# Validation bal íčků

Each package is validated within its context, which is formed of the data from title package :

* both
  + packageName – název balíčku bez přípon
* remote access:
  + IDProvozovatel
  + Model
  + VykazovanéObdobí
  + DruhHry
* daily reporting
  + IDProvozovatel
  + Model
  + VykazovanéObdobí
  + DruhHry
  + HerníMísto

Validation packages is described by a set of configuration files in CSV format , which they are captured in individual excel tabs in structured-schema-description.xlsx .

Mechanism downloading packages ensures that values in name package they are always right, because to search the package is being used verified values coming from the registration. Name of the data package (without extensions) of d says the following regular expression:

Remote Access

IDProducer : [ 0-9] {9}

Model: [MV]

Date: 20 [1-9] [0-9] (01 | 02 | 03 | 04 | 05 | 06 | 07 | 07 | 09 | 10 | 11 | 12) (3 [01] | 012] [0-9 ])

Time: ( 00 | 08 | 16)

Type games : [ LKRBTZ ]

Version : (0 [1-9] | [1-9] [0-9])

/ [0-9] {9} [mV] 20 [1-9] [0-9] (?: 01 | 02 | 03 | 04 | 05 | 06 | 07 | 07 | 09 | 10 | 11 | 12) ( ?: 00 | 08 | 16 ) [ LKRBTZ] (?: 0 [1-9] | [ 1-9] [0-9]) /

daily reporting

IDProducer : [ 0-9] {9}

Model: [MV]

Moon : 20 [1-9] [0-9] (01 | 02 | 03 | 04 | 05 | 06 | 07 | 07 | 09 | 10 | 11 | 12)

Type games : [LKRBTZ]

Game point: [0-9a-zA-Z -\_] +

Version : (0 [1-9] | [1-9] [0-9])

/ [0-9] {9} [mV] 20 [1-9] [0-9] (?: 01 | 02 | 03 | 04 | 05 | 06 | 07 | 07 | 09 | 10 | 11 | 12) [ LKRBTZ] (?: 0 [1-9] | [ 1-9] [0-9]) /

## Validation electronic seals package

If there is no verification electronic seal ends treatment one of the following errors :

ERR\_ PKG\_SIG\_ INVALID\_DER - if is not possible read the ASN.1 DER structure package

ERR\_PKG\_SIG\_ INVALID\_SIGNATURE - if disagree cryptographic check signature

ERR\_PKG\_SIG\_ CERT\_UNTRUSTED - if is not possible verify validity certificate

ERR\_PKG\_SIG\_ CERT\_UN KNOWN - if certificate is not between registered certificates operator

Any error document validation ends processing of the package .

## Decryption of the package

If fails decryption package is being reported some of the errors :

ERR\_PKG\_ ENC \_ INVALID\_DER - if is not possible read the ASN.1 DER structure package

ERR\_PKG\_ENC \_ RECIPIENT - if it fails find suitable certificate recipient

ERR\_PKG\_ENC\_BAD\_DATA - if it fails unscramble

Any error decryption ends processing of the package .

## Unzip the ZIP archive

If it fails to expand ZIP file package is processing ended error ERR\_PKG\_UNZIP

Erro during unpacking ends processing of the package .

.

## Validation present File - tab "Package\_files"

According to the tab "Package\_files" it is judged that the files to be in package present. If the package contains files other than those listed in the table will be labeled as faulty, and the processing will end with the error code ERR\_PKG \_EXTRA\_FILES and enumeration of the extra files.

Error handling the present files is finished processing the package.

## Validation of required files - tab " Game\_types\_and\_files "

Operation is performed obtain a list of files using a process that can be sat at p using SQL as follows:

SELECT fileName FROM Game\_type\_and\_files as config

WHERE context. TypeHeats = config.gameType AND Context.Model = config.model

If the set of file names is not empty, then the resulting files must be present in packag. If the required file (s) is missing , the package processing is terminated with an error s ERR\_PKG\_MISSING\_FILE s by listing the missing files.

If the set is empty then no presence check is performed.

## Validate content of files

Each file is loaded in the context of a file that takes over all package context attributes and add context.fileName . The file that loads the line per line for each line is applied to the following checks:

1) the line ends with a pair of CRLF characters (0x0d 0x0a) - other characters do not end the line

2) the maximum line length must not exceed the MAX\_CSV\_LINE\_LENGTH configuration parameter, which has the default value of 64000. The line length is calculated in bytes. ERR\_LINE\_TOO\_LONG

3) the line is made entirely of valid UTF-8 characters - ERR\_ LINE\_BAD\_UTF8

4) for the first line in file, its content must match the regular expression:

and) the line can be divided according to the rules listed in the Ordinance into individual fields - ERR\_LINE\_INVALID\_CSV

b) the result of the previous step is a set of values ​​for each field, the set is validated by

C) split is done using the ";" and shall result in exactly 4 parts - is not split into 4 parts processing of the line ends with error ERR\_LINE\_SPLIT\_META.

(d) metadata obtained by splitting in the previous steps are validated as follows:

and) the first field matches the names em package without extension - if not match (case insensitive) s package name, ending field validation with error ERR\_META\_FIELD \_BAD\_PACKAGE

ii) the second field matches the filename (case insensitive ) if not, field validation ends with ERR\_META\_FIELD\_BAD\_NAME

(iii) the third field contains the date and time in the form of the same Domain type validation T\_DATETIME

(iv) the fourth field contains a text value of "1.0"

5) with respect to the second line of the file, the CSV line breakout rules are divided into a header set

6) From the " Fields " tab, you get a description of the contents of the header files in the same way as

and) SELECT FieldNameCZ FROM Fields WHERE fileName = fileName AND model = model

7) List of headers must match the list of known fields as extracted from the Fields tab. If it does not match, the whole file is finished by ERR\_LINE\_BAD\_HEADER

8) A set of field names is stored in a configuiration of the mapping function fieldIdxToFieldName ()

9) For a line other than 1st or 2nd or second, the data line validation is performed

File validation terminated if a sufficient number of defective lines are detected (CFG\_MAX\_LINE\_ERR configuration entry), and file validation ends with an error ERR\_FILE\_TOO\_MANY\_ERRORS.

## Validation of the data line

The line is divided into individual fields according to CSV readings

If the number of fields is different from the number of file header items, line processing is terminated by ERR\_LINE\_BAD\_FIELD\_COUNT

For each field value, use the function fieldIdxToFieldName () determines what is the field name ( fieldName ).

For each field are then checks are carried out by by content corresponding line in the Fields tab . Line is selected using an equivalent:

SELECT \* FROM Fields

WHERE kontext.model = Model AND kontext.fileName = fileName AND fieldName = fieldName

## Validation of fields

1) FieldType - according to field value, basic data type validation - ERR\_FIELD\_TYPE (see Field types )

2) DomainType - According to the field value, the basic data domain type validation - ERR\_ FIELD\_ DOMAIN\_TYPE (see the Domain tab types )

3) Nullable - by worthy field check is made whether or not allowed blank value - ERR\_ FIELD\_NOT\_NULL

4) Presence - ERR\_FIELD\_MANDATORY is checked by field value

and) mnandatory - non-empty value must be present

b) mandatoryIf - if the condition in the Condition column is met , a value must be present

5) Condition - A condition that is evaluated for the use of mandatoryIf

and) refFieldName.isEmpty - met when refFieldName is empty

b) refFieldName.notIn - met when refFieldName is not included in the enumeration given in parentheses

C) refFieldName.In - met when refFieldName is included in the enumeration given in parentheses

6) MinLength - Minimum string length of values ​​calculated in characters - ERR\_FIELD\_TOO\_SHORT

7) MaxLength - the maximum length of chain RET values in characters comput Tana - LONG ERR\_FIELD\_TOO\_

8) RegExp - field contents must match regulatory expression - ERR\_FIELD\_BAD\_FORMAT\_RE

9) Context - The content of the field under consideration must meet the constraints of the package context:

and) equalsGameType - identical with context.gameType - ERR\_FIELD\_CTX\_GAMETYPE

b) equalsOperatorId - identical with context.OperatorId - ERR\_FIELD\_CTX\_OPERTORID

C) equalsPackageDate - identical with kontext.PackageDate - ERR\_FIELD\_CTX\_PGKDATE

(d) startsWithOperatorId - the value prefix must be the same as context.OperatorId - ERR\_FIELD\_CTX\_OPERATORID\_PREFIX

E) withinPackageTimespan - the date and time value must fall within the time range that is covered by the package. ERR\_FIELD\_CTX\_PKG\_TIMESPAN

If it occurs error validation of any of the ERR\_LINE\_ INVALID\_FIELDS error is reported after all validations have been executed .

## Structure of error messages

Each error begins with a new line of error codes of one of the types:

/^ERR\_PKG\_.\* : /

/^ERR\_FILE\_.\* [filename.csv]: /

/^ERR\_LINE\_.\* [filename.csv] [ row number ]: /

/^ERR\_FIELD\_.\* [filename.csv] [ line number ] [ fieldName / fieldPosition ]: /

Each error record can be written to multiple rows.

Processing flow is as follows:

ERR\_PKG - Ends package processing

ERR\_FILE - finishes processing the file

ERR\_LINE - Ends line processing

ERR\_FIELD - All available validations on the item are always executed

## Structure of the error report

The error report always contains errors as described in the preceding chapters as they were created during processing.

## Structure of OK report

The confirmation report contains a single line:

OK: YYYYMMDD-HHMMSS

Where the date and time capture the moment of generating an OK report.