v katalogu sestav a uživatel není oprávněn jej měnit.

Na nižší úrovni definuje způsob uložení uživatel, který sestavu spouští, ovšem pouze u sestav, u nichž je to dovoleno autorem (uvedeno v katalogu sestav).

2.3 PROHLÍŽENÍ SESTAV

Všechny vytvořené sestavy jsou evidovány do společného žurnálu sestav. V tomto katalogu lze provádět různé operace v závislosti na oprávněních uživatele.

Administrátor tiskového subsystému

Administrátor tiskového subsystému má možnost prohlížet všechny sestavy pro všechny aplikace a všechny uživatele. Administrátor má k dispozici všechny potřebné informace – identifikační údaje sestavy, parametry zpracování, zpracovatele i místo uložení sestavy. V režimu aktualizace je možné sestavu odstranit, hromadně zrušit více sestav, přesunout do jiného úložiště. V režimu prohlížení lze sestavu prohlížet popřípadě uložit kopii sestavy na klientské PC.

Administrátor aplikace

Tiskový subsystém nabízí administrátorovi aplikace stejnou funkcionalitu jako administrátorovi celého tiskového subsystému s tím rozdílem, že administrátor aplikace spravuje sestavy pouze pro danou aplikaci nikoliv pro všechny aplikace.

Uživatelé aplikace

Běžní uživatelé aplikace mají k dispozici formulář pro prohlížení, mazání a vyhledávání sestav vytvořené daným uživatelem. Formulář umožňuje vyhledávání sestav podle výběrových kritérií.

2.4 ADMINISTRACE TISKOVÉHO SYBSYSTÉMU

Administrátor systému kromě práce s prohlížecími formuláři sestav, kde má možnost jednotlivé sestavy prohlížet, mazat, archivovat nebo přemístit na jiné úložiště, má ve své kompetenci i rozhodnutí o fyzickém umístění generovaných sestav *pro jednotlivé aplikace*.

Definování úložišť

Každá aplikace může mít nadefinováno své vlastní úložiště pro sestavy nebo i několik úložišť. Pro každou aplikaci je tedy určen identifikační kód, pomocí kterého se aplikace identifikuje směrem k tiskovému subsystému.

Odklad sestav

Tiskový subsystém eviduje tiskové výstupy určené k archivaci. Vlastní archivace bude spouštěna administrátorem speciální úlohou, která bude umožňovat vytvářet archivačního média (předpokládá se archivace na CD-ROM 700 MB, vlastní vypalování řešeno nebude) a vymazání zaarchivovaných výstupů z úložiště sestav. Zařazením tiskového výstupu na archivační médium bude zaznamenáno v evidenci, včetně identifikace média. Vlastní formát archivačních médií bude zpřesňován v průběhu vývoje tiskového subsystému.

3. APLIKAČNÍ ROZHRANÍ

Většina sestav je spouštěna z formuláře, proto jako základ pro vývojáře bude sloužit API implementované v package **PBREPORT** v knihovně **PBREP.pll**. Existují též sestavy spouštěné mimo prostředí formulářů, např. validační sestavy. Musí tedy existovat též aplikační rozhraní přístupné mimo toto prostředí, nejlépe uložené přímo v databázi. Bude použito též pro registraci sestav (typicky SQL skriptem spouštěným při upgrade aplikace).

Jako vhodný model se jeví vrstvená architektura, tj. základní API bude uloženo v databázi v package **PBREP** a přístupné neinteraktivním úlohám (validace apod.) a API ve formulářové PBREP.pll implementovat s využitím základního API.

3.1 KNIHOVNA PBREP.PLL

3.1.1 Konstanty

Package PBREPORT v knihovně PBREP.pll obsahuje následující veřejné konstanty:

Pro typy výstupu:

OUTPUT_SCREEN – na obrazovku

OUTPUT_STORE – k opětovnému přístupu

OUTPUT_ARCHIVE – archivovat

OUTPUT_CHOOSE – uživatelská volba

Pro stavy zpracování požadavku:

STATUS_NEW – nový požadavek

STATUS_WAITING – čeká na zdroje

STATUS_SLEEPING – nelze vyřídit, neuplynul určený okamžik naplánovaného spuštění

STATUS_PENDING – zahájeno vyřízení

STATUS_OK – vyřízen úspěšně

STATUS_ERROR – vyřízen s chybou

STATUS_VIEWED – výstup již byl zobrazen (při volbě OUTPUT_SCREEN to znamená, že výstup již není k dispozici)

STATUS_STORED – uloženo pro další prohlížení, tj. je dostupné pro prohlížení

STATUS_ARCHIVED – výstup odložen na archivačním médiu

Package PBREPORT v knihovně PBREP.pll obsahuje následující veřejné výjimky (exceptions):

UNKNOWN_APP – neznámá (neregistrovaná) aplikace

NOT_INITIALIZED – nebyl inicializován tiskový subsystém

UNKNOWN_REP – neznámý (neregistrovaný) typ sestavy

UNKNOWN_PAR – neznámý (neregistrovaný) parametr

INVALID_TYPE – požadován výstup nepovoleného typu

INVALID_PRIORITY – požadavek s nepovolenou prioritou

INVALID_PLAN – požadavek s chybným plánem

3.1.2 Funkce a procedury

Dále knihovna obsahuje následující veřejné funkce a procedury:

apiver – verze API

ver – verze implementace

mission – typ nasazení

init – inicializace tiskového subsystému

run – požadavek na tvorbu sestavy

name – název tiskového výstupu

url – URL pro získání tiskového výstupu

status – stav vyřizování požadavku na tvorbu sestavy

lor – seznam sestav aplikace

plist – sestavení seznamu parametrů sestavy

plist_destroy – zrušení seznamu parametrů

selectable_output – ověření, zda si uživatel může vybrat druh výstupu

requests – počet nezpracovaných požadavků

outputs – počet nových výstupů

trace_on – zahájení trasování (pro ladění)

trace_off – ukončení trasování (pro ladění)

Popis jednotlivých funkcí a procedur je uveden v následujících odstavcích.

3.1.2.1 Funkce api ver

Prototyp: FUNCTION apiver RETURN varchar2;

Popis: Funkce *apiver* vrátí identifikaci verze API.

3.1.2.2 Funkce ver

Prototyp: FUNCTION ver RETURN varchar2;

Popis: Funkce *ver* vrátí identifikaci verze implementace API.

3.1.2.3 Funkce mission

Prototyp: FUNCTION mission RETURN varchar2;

Popis: Funkce *mission* je primárně určena pro odlišení při vývoji. V tzv. prázdné implementaci knihovny PBREP.pll, od níž nelze očekávat žádnou smysluplnou funkcionalitu (pouze úspěšné přeložení formuláře), bude funkce *misson* vracet řetězec *PROTOTYPE*

3.1.2.4 Procedura ini t

Prototyp: PROCEDURE init (app varchar2);

Popis: Procedura init zinicializuje tiskový subsystém pro použití aplikací app.

Výjimky: UNKNOWN_APP

3.1.2.5 Funkce run

Prototypy:

FUNCTION run (rep_id integer, par paramlist, typ varchar2 default null,

sync boolean default false,

pri integer default 0, target varchar2 default null)

RETURN integer;

FUNCTION run (rep_id integer, par paramlist, typ varchar2 default null,

plan date, target varchar2 default null, after date default sysdate)

RETURN integer;

Popis: Funkce předají požadavek na vytištění sestavy rep_id s parametry par. Parametr typ určuje, zda je požadováno vytvoření sestavy pro zobrazení na obrazovce, k opětovnému přístupu či k archivaci. V případě, že u sestavy rep_id je určen typ výstupu (viz funkce selectable_output), je hodnota parametru typ ignorována.

Prostřednictvím parametru target lze určit, do jakého úložiště se má vzniklá sestava uložit. Implicitně se ukládá do úložiště příslušné aplikace, resp. sestavy.

V první variantě lze nastavit, že má být aplikace blokována do vyřízení požadavku (parametr sync), a dále snížit prioritu o hodnotu parametru pri. Implicitní priorita tvorby daného typu sestavy je definována při registraci sestavy (viz dále). Tato funkce reprezentuje požadavek na okamžité vytvoření sestavy.

Ve druhé variantě lze určit časový limit (parametr plan), do kdy je vytvoření sestavy požadováno. Vytvoření sestavy je tak odloženo, neproběhne okamžitě, ale v závislosti na vytížení tiskového subsystému, nejpozději v čase plan. Prostřednictvím parametru after lze naplánovat spuštění sestavy na určitý čas Vždy však musí být splněna podmínka after <= plan.

Obě varianty funkce run vrací identifikátor požadavku.

Výjimky: NOT_INITIALIZED, UNKNOWN_REP, UNKNOWN_PAR, INVALID_TYPE,

INVALID_PRIORITY, INVALID_PLAN.

3.1.2.6 Funkce name

Prototyp: FUNCTION name (job_id integer) RETURN varchar2;

Popis: Funkce *name* vrací název výstupního souboru vygenerovaného vyřízením požadavku *job_id.* V případě, že požadavek ještě nebyl zpracován, funkce vrací NULL.

3.1.2.7 Funkce url

Prototyp: FUNCTION url (job_id integer) RETURN varchar2;

Popis: Funkce *url* vrací URL pro přístup k výstupnímu souboru vygenerovanému vyřízením požadavku *job_id.* V případě, že požadavek ještě nebyl zpracován, funkce vrací NULL.

3.1.2.8 Funkce status

Prototyp: FUNCTION status (job_id integer) RETURN varchar2;

Popis: Funkce *status* vrací stav zpracování požadavku *job_id.* Funkce vrací některou z hodnot určených konstantami STATUS_xy.

3.1.2.9 Procedura lor

Prototyp: PROCEDURE lor (name varchar2);

Popis: Procedura je určena pro sestavení seznamu typů sestav pro LOV (list of values) nazvaný *name.*

3.1.2.10 Funkce pli st

Prototyp: FUNCTION plist (rep_id integer) RETURN paramlist;

Popis: Funkce sestaví a vrátí seznam parametrů sestavy *rep_id,* do něhož lze vyplnit hodnoty jednotlivých parametrů pro spuštění sestavy pomocí vestavěné funkce SET_PARAMETER_ATTR. Seznam parametrů je sestaven na základě informací z katalogu typů sestav, implicitní hodnoty parametrů jsou předvyplněny.

Výjimky: NOT_INITIALIZED, UNKNOWN_REP

3.1.2.11 Procedura plist_destroy

Prototyp: PROCEDURE plist_destroy;

Popis: Procedura zruší seznam parametrů dříve vytvořený funkcí *plist.*

3.1.2.12 Funkce selectable_output

Prototyp: FUNCTION selectable_output (rep_id integer) RETURN boolean;

Popis: Funkce vrací TRUE, pokud sestava *rep_id* umožňuje uživateli zvolit typ výstupu. V opačném případě vrací FALSE.

Výjimky: NOT_INITIALIZED, UNKNOWN_REP

3.1.2.13 Funkce requests

Prototyp: FUNCTION requests RETURN integer;

Popis: Funkce *requests* vrací počet požadavků zadaných uživatelem, které jsou ve stavu NEW, WAITING nebo SLEEPING. Funkci je vhodné volat například každých 5 minut a její výsledek zobrazovat v hlavním formuláři, podobně jako funkci *outputs.*

3.1.2.14 Funkce outputs

Prototyp: FUNCTION outputs RETURN integer;

Popis: Funkce *outputs* vrací počet požadavků zadaných uživatelem, které jsou ve stavu OK nebo ERROR. Funkci je vhodné volat například každých 5 minut a její výsledek zobrazovat v hlavním formuláři, podobně jako funkci *requests.*

3.1.2.15 Procedura trace_on

Prototyp: PROCEDURE trace_on;

Popis: Procedura *trace_on* zapne logování API, které je určeno pro účely ladění.

Procedura trace_off

Prototyp: PROCEDURE trace_off;

Popis: Procedura *trace_off* vypne logování API, které je určeno pro účely ladění.

3.2 PACKAGE PBREP

3.2.1 Funkce a procedury

Package *PBREP* obsahuje následující veřejné funkce a procedury:

apiver – verze API

ver – verze implementace

mission – typ nasazení

reg_app – registrace nové aplikace

reg_rep – registrace nového typu sestav

reg_rep_alt – registrace nového typu sestav, který nebude produkován

Report Serverem, nýbrž alternativním aplikačním modulem

reg_par – registrace parametru sestavy

init – inicializace tiskového subsystému pro zadávání

požadavků na tvorbu sestav

prepare – zahájení požadavku na tvorbu sestavy

add_par – zadání parametru pro tvorbu sestavy

selectable_output – ověření, zda si uživatel může vybrat druh výstupu

run – předání požadavku na tvorbu sestavy

name – název výstupního souboru

url – URL pro získání výstupního souboru

status – stav vyřizování požadavku

lor – sestavení seznamu typů sestav

lor_get – získání položky seznamu typů sestav

lop – sestavení seznamu parametrů typu sestavy

lop_get – získání položky seznamu parametrů typu sestavy

requests – počet nezpracovaných požadavků

outputs – počet nových výstupů

trace_on – zahájení trasování (pro ladění)

trace_off – ukončení trasování (pro ladění)

Podrobný popis funkcí a procedur následuje.

3.2.1.1 Typ rep_type

Typ *rep_type* je definován takto:

```
TYPE rep_type IS RECORD (
    id integer,
    name varchar2(100)
 );
```

Komponenta *id* obsahuje identifikátor sestavy pod kterým byla registrována, komponenta *name* obsahuje název sestavy.

3.2.1.2 Typ par_type

Typ *par_type* je definován takto:

```
TYPE par_type IS RECORD (
    name varchar2(100),
    value varchar2(100)
);
```

Komponenta *name* obsahuje název parametru sestavy, komponenta *value* obsahuje řetězcovou (znakovou) reprezentaci hodnoty parametru.

3.2.1.3 Výjimky

Package PBREP definuje následující seznam veřejných výjimek (exceptions). U každé je uvedena příčina vzniku.

DUP_APP_ID – duplicitní identifikátor aplikace

DUP_REP_ID – duplicitní identifikátor sestavy

DUP_PAR_ID – duplicitní název parametru

UNKNOWN_APP – neznámá aplikace

UNKNOWN_REP – neznámá sestava

UNKNOWN_PAR – neznámý parametr

NOT_INITIALIZED – tiskový subsystém není inicializován

NOT_PREPARED – nebyla zahájena příprava požadavku

INVALID_TYPE – nepovolený typ sestavy

INVALID_PRIORITY – nepovolená změna priority

INVALID_PLAN – chybná specifikace plánu spuštění sestavy

3.2.1.4 Funkce api ver

Prototyp: FUNCTION apiver RETURN varchar2;

PRAGMA RESTRICT_REFERENCES (apiver,wnds,wnps);

Popis: Funkce *apiver* vrátí identifikaci verze API.

3.2.1.5 Funkce ver

Prototyp: FUNCTION ver RETURN varchar2;

PRAGMA RESTRICT_REFERENCES(ver,wnds,wnps);

Popis: Funkce *ver* vrátí identifikaci verze implementace API.

3.2.1.6 Funkce mission

Prototyp: FUNCTION mission RETURN varchar2;

PRAGMA RESTRICT_REFERENCES(mission,wnds,wnps);

Popis: Funkce *mission* je primárně určena pro odlišení při vývoji. V tzv. prázdné implementaci package PBREP, od níž nelze očekávat žádnou smysluplnou funkcionalitu, bude funkce misson vracet řetězec *PROTOTYPE.*

3.2.1.7 Procedura reg_app

Prototyp:

PROCEDURE reg_app (id varchar2, name varchar2, pri integer, target varchar2);

Popis: Procedura zaregistruje aplikaci *id,* s názvem *name.* Implicitní priorita požadavků na tvorbu sestav pro aplikaci je *pri,* implicitní úložiště výstupních souborů je *target.*

Výjimky: Procedura může skončit výjimkou *DUP_APP_ID.*

3.2.1.8 Procedura reg_rep

Prototyp:

PROCEDURE reg_rep (app_id varchar2, id integer, name varchar2, rdf varchar2,

output_type varchar2, pri integer default null, target varchar2 default null,

regular_behaviour boolean default true);

Popis: Procedura zaregistruje v rámci aplikace app_id nový typ sestavy id. Název sestavy je name, sestavy budou generovány modulem rdf. Typ výstupu určuje parametr output_type, povoleny jsou hodnoty konstant OUTPUT_%. Implicitní prioritu typu sestavy určuje parametr pri (není-li uveden, platí implicitní priorita aplikace). Implicitní cílové úložiště typu sestavy udává parametr target (není-li uveden, platí implicitní úložiště aplikace). Parametr regular_behaviour určuje způsob zacházení se sestavou. Implicitně bude nabývat hodnoty true, která určuje normální zacházení se sestavami registrovaného typu – sestavy jsou tisknuty a registrovány do žurnálu pomocí tiskového subsystému. Pokud parametr regular_behaviour bude při registraci nastaven na hodnotu false, bude zacházení se sestavami registrovaného typu změněno takto:

• bude možné spouštět funkci special_store_report_info pro zaevidování hotové sestavy

• funkce run pro takto registrovaný typ sestav bude končit výjimkou UNKNOWN_REP

Výjimky: Procedura může skončit výjimkami DUP_REP_ID, INVALID_PRIORITY,

INVALID_TYPE.

3.2.1.9 Procedura reg_rep_alt

Prototyp:

PROCEDURE reg_rep_alt (app_id varchar2, id integer, name varchar2, module

varchar2, loc varchar2, output_type varchar2, pri integer default null, target

varchar2 default null);

Popis: Procedura zaregistruje v rámci aplikace app_id nový typ sestavy id, který nebude generován prostřednictvím Report Serveru, ale alternativním modulem module, který je umístěn na databázovém serveru v adresáři loc. Význam ostatních parametrů procedury je shodný jako u procedury reg_rep.

3.2.1.10 Procedura reg_par

Prototyp:

PROCEDURE reg_par (app_id varchar2, rep_id integer, name varchar2, dvalue

varchar2 default null, hidden boolean default false, utext varchar2 default null);

Popis: Procedura zaregistruje parametr name typu sestavy rep_id aplikace app_id. Implicitní hodnota parametru je udána hodnotou dvalue. Parametr hidden určuje, zda se jedná o tzv. skrýtý parametr – není viditelný pro uživatele. Parametr utext umožňuje zadat komentář parametru.

Výjimky: Procedura může skončit výjimkou *DUP_PAR_ID*.

3.2.1.11 Procedura ini t

Prototyp: PROCEDURE init (app varchar2, un varchar2, pw varchar2, cs varchar2);

Popis: Procedura init zinicializuje tiskový subsystém pro použití aplikací app. Při inicializaci se nastavuje uživatelské jméno un a heslo pw uživatele, dále pak cs connect string. Heslo je chráněno před eventuálním zneužitím tak, že je v čitelné formě předáno pouze formálnímu parametru pw; pro další použití je zašifrováno šifrovacím systémem DES.

Výjimky: Procedura může skončit výjimkou *UNKNOWN_APP*.

3.2.1.12 Procedura prepare

Prototyp: PROCEDURE prepare (rep_id integer);

Popis: Procedura *prepare* zahájí přípravu požadavku na spuštění sestavy *rep_id.*

Výjimky: Procedura může skončit výjimkou *UNKNOWN_REP, NOT_INITIALIZED*.

3.2.1.13 Procedura add_par

Prototyp: PROCEDURE add_par (name varchar2, value varchar2);

Popis: Procedura připojí parametr *name* s hodnotou *value* do připravovaného požadavku na spuštění sestavy.

Výjimky: Procedura může skončit výjimkou *UNKNOWN_PAR, NOT_INITIALIZED,*

NOT_PREPARED.

3.2.1.14 Funkce run

Prototypy:

FUNCTION run (rep_id integer, typ varchar2,

pri integer default 0, target varchar2 default null)

RETURN integer;

FUNCTION run (rep_id integer, typ varchar2,

plan date, target varchar2 default null, after date default sysdate)

RETURN integer;

Popis: Činnost funkce je popsána výše u funkce pbrep_run. Funkce předá požadavek na vytvoření sestavy *rep_id* s parametry, které byly zadány voláním procedury *add_par* od posledního zavolání procedury *prepare.*

Výjimky: Procedura může skončit výjimkou *NOT_PREPARED, INVALID_TYPE,*

INVALID_PRIORITY.

3.2.1.15 Funkce name

Prototyp: FUNCTION name (job_id integer) RETURN varchar2;

Popis: Funkce *name* vrací název výstupního souboru vygenerovaného vyřízením požadavku *job_id.* V případě, že požadavek ještě nebyl zpracován, funkce vrací NULL.

3.2.1.16 Funkce url

Prototyp: FUNCTION url (job_id integer) RETURN varchar2;

Popis: Funkce *url* vrací URL pro přístup k výstupnímu souboru vygenerovanému vyřízením požadavku *job_id.* V případě, že požadavek ještě nebyl zpracován funkce vrací NULL.

3.2.1.17 Funkce status

Prototyp: FUNCTION status (job_id integer) RETURN varchar2;

Popis: Funkce *status* vrací stav zpracování požadavku *job_id.* Seznam možných návratových hodnot je uveden výše v popisu funkce pbrep_status.

3.2.1.18 Funkce lor

Prototyp: FUNCTION lor (app_id varchar2) RETURN integer;

Popis: Funkce *lor* sestaví seznam typů sestav aplikace *app_id.* Funkce vrací počet položek sestaveného seznamu.

Výjimky: Procedura může skončit výjimkou *UNKNOWN_APP*.

3.2.1.19 Funkce lor_get

Prototyp: FUNCTION lor_get (i integer) RETURN rep_type;

Popis: Funkce vrátí *i*-tou položku seznamu sestaveného funkcí *lor.*

Výjimky: Procedura může skončit výjimkou *UNKNOWN_REP*.

3.2.1.20 Funkce lop

Prototyp: FUNCTION lop (rep_id integer) RETURN integer;

Popis: Funkce *lop* sestaví seznam parametrů sestavy *rep_id.* Funkce vrací počet položek sestaveného seznamu.

Výjimky: Procedura může skončit výjimkou *UNKNOWN_REP*.

3.2.1.21 Funkce lop_get

Prototyp: FUNCTION lop_get (i integer) RETURN par_type;

Popis: Funkce vrátí *i*-tou položku seznamu sestaveného funkcí *lop.*

Výjimky: Procedura může skončit výjimkou *UNKNOWN_PAR*.

3.2.1.22 Funkce selectable_output

Prototyp: FUNCTION selectable_output (rep_id integer) RETURN boolean;

Popis: Funkce vrací TRUE, pokud sestava *rep_id* umožňuje uživateli zvolit typ výstupu. V opačném případě vrací FALSE.

Výjimky: NOT_INITIALIZED, UNKNOWN_REP

3.2.1.23 Funkce requests

Prototyp: FUNCTION requests RETURN integer;

Popis: Funkce *requests* vrací počet požadavků zadaných uživatelem, které jsou ve stavu NEW, WAITING nebo SLEEPING.

3.2.1.24 Funkce outputs

Prototyp: FUNCTION outputs RETURN integer;

Popis: Funkce *outputs* vrací počet požadavků zadaných uživatelem, které jsou ve stavu OK nebo ERROR.

3.2.1.25 Procedura trace_on

Prototyp: PROCEDURE trace_on;

Popis: Procedura *trace_on* zapne logování API, které je určeno pro účely ladění.

3.2.1.26 Procedura trace_off

Prototyp: PROCEDURE trace_off;

Popis: Procedura *trace_off* vypne logování API, které je určeno pro účely ladění.

3.2.1.27 Procedura special_zurnal_repadd

Prototyp: FUNCTION special_zurnal_repadd (rep_id integer, target varchar2,

file_name varchar2) RETURN integer;

Popis: Funkce special_ zurnal_repadd zaeviduje hotový tiskový výstup typu rep_id uložený v adresáři target jako soubor file_name. Funkce vrátí identifikátor tiskového výstupu.

V případě, že typ sestav rep_id není zaregistrován s regular_behaviour false funkce skončí výjimkou UNKNOWN_REP.

3.2.1.28 Čtení informací z katalogů

FUNCTION get_app_info (app_id varchar2) RETURN app_info_type;

FUNCTION get_rep_info (app_id varchar2, rep_id integer) RETURN rep_info_type;

FUNCTION get_par_info (app_id varchar2, rep_id integer, name varchar2) RETURN

par_info_type;

FUNCTION get_par_info (app_id varchar2, rep_id integer, pos integer) RETURN

par_info_type;

Nové typy app_info_type, rep_info_type a par_info_type budou záznamy, jejichž položky budou přesně specifikovány a zveřejněny v hlavičce balíku PBREP. Pro zjišťování informací o parametru budou existovat dvě varianty funkce get_par_info; v první variantě bude parametr specifikován svým názvem, ve druhé pořadím.

3.3 FORMULÁŘ PBF001_PREHLSES

Pro zobrazení žurnálu sestav slouží formulář PBF001_PREHLSES. Formulář může být zpuštěn v několika režimech – uživatelském, administrátor aplikace a administrátor tiskového subsystému. Pro volání z externích aplikací jsou přístupné první dva režimy.

Formulář má dva parametry 'P_COMPETENCE', který určuje v jakém režimu bude formulář spuštěn a parametr 'P_APPLICATION' udávající kód aplikace.Hodnota parametru P_COMPETENCE může nabývat hodnot:

'U' – běžný uživatel

'A' – administrátor aplikace

4. NÁVOD NA VYUŽÍVÁNÍ TISKOVÉHO SUBSYSTÉMU

V této kapitole naleznete návod na použití některých dostupných funkcí tiskového subsystému. Neobsahuje použíti všech API funkcí, proto VŽDY zledujte i API uvedené v předcházejících kapitolach.

Tato kapitola je návodem pro používání tiskového subsystému. Popisuje následující způsoby používání:

- Zjištění verze API a verze implementace
- Registrace aplikace
- Registrace sestavy
- Registrace sestavy nevyužívající Report Services
- Požadavek na tisk (forms API)
- Sestavení seznamu parametrů pro tiskový požadavek mimo Forms
- Požadavek na tisk (stored PL/SQL API)
- Zjištění stavu zpracování
- Prohlížení sestav

4.1 ZJIŠTĚNÍ VERZE API A VERZE IMPLEMENTACE

Zjišťování verze API a verze implementace je důležité pro ověřování funkcionality a odstraňování chyb tiskového subsystému. Vždy budou důležité 2 hodnoty, které vracejí funkce pbrep.apiver a pbrep.ver:

SELECT pbrep.apiver, pbrep.ver FROM dual;

Ve forms API též funkce pbreport.apiver a pbreport.ver, použité např. takto:

message ('Verze API: '||pbreport.apiver||' Verze implementace:

'||pbreport.ver);

Sledujte příslušné ReleaseNotes, kde jsou shrnutý nové prvky a omezení dané verze.

4.2 REGISTRACE APLIKACE

Prvním krokem k používání tiskového subsystému je zaregistrování aplikace. Tento krok je nutné provést jednou pro každou aplikaci, lze jej provést pouze pomocí stored PL/SQL API:

```
BEGIN
    pbrep.reg_app('X','Zkušební aplikace',1,'/vzpdata/data/vzpp2/x/pdf');
END;
```

Tímto voláním jsme zaregistrovali zkušební aplikaci X, s implicitní prioritou tvorby sestav 1 a implicitním úložištěm /vzpdata/data/vzpp2/x/pdf.

Aplikace registrované do tiskového subsystému by měly odpovídat aplikacím definovaným ve Forms Services. Po inicializaci tiskového subsystému budou uživateli přístupny pouze sestavy zaregistrované pod aplikací nastavenou při inicializaci.

V současné verzi tiskový subsystém rozeznává pouze 2 priority: 1 – velká, 9 – malá.

4.3 REGISTRACE SESTAVY

Existuje-li zaregistrovaná aplikace, je možné pod ní registrovat typy sestav. Opět jej lze provést pouze pomocí stored PL/SQL API:

```
BEGIN
    pbrep.reg_rep ('X',14,
    'Zkušební sestava 14 zkušební
    aplikace','XX_sestava14.rdf',pbrep.OUTPUT_SCREEN,9);
END;
```

Tímto voláním jsme zaregistrovali zkušební sestavu využívající Report Services pod aplikací X. Identifikátor typu sestavy (14) musí být unikátní v rámci aplikace X. Sestava je implementována v modulu XX_sestava14.rdf. Bude generována s malou prioritou (odlišně od implicitní priority aplikace X) do implicitního úložiště. Po prvním prohlédnutí uživatelem bude z úložiště odstraněna.

Registrace parametrů sestavy se provádí procedurou pbrep.reg_par takto:

```
BEGIN
    pbrep.reg_par('X',14,'parametr1');
    pbrep.reg_par('X',14,'parametr2','default');
END;
```

V ReleaseNotes, jsou uvedeny současná omezení přeregistrace a rušení registrace. V aplikaci musí týt korektně nastavena REPORTS_PATH.

4.4 REGISTRACE SESTAVY NEVYUŽÍVAJÍCÍ REPORT SERVICES

Sestavy, které nejsou implementovány pro Report Services, se registrují analogicky.

```
BEGIN
    pbrep.reg_rep_alt ('X',42,
        'Zkušební sestava 42 zkušební aplikace',
        'xy','/appl/vzpp2/x/prg',pbrep.OUTPUT_SCREEN,9);
```

END;

Zaregistrovali jsme sestavu 42 pod aplikací X, která bude produkována programem /appl/vzpp2/x/prg/xy

Parametry sestavy se registrují totožně jako u běžných sestav. Upozornění: v případě těchto sestav záleží na pořadí volání procedury pbrep.reg_par, které určuje pořadí parametrů při volání programu. I když jsou v tomto případě názvy parametrů irelevantní je nutné zaregistrovat každému parametru unikátní pseudonázev.

4.5 POŽADAVEK NA TISK (FORMS API)

Nejprve je nutné inicializovat tiskový subsystém procedurou init s uvedením identifikačního kódu aplikace:

pbreport.init('X');

Voláním funkce plist obdržíme seznam parametrů (datový typ parameter_list), který obsahuje všechny registrované parametry sestavy včetně nastavené implicitní hodnoty:

muj_list := pbreport.plist(14);

V seznamu parametrů formulář provede příslušné úpravy – vyplní požadované hodnoty parametrů sestavy a předá požadavek pomocí funkce run:

jobid := pbreport.run(14,muj_list);

Funkce run vrací identifikátor běhu sestavy, který lze použít k zjišťování stavu zpracování sestavy.

4.6 SESTAVENÍ SEZNAMU PARAMETRŮ PRO TISKOVÝ POŽADAVEK MIMO FORMS (PL/SQL)

K sestavení seznamu parametrů a předání požadavku mimo prostředí Forms slouží funkce:

pbrep.lop

pbrep.lop_get

pbrep.add_par

pbrep.prepare

pbrep.run

Jejich podrobný popis je uveden ve specifikaci aplikačního rozhraní tiskového subsystému. Typické použití ilustruje následující příklad:

```
declare
    r_id constant integer := 14; -- typ sestavy pro tisk;
    p pbrep.par_type;
    i integer;
begin
    ... -- cinnost predchazejici vlastnimu pozadavku na tisk, vc.
inicializace
    pbrep.prepare(r_id);
    FOR i in 1..pbrep.lop(r_id) LOOP
```

```
            p:=pbrep.lop_get(i);
            ... -- zjisteni pozadovane hodnoty parametru
            pbrep.add_par(p.name,pozadovana_hodnota);
    END LOOP;
    ... -- predani pozadavku pomoci pbrep.run
end;
```

4.7 POŽADAVEK NA TISK (STORED PL/SQL API)

Nejprve je nutné inicializovat tiskový subsystém procedurou init s uvedením identifikačního kódu aplikace, uživatelského jména a přihlašovacího hesla uživatele:

```
pbrep.init('X','uzivatel','heslo', 'pripojeni');
```

Inicializaci je nutné provést jen jednou během existence databázového spojení. Vlastní požadavek na vytvoření tiskového výstupu se předá posloupností volání procedur a funkcí prepare, add_par a run:

```
pbrep.prepare(14); -- vytvarime pozadavek na sestavu cislo 14
pbrep.add_par('parametr1','hodnota_parametru1');
pbrep.add_par('parametr2','hodnota_parametru2');
jobid := pbrep.run(14);
```

Proceduru add_par je nutné zavolat pro každý parametr sestavy, včetně těch, jejichž hodnota zůstává implicitní. Komfort, jaký poskytuje forms API prostřednictvím seznamů parametrů, v tomto rozhraní chybí.

Funkce run vrací identifikátor běhu sestavy, který lze použít k zjišťování stavu zpracování sestavy.

V ReleaseNotes, jsou uvedeny případná omezení pro spouštění generování sestavy.

4.8 ZJIŠTĚNÍ STAVU ZPRACOVÁNÍ

V obou API je definována funkce status a sada konstant, které umožňují sledovat zpracování požadavku:

```
stav := pbreport.status(jobid);
stav := pbrep.status(jobid);
```

Konstanty jsou popsány ve specifikaci API.

4.9 PROHLÍŽENÍ SESTAV

K prohlížení sestav je možné použít dvě strategie. V obou se použije funkce url příslušného API, která vrátí URL, přes které lze získat vygenerovanou sestavu (v případě, že stav zpracování je STATUS_OK):

```
moje_url := pbreport.url(jobid);
moje_url := pbrep.url(jobid);
```

Se získaným URL aplikace může provést buď:

- pomocí WEB.SHOW_DOCUMENT built-in otevřít nové okno webového prohlížeče a v něm nechat zobrazit sestavu; nebo

- pomocí standardní stored package UTL_HTTP získat obsah tiskové sestavy přímo v PL/SQL (např. do proměnné typu BLOB) a prostředky WebUtil přenést na klientskou stanici k zobrazení, tisku apod.

Výhodou první možnosti je dopravení tiskového výstupu k očím uživatele jediným řádkem kódu; výhodou druhého přístupu je utajení URL (uživatel se jej nedozví, což přispěje bezpečnosti uložení sestav).

Třetí možností je použít formulář pro sledování činnosti tiskového subsystému, který umožňuje zobrazit libovolný tiskový výstup. Způsob volání tohoto formuláře z aplikace bude ještě upřesněn.

4.10 FORMULÁŘ (PBF001_PREHLSES) - ADMINISTRÁTORSKÝ/UŽIVATELSKÝ PŘEHLED SESTAV

Začlenění formuláře do aplikace vyžaduje 2 kroky:

1. Upravit nastavení Form Services:

V souboru pro specifické nastavení aplikace aplikace.env (v adresáři /oracle/product/IAS904/forms/server) doplnit do proměnné cesty FORMS90_PATH adresář /appl3w/vzpp2/PSi/prg

2. Zavolat formulář PBF001_PREHLSES

Příklad: Uživatelský přehled sestav:

Do parametru 'P_APPLICATION' předejte kód, pod kterým byla aplikace zaregistrována v tiskovém subsystému pomocí funkce pbreg.reg_app a do parametru 'P_COMPETENCE' hodnotu 'U'

Příklad: volání Uživatelského přehledu ve Zkušební aplikaci X:

```
DECLARE
pl_id ParamList;
pl_name varchar2(13) := 'prohl_ses_par';
BEGIN
        pl_id := create_parameter_list(pl_name);
        add_parameter(pl_id, 'P_COMPETENCE', TEXT_PARAMETER, 'U');
        add_parameter(pl_id, 'P_APPLICATION', TEXT_PARAMETER, 'X');
        open_form('PBF001_PREHLSES', ACTIVATE, SESSION,
NO_SHARE_LIBRARY_DATA, pl_id);
END;
```