IE Domain Registry Ltd

IEDR BPR 2012 System Requirements Specification

Issued by:	Paul Delany, ASD Team
To:	NASK
Cc:	Dermot Tynan, Billy Glynn, David Curtin, Angela Butler
Version:	0.4
Date:	19th December 2011

Date	**Version**	**Updated By**	**Comment**
2011-12-13	0.1	Paul Delany	**For internal review only**. Supersedes the IEDR BPR Scope of Change document.
2011-12-15	0.2	Paul Delany	**For internal review only.** Changes: Updated CRS-API, CRS-IE-Beo sections. Incorporated review notes from Dermot Tynan.
2011-12-16	0.3	Paul Delany	Edited for initial review & estimation by NASK. The document is essentially still in draft, however, the major points of scope are now included. Also included are review points post internal meeting 16 Dec (DC, AB, PD, BG).
2011-12-19	0.4	Paul Delany	Edited for conformance to MoSCoW conventions.

Note: The usage of the words MUST, SHOULD, COULD and WON'T applied here are taken to conform to the MoSCoW convention. [http://en.wikipedia.org/wiki/MoSCoW_Method].

IEDR BPR 2012 System Requirements Specification

This document is currently in draft.

This document contains the business and technical requirements specifications for the IEDR BPR 2012 Project.

Business Requirements

Policy Changes

ISSUE#0003: Policy issues regarding charities.

«Political» Issue:
Open Issues for IEDR Resolution:

On registration we could set the initial renewal date for plus 3 years/5 years (TBD). (Alternative:-could set the billing status to 'AutoRenew' ?).

Prohibit 'InVoluntary' NRP by Registrars (to deal with situations where the Registrar just wants to get the charity out of his portfolio). Instead, allow transfer to IEDR's own account (for nonbillable Charities). TBD

However, allow 'voluntary' entry to NRP when authorised by the AdminC.

Allow transfers from one Registrar billC to another Registrar billC - without requiring financial payment.

REQ#0014: Invoices are issued when domains are paid for.

«Political» Requirement:
Invoices are no longer issued in arrears of the renewal date / transfer date / new registration date.

REQ#0016: The 30 day billing cycle is replaced with a 29 day renewal cycle.

«Political» Requirement:
30 day 'billing cycle' credit terms will be eliminated for all transactions (renewals, new registrations and transfers) to be replaced by a 29 day 'renewal cycle', i.e., 29 days from the renewal date until the deletion date (1+14+14=29).

REQ#0027: Admin, Technical and Financial passes for a domain are required within the 27 day period.

«Political» Requirement:
The 'triple passing' of a registration ticket is the successful completion of the three checks, in this order:

Admin: Registration services vet the ticket for adherence to IEDR policy and requirements.

Technical: The technical details of the registration request, including nameserver details, are validated.

Financial: Payment is taken with respect to the registration.

If the above three are not all completed within 27 days of the ticket being raised, then [the ticked is removed - PD to confirm].

REQ#0101: New registrations and transfers must be paid for before the ticket is completed on CRS.

«Political» Requirement:
The ticket is stalled at financial failure if the transaction fails.

REQ#0118: Processing of payment must complete before a domain is created.

«Political» Requirement:
Processing of payment must complete before a domain is created.

This is a high priority requirement of the new policy.

Procedure Changes

REQ#0017: Month-end batch processing procedures are replaced by real-time transaction processing.

«Purpose» Requirement:
Invoices and receipts are generated at the same time, on receipt of payment. As such, there are no month-end batch processing procedures.

REQ#0018: Replace MSD procedure with NRP procedure.

«Political» Requirement:
Eliminate MSD process and procedures (and therefore Credit Notes) for all transactions types (i.e., new registrations, Transfers).

Introduce new non-renewal process and procedures for renewals process (NRP). Non-renewed domains will be Mailed on RenDt+1 day (suspended after 14 days, deleted after 14 days)

REQ#0021: Eliminate off-line receipting options.

«Political» Requirement:
Eliminate the Registrars off-line (ROF) Directs off-line (GOF) procedures.

The existing PIA and ADP methods are available to registrars.

ADP payment is not available to direct registrants, however.

REQ#0098: NRP notification mails sent to direct registrants.

«Functional» Requirement:
NRP notification emails for domains held by direct registrants are send to the Admin-C as well as the Bill-C for the domain.

REQ#0103: Invoices sent to registrars and accounting system.

«Functional» Requirement:
When invoices are generated, they must be available to both Registrars for re-download and the accounting system to email via CRS.

Note: The timing of the availability of the invoice in the accounting system depends on the timing of the import of data from CRS to the accounting system.

REQ#0104: Invoice sequencing must be controlled within CRS.

«Functional» Requirement:
Invoice sequencing must be controlled within CRS.

No duplicates/no missing numbers.

REQ#0105: Statement of chronological transaction history available on CRS.

«Functional» Requirement:
Deposit account on CRS must be updated for all transactions, and CRS must display Statement of transaction/chronological Transaction History.

REQ#0030: The migration to the new processes must take a 'big bang' approach.

«Migration» Requirement:
The changeover from MSD to NRP is particularly crucial as a transient parallel operation is not feasible.

Soft launch

However, Soft Launch of certain features may be required (in order to minimise the number of domains in the billing cycle)

Not issuing the R&R or Transfers invoice(s) in month minus 1, nor month minus 2 – letting them be PIA'd instead. (if not paid, the new code will invoice them in bulk after go-live (as all will have RenDt<today)

By moving the 'D' date up by a few days, before the go-live date (to catch the MSD auto-credit note of unpaid invoices), and Dealing with the 'rump' or 'tail' with manual credit notes.

Scenario:

e.g.
- supposing the go live is Apr 1, then
- Feb 1 invoice: for Jan as normal
- no invoicing on Mar 1, however.
- either domain gets paid for by PIA (payment in advance) OR goes into quarantine(NRP) on Apr 1

- Jan will be last set for debtors ledger, credit notes. last transfer 31 Jan.
- Feb 28: normal clear down and MSD
- Mar 09: Mail
- Mar 29: suspend
- Mar 31: new delete date
- Apr 06: original delete date
- so its 7 days shortened.
- deletion period shortened rather than suspend, as customers will tend to react to suspension

RISK: as customers will be given much advance notice of the transition date, the schedule needs to be held.
- mitigation: final user UAT complete T -1 month from go live. so:
- tech testing
- uat testing
- 1 month
- GO LIVE

Payments & Deposit Accounts

REQ#0112: Accounts team must be emailed for all receipts.

«Functional» Requirement:
This is currently the status quo, and needs to persist.

These emails function as double-checks in revenue assurance.

REQ#0113: The deposit account must be topped up from within CRS

«Functional» Requirement:
The deposit account must be topped up from within CRS. Registrars may do so directly by credit card, or indirectly by off-line means (send a cheque or CT to IEDR). Accounts staff will process the latter from within CRS.

REQ#0114: A visa debit card payment option could be available for direct registrants

«Functional» Requirement:
For Directs - investigate adding Visa debit cards (now that the laser offering has been withdrawn from the market). Very expensive compared to Visa credit card which is approximately 2.5%.

REQ#0120: There must be a direct correspondance between invoices and receipts.

«Constraint» Requirement:

Every receipt will correspond to an invoice, and an invoice will correctly tally with all associated receipts.

REQ#0122: A deposit account reservation mechanism is used for all ADP payments.

«Soln. Idea» Requirement:
A deposit a/c reservation mechanism is used for all ADP payments.

[document flow chart]

DEF: A 'reservation' on a Deposit account is the setting aside of a specific amount associated with a single transaction.

The 'available deposit a/c balance' is the actual deposit balance less the total current reservation amount against the deposit a/c.

The available deposit balance must never be less than zero. [There may be a potential requirement for an overdraft facility. TBD]

Reservations apply to deposit a/c only.

Rationale

If a deposit account reservation mechanism was not used, and:

- an invoice / receipt is generated for each ADP payment, especially for new registrations and transfers

then

- some registrars would receive an unacceptably high volume of invoices in a time period

REQ#0129: Deposit a/c balances must always be shown net of current reservations.

«Usability» Requirement:
When displaying deposit a/c balance, it must always be shown net of current reservations.

REQ#0123: In a batch payment against multiple domains, either payment succeeds for all, or the transaction msut fail for all.

«Functional» Requirement:
A batch payment is one made against multiple domains in one go. The IEDR must never choose what subset of domains get paid for if payment against all domains in the batch cannot be satisfied simultaneously.

Scenario:

- account:pay called for 10 domains, but only available funds for 6, so fails.
- account:pay then called for 5 domains

The second one succeeds immediately.

REQ#0130: Direct registrants receive one invoice per credit card transaction.

«Legal» Requirement:
Direct registrants receive one invoice per credit card transaction.

REQ#0131: Registrars must be able to use any medium for any transaction type at any time for any term.

«Purpose» Requirement:
Registrars can use any medium (ADP / PIA) for any transaction type at any time for any term.

REQ#0132: Directs must be able to use CC payment method for any transaction type at any time.

«Purpose» Requirement:
Directs can use CC payment method for any transaction type at any time for any term.

REQ#0133: Outcome of insufficient funds for registrations.

«Functional» Requirement:
When the financial check for domain registrations fail due to insufficient funds then:

- e-mail notification is sent to the Bill-C – just once (no further notifications).
- The ticket will wait in the queue for the customer to pay
- if no payment is received within 27 days, ticket to be deleted (as per current practice)

[The specifics of what triggers the events needs to be drawn out at this time.]

REQ#0134: Outcome of insufficient funds for transfer transactions.

«Functional» Requirement:

When the financial check for domain transfers fail due to insufficient funds then:

- e-mail notification is sent to the Bill-C – just once (no further notifications).
- The ticket will wait in the queue for the customer to pay
- if no payment is received within 27 days, ticket to be deleted (as per current practice)

REQ#0135: Outcome of insufficient funds for renewals.

«Functional» Requirement:
When the financial check for domain renewals (either in or prior to NRP) fail due to insufficient funds then:

- an appropriate notification is presented – just once (no further notifications), either by way of an error-message on the API or an on-screen error message for the NRC.

REQ#0137: For charities, CHY number is required on a new registration ticket.

«Constraint» Requirement:
This is to avoid compulsory payment.

If not initially entered, the CHY may be edited on the Ticket any time up to 'double passed' i.e. before being considered for "financial passed".

No refund following receipt, if CHY not provided.

REQ#0138: Allow for re-labelling of charities as non-billable post registration.

«Functional» Requirement:
To apply from year two onwards.

REQ#0139: WIPO will remain as a manual process.

«Functional» Requirement:
Will remain as a manual process for the foreseeable future.

Registration Services to manually flag the domain as being in WIPO arbitration when IEDR is notified of the ieDRP by WIPO.

Registrar action is prohibited as long as the domain is flagged as being in WIPO arbitration.

Transfers can only be undertaken by Registration Services. (In accordance with the WIPO's decision).

REQ#0140: Payment of renewal fees must be permitted for a domain in WIPO arbitration.

«Functional» Requirement:
Payment of renewal fees should be permitted for a domain in WIPO arbitration (and the renewal date rolled forward, in the normal way).

However, registrant would be separately advised (see IEDR web site) that there will be no refunds, in the event that the WIPO decision is negative for him.

VAT and Invoicing

REQ#0084: VAT rate changes must be accommodated in a controlled manner.

«Sup/Maint» Requirement:
There will be a mechanism to allow changes in VAT rates on specified dates to be affected.

Rationale

VAT rates can change and so the system needs to accommodate this in a controlled manner. As such, applicable VAT rates will have periods of validity.

Potential Implementation

The VAT Rate table would consist of the following:

- ID - Primary Key
- Category - The VAT Category (has values such as 'A', 'B', etc.). Every VAT charge applicable to a transaction is associated with a particular category. Different categories may have different rates.
- From date - The date from which (inclusively) the VAT rate applies to the category.
- Rate - The rate.

NOTE: It must be ensured that no inserts / updates / deletes ever affect existing transaction details.

[Potentially, the invoice XML can contain a copy of the VAT table record used at the time. Reservation table may also do this as double-check.]

REQ#0058: CRS must be primary source for financial data

«Constraint» Requirement:
Data will flow from CRS to G/L only, not the other way.
G/L can perform double-check on calculation and does exception report.

REQ#0108: Invoices delivered from CRS and G/L must match in form and content.

«Functional» Requirement:
Invoices delivered from CRS and G/L must match in form and content.

There will be a 'softcopy format' of an invoice (for example, a PDF) that is available to Registrars and the G/L system.

The softcopy document format needs to be the same in every respect. It is an issue if even the format alone changes.

Potential solution

Approach may be to generate softcopy one time from XML via XSLT and store. Both CRS and G/L would deliver this softcopy when necessary to the registrar.

REQ#0052: CRS must be the source of sequence numbers.

«Constraint» ***<u>Requirement</u>:***
Sequence numbers used, such as Invoice numbers, will be generated by CRS, and not from an external source, such as the external G/L system, as CRS is a real-time system, and such dependencies cause technical and usability issues (delays).

<u>REQ#0109: Invoice sequencing must be complete.</u>

«Functional» ***<u>Requirement</u>:***
Invoice sequencing must be complete and without gaps when imported into the G/L system. Therefore, it must be complete in the CRS system also, as CRS will be the source of invoice data.

<u>REQ#0034: There will be a clear delineation between accounting months when importing invoices into G/L.</u>

«Constraint» ***<u>Requirement</u>:***

As such there must be a clear point in the sequence of invoices where the run of invoices prior to the point are from the previous month and the run following the point belong to the current month.

One potential solution approach is for the CRS system to generate the primary invoice data in XML format and store it in date formatted folders, which can then be accessed by the G/L system.

<u>REQ#0125: VAT rate, if applicable, must be included in reservation amount.</u>

«Functional» ***<u>Requirement</u>:***
With regard to any potential deposit account reservation mechanism, VAT needs to be included in reservation amount, because the reservation is effectively 'taking the money' prior to generating the receipt and invoice.

It also follows, therefore, that the VAT rate to be applied invoicing needs to be known at the time of reservation.

A reference to the VAT rate table can be included in the table storing reservation details.

It can be assumed that any pending invoices against reservations will be run just prior to a VAT rate change. Therefore, the current VAT rate (at time of reservation) is used in reservation calculations.

<u>REQ#0127: VAT must be applied to invoice line items, not to subtotal.</u>

«Legal» ***<u>Requirement</u>:***
Given that VAT is used in reservation calculation per line item it must be likewise calculated when invoicing, otherwise discrepancies could occur between the two.

<u>REQ#0128: Domain pricing will always ensure two decimal places when VAT included.</u>

«Assumption» ***<u>Requirement</u>:***
Domain pricing will never be set by IEDR such that application of VAT rate will result in a third decimal place in cash value (fraction of a cent).

If this is violated, then discrepancies could arise between subtotals of reservations and invoice totals.

<u>REQ#0150: The system must conform to the Irish Revenue regulations concerning e-invoicing.</u>

«Legal» ***<u>Requirement</u>:***
If electronic format is the primary format for storing invoices, there are regulations set out by the Irish Revenue Commission which must be adhered to.

Broadly, they stipulate that if invoices are not printed primarily, then they must either:

a) be serviced by an EDI system
b) be supported by AES, an encryption standard which can be used to digitally sign electronic documents

REQ#0107: Authorised CRS users must be able to update the VAT exempt status variables.

«Functional» Requirement:
Authorised CRS users can update the VAT exempt status variables.
- not done in G/L system.

G/L Requirements

Note: These requirements pertain to the G/L system, and meeting them is technically outside the scope of the core BPR. However, they are included here for context and clarity. They should be considered as facts / assumptions, rather than as requirements.

ISSUE#0004: Creation of Debtor's ledgers in G/L system if they don't already exist.

«Open Issue» Issue:

Only affects Directs or Dep a/c.

- current process - the C batch program pre-creates any necessary accounts before processing the transactions (importing the transaction data).

- G/L would need to have this done

-- if we can't include the detail in the invoice and have G/L import do the account creation work (preferred, if not cost prohibitive), then we would need process on our side to perform this in advance.
-- depends on push / pull to G/L: if CRS initiates then the code should be on our side.

REQ#0031: Existing VISION accounting package to be replaced.

«Assumption» Requirement:
The accounting package will be replaced. The new system will interact with CRS.

REQ#0033: The flow of invoice data from CRS to G/L system can be controlled by IEDR Finance.

«Usability» Requirement:
The CRS will perform invoicing. The G/L will accept invoice data from CRS.

There will be some form of invoice batch run initiated by IEDR Finance.

REQ#0035: Invoice data transfer from CRS to G/L system will be XML.

«Constraint» Requirement:
This is an implementation constraint. XML (as opposed to CSV or other) allows greater flexibility in terms of data transformation and future proofing.

REQ#0047: G/L system must import financial data as-is.

«Constraint» Requirement:
The system must not re-calculate independently. This must be the case even if CRS were to calculate VAT incorrectly.

REQ#0045: PI Invoices will reside solely in G/L.

«Assumption» Requirement:
PI invoices are out-of-scope for BPR.

REQ#0110: Invoice discrepancies handled in G/L.

«Assumption» Requirement:

In cases where CRS and G/L differ on the details of an invoice, it will be handled as an exception by the G/L somehow.

REQ#0050: A discrepancy in an invoice received by th G/L system from CRS will not prevent processing of the batch.

«Usability» Requirement:
When batch processing invoices between CRS and the G/L, discrepancies in one item will not interfere with the processing of other unrelated items.

REQ#0116: An invoice must be imported into G/L system exactly once.

«Constraint» Requirement:

There will be a structured directory system used to deposit the invoice XML as it is generated. The structure will be:

<invoice_root>/<YYYYMMDD>/NEW
<invoice_root>/<YYYYMMDD>/ARCHIVE

Generated XML will be deposited in the appropriate /NEW directory.
The process that imports invoices to the G/L system will move them from the NEW to the ARCHIVE directory.

The <invoice_root> path should be configurable as a system environment variable. See REQ#100.

AUTH Codes for Transfers

REQ#0085: Auth Codes could be used for automating transfers of domains between Registrars.

«Functional» Requirement:
NOTE: This requirement needs further detail / scope. At present it is a 'nice-to-have'.

The system will support the use of AUTH codes for Registrar -> Registrar transfers. The following is the typical scenario:

- A domain holder (considered to be the Admin-C of the domain) that is a customer of a registrar (losing registrar) wishes to transfer the domain to a new registrar (gaining registrar).
- The holder requests from the IEDR an AUTH code for their domain as they wish to move
- The system generates a one-off time-limited AUTH code and emails it to the Admin-C
- The Admin-C can then supply the gaining registrar with the AUTH code.
- The gaining registrar can then supply the AUTH code along with the transfer request, thus validating the authorisation to move the domain.

Rationale

This will expedite transfers, and remove the need for Registration Services to manually check authorisation.

Miscellaneous Business Requirements

REQ#0026: Custom emails per email type.

«Usability» Requirement:
[This was in relation to the NRP discussion on 17 Nov. Specific detail required. - PD]

REQ#0087: Reports must be able to distinguish the method of payment used for an invoice (ADP/PIA).

«Constraint» Requirement:

Rationale: Finance require reports to be run based on the distinction between them. The current implementation encodes this information in the Receipt Order ID.

The payment method used to pay an invoice and the associated set of domain status updates must be recorded for reporting purposes.

- by domain update we mean whether the domain went from NRP to Active etc. Approach will be to include domain state at payment and any other necessary detail.

REQ#0095: Fast-track deletion for domains must be accommodated.

«Functional» Requirement:
Currently, Registration Services can place a domain straight into deleted state, usually upon request of the associated registrar.

This capability for authorised IEDR users needs to be implemented in CRS-WEB, and reflected in the DSM.

Because domains in the deleted state may be removed very soon after being placed in the deleted state, there may be a need to have an alternate state to account for a 'cooling-off' period. This would mitigate the risk of erroneously deleting a domain with severe consequences.

[Further clarification required.]

REQ#0115: There must be sufficient revenue assurance controls.

«Functional» Requirement:

[This needs to be made concrete.]
DC: Perhaps exception reports (report/e-mail generated when renewal dates rolled forward, without recording a cash receipt).
TBD - consider the overall control environment - the entirety of individual controls.
DT: a separate monitoring process may be appropriate.

Source: Meeting NASK Thu 8 Dec

REQ#0119: Minimal time lag between registration triple-pass and domain go-live.

«Purpose» Requirement:
There should be minimal / no time-lag between successful Triple-passing of a registration ticket and the creation of the domain.

Technical Requirements

The following areas of scope are included:

CRS-API - A description of the changes required to be made to the CRS-API component as a result of changed IEDR business processes.

CRS-WEB-DNSCHECK - New functionality to be part of the Core component which will perform technical validation of domain registration tickets by validating the details of the associated nameservers.

CRS-WEB-IE-Beo - Describes the changes required to CRS-WEB to enable the immediate checking of financial information and processing of payment at the earliest opportunity in the domain registration process.

CRS-WEB-PushQ - Describes the changes required to the CRS-WEB component necessary to implement the new Non-Renewal Process (NRP), mandated by the upcoming changes in IEDR billing policy.

In addition, a general Domain State Machine is described which underpins several of the areas of change.

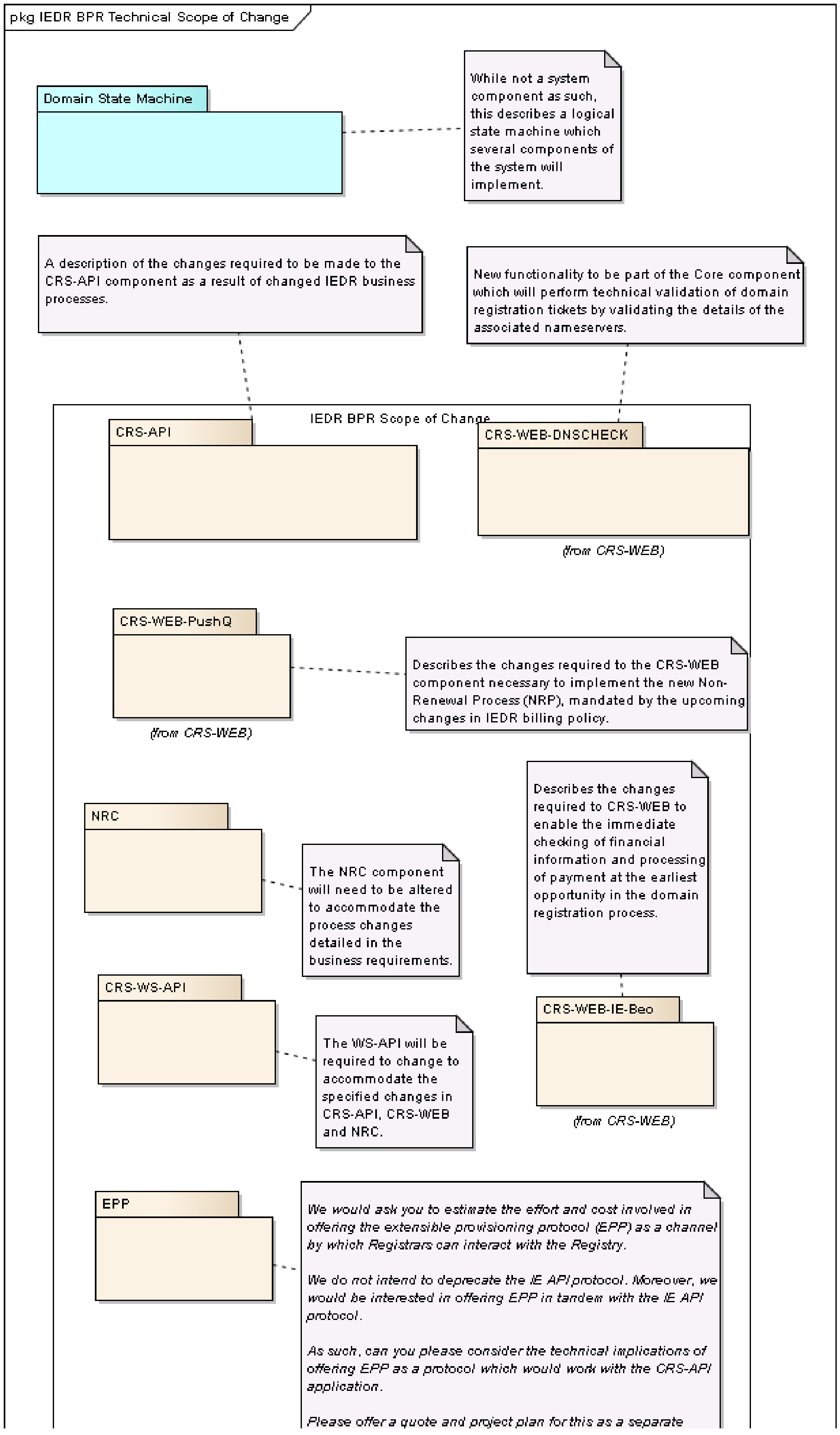

pkg IEDR BPR Technical Scope of Change
Domain State Machine
While not a system component as such, this describes a logical state machine which several components of the system will implement.
A description of the changes required to be made to the CRS-API component as a result of changed IEDR business processes.
New functionality to be part of the Core component which will perform technical validation of domain registration tickets by validating the details of the associated nameservers.
IEDR BPR Scope of Change
CRS-API
CRS-WEB-DNSCHECK
(from CRS-WEB)
CRS-WEB-PushQ
(from CRS-WEB)
Describes the changes required to the CRS-WEB component necessary to implement the new Non-Renewal Process (NRP), mandated by the upcoming changes in IEDR billing policy.
NRC
The NRC component will need to be altered to accommodate the process changes detailed in the business requirements.
Describes the changes required to CRS-WEB to enable the immediate checking of financial information and processing of payment at the earliest opportunity in the domain registration process.
CRS-WS-API
The WS-API will be required to change to accommodate the specified changes in CRS-API, CRS-WEB and NRC.
CRS-WEB-IE-Beo
(from CRS-WEB)
EPP
We would ask you to estimate the effort and cost involved in offering the extensible provisioning protocol (EPP) as a channel by which Registrars can interact with the Registry.
We do not intend to deprecate the IE API protocol. Moreover, we would be interested in offering EPP in tandem with the IE API protocol.
As such, can you please consider the technical implications of offering EPP as a protocol which would work with the CRS-API application.
Please offer a quote and project plan for this as a separate

Figure 1 : IEDR BPR Technical Scope of Change

Domain State Machine

As part of the specification of the IEDR business processes affected by the scope of change, a Domain State Machine (DSM) is defined which specifies:

domain states: the set of billing states that a domain object can be in during its lifetime, from creation to ultimate removal from the system.

events: the set of system events that can affect the state of a domain (i.e., change the state of the domain). These events are triggered by various components of the system, either as part of scheduled processes, or actor actions.

transitions: the set of valid next states that each state results in given a particular event. Each transition specifies a current state, an event, a next state, and zero or more actions to be performed when the transition occurs.

actions: the set of system actions that can be associated with state transitions.

States

The full set of states is specified in the Domain State Machine Specification Spreadsheet document. [This is currently in draft.] The set of states has been derived by the following process:

Each factor (parameter type) that needs to be represented by the state machine (because the business / billing logic is dependant upon it) and the possible set of values of each is identified. The parameters and their value sets are:

Domain Holder Type: This indicates the type of domain holder entity that currently holds the domain.
Values: [B=Billable, C=Charity, I=IEDR, N=Non-Billable]

Renewal Type: This indicates the renewal option currently placed by the holder on the domain.
Values: [N=No auto renew, R=Renew Once, A=Auto renew]

WIPO Dispute: This indicates whether the domain is currently under WIPO dispute arbitration.
Values: [Y = Yes, in WIPO dispute, N = No, not in WIPO dispute]

Customer Type: This indicates the type of IEDR customer that registered the domain.
Values: [R=Registrar, D=Direct]

NRP Status: This indicates where in the new Non-Renewal Process (NRP) the domain currently is.
Values: [A = Active, IM = Involuntary Mailed, IS = Involuntary Suspended, VM = Voluntary Mailed, VS = Voluntary Suspended, D = Deleted, P = Post-Transaction Audit, T = Transaction Failed.]

All possible permutations of the values for each parameter are enumerated.

Each permutation represents a potential state in the DSM, and is given a unique ID.

Many states are not feasible by the business rules, and so are eliminated from the DSM.

Some states are equivalent to each other, and so are collapsed into a single state.

The above process results in a state machine with [currently] 62 states.

A DSM Transition is associated with one current state, one next state, an event, and a sequence of actions. Note: this is a logical relationship diagram, not a DB schema portion.

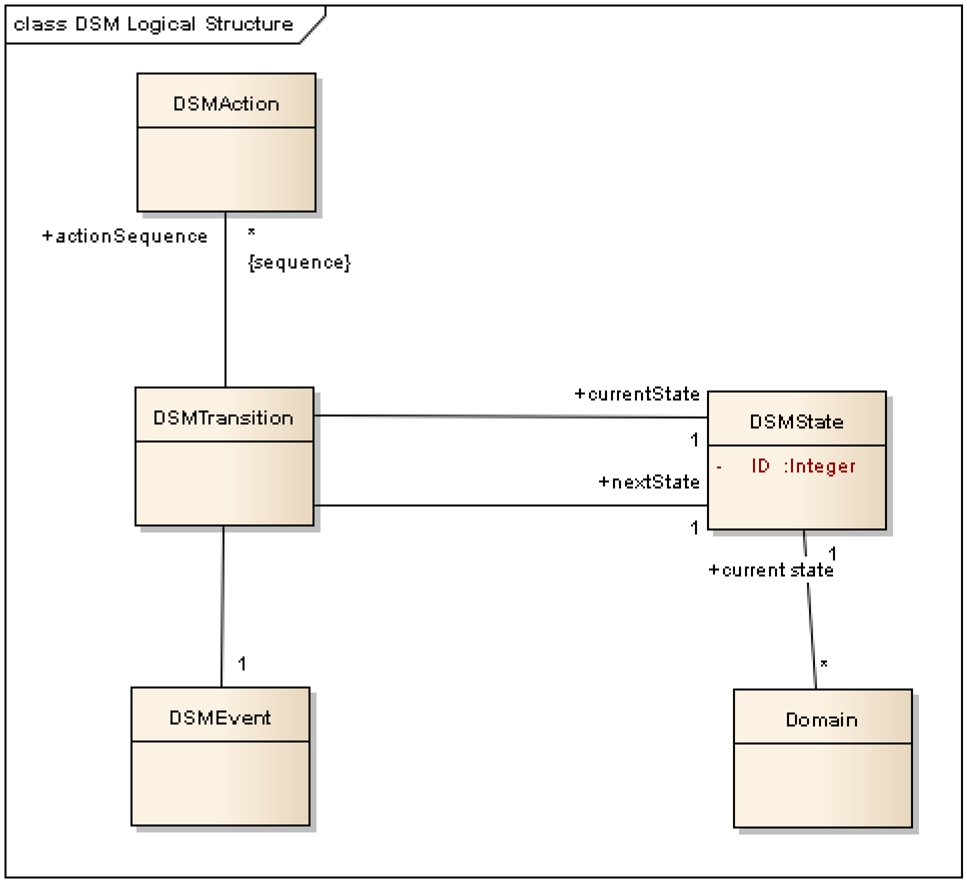


Figure 2 : DSM Logical Structure

Domain State Machine::DSMState
public Class:

REQ#0063: The system must implement the Domain State Machine model.
«Constraint» Requirement:
Proposed Implementation Approach

The specification of the business events and actions affecting the lifecycle of a domain as a formal state machine aids overall clarity and identification of edge cases which need to be addressed. Implementing the logic as a state machine has the additional benefit of future-proofing the system, to some extent, against changes in business rules and requirements without needing to make modifications to code and deploying full revisions of components.

To that end, one recommended approach to implementing the DSM logic is as follows:

States

Implement states as a DSM State Table, with one row per state, and the unique ID ('state number') being the primary reference. In addition, the table should include one field for each of the five parameters (Domain Holder Type, Renewal Type etc.) as shown in the DSM Specification. These fields are not intended to be referenced anywhere in code, but would greatly assist understanding the meaning of states, as well as debugging efforts.

It should be noted from the DSM Spec 'States' tab that the ID numbers are not sequential. This is because infeasible and redundant states have been removed. However, it is recommended that the state numbers are maintained as is, and not altered. This is to avoid confusion when cross-referencing from the specification to the implementation.

Events

The set of events that the DSM responds to is enumerated in a DSM Event table, and uniquely identified.

Actions

The set of actions that the DSM are caused by a transition is enumerated in a DSM Action table.

Transitions

The full set of transitions in the DSM are contained in a DSM Transition table. Each record of this table will reference the current state, an event, next state, and a sequence of actions.

Execution Model

When a business process responsible for one or more events in the DSM is executing, it will find all domains currently in a state that is a current state for any transition initiated by the event. Then, for each domain, it will select and execute the actions associated with the transition to the next state, and then update the domain to be in the next state of the transition.

REQ#0062: Constraints on Domain States

«Assumption» ***Requirement*:**
The following assumptions are made to reduce the number of potential domain states in the DSM:

- Charities don't require renewal options as they are not charged, so Renewal Type is never 'R' or 'A'.
- IEDR domains don't have customers, billing or WIPO issues, so there is a single state where Holder Type = 'I' and the other values are irrelevant.
- Involuntary NRP doesn't apply to Non-Billables, so there are no states where NRP Status is one of IM, IS when Holder Type = 'N'.
- No Involuntary NPR for Charities, so no states where Holder Type = 'C' and NRP status is one of IM or IS.
- Only one DELETED state is required.
- Renewal Type R and A applicable to registrars only.
- Renewal Types and Involuntary NPR don't apply to Non-Billable Holders.
- WIPO - no NPR status changes: IEDR cannot alter domain status / bill until WIPO resolved. Note that renewal payments can be received for domains in WIPO. See REQ#140.
- GIBO only applies to registrars (Post registration audits, Transaction failed).

[Q: If payments can be made against domains in WIPO - can they also go into involuntary NRP while in WIPO? If so, what happens if no payment received - can they ultimately be deleted?]

REQ#0088: The history of domain states for a domain must be recorded.

«Functional» ***Requirement*:**
The current status quo of recording the full domain record history is also to be maintained. The recording of domain state could be part of this full record.

This will greatly assist issue diagnosis and resolution, both in terms of system debugging and customer issue resolution.

REQ#0099: DSM transitions must only complete on success of all associated actions.

«Constraint» Requirement:
Description

If a DSM transition is effected and has associated actions, the process affecting the transition must ensure that the state is not updated to the next state until all the actions are confirmed successfully completed. In other words, the updating of the domain's current state is the very last thing to happen, with the exception of updating the domain state history.

Rationale

Should there be exceptional interruption of processing (e.g., database failure, or similar), the restarting of the DSM will not result in domains erroneously being in a new state without the successful completion of all DSM actions.

Implementation Note

SQL Transactions should be used appropriately.

REQ#0089: Explicit publish flag for domain states should be incorporated.

«Usability» Requirement:
Description

The implementation of the DSM will include a flag for each state which will solely control whether domains in that state are included in the IE zone file on a Rebuild.

Rationale

This is to give precise control over this critical business function. It will also make the Rebuild logic clear regarding the selection of domains to publish.

REQ#0152: DSM exceptions should be recorded for diagnostic purposes.

«Sup/Maint» Requirement:
If a DSM event occurs with reference to a particular domain in a particular state, and the DSM does not have any transition defined for that state / event, then an exception should be recorded for diagnostic purposes.

Architecture

REQ#0003: All system changes should progress the architecture of the system towards a coherent MVC design.

«Constraint» Requirement:
At present, the core Phoenix database is accessed via several disparate systems and processes, including batch process cron jobs. This situation is less than ideal as it increases operational risk of data integrity loss or error, and limits the long-term maintainability of the system.

It is an overarching technical consideration to progress the design of the system to a clean MVC design, with the Phoenix database accessed directly by the core alone, with the CRS-WS-API layer alone accessing the core, and all other system components needing to interact with the Phoenix database accessing the CRS-WS-API.

The rationale for this is to mitigate risk to the business of unintended data integrity violation or other disruption to the normal flow of business processes that may result from unnecessarily complex implementations. There may also be reduced long-term maintenance costs associated with the 'clean' architecture.

While it may not be feasible to implement the ideal architecture in any single development project, any development which alters the lines of communication / interaction with the Phoenix database should, at the very least, not move away from this goal. For example, no components should be changed in or added to the system which result in additional direct Phoenix database connections that bypass the CRS-WS-API layer API.

REQ#0144: Unit testing should be done at the Core / WS-API level.

«Sup/Maint» Requirement:
Description

Unit tests (JUnit) will be performed at the Core / WS-API level. This will be in addition to any other existing test suites (such as the Selenium tests at the browser client level).

Rationale

The ideal architecture will have the DB and core accessed via the WS-API only. All other components will interact with the WS-API. By ensuring that there is a unit test suite at the Core / WS-API level, we can ensure that testing done at the 'outer' layers need only exercise code and functionality at those layers.

REQ#0145: Deployment model should permit isolated updates of the core.

«Sup/Maint» Requirement:
Description

This means the ability to take a core release in isolation, with dependant components separately thereafter. So the sequence might be:

- Deploy new core (and WS-API).
- Deploy any updated components (NRC, CRS-WEB, etc.) if necessary.

Rationale

Changes to the model / schema can be made in isolation of other components and reduce risk of customer issues.

REQ#0146: The architecture could provide the ability to balance the load over several core instances.

«Sup/Maint» <u>Requirement</u>:
Rationale

This will provide for future scalability.

<u>REQ#0100: System-wide environment configuration should be configurable by a centralised manageable mechanism.</u>

«Usability» <u>Requirement</u>:
Description

There is a clean and effective mechanism whereby the principal system environment configuration details can be maintained, and where separate configuration sets for production versus test environments can be separated. The environment details would include:

- DB server connection details
- RealEx payment server details
- any significant limits (maximum values etc.) which may need to be configured
- (any other significant configuration details which could differ between production and non-production environments.)

Rationale

This will enhance usability from a system administration perspective, and also reduce risk of erroneous (e.g., test) environment variables being used in the production system.

Suggested Implementation Approach

One approach would be to have a general configuration file containing all settings for a component, which would be deployed when a new version of the component is deployed, and in addition, a second environment-specific file which could override specific configuration options as necessary. This second configuration file would not be part of a deployment to production.

Also, given REQ#145, it may be necessary to have separate configurations for the Core and each separately deployable component.

CRS-API

Introduction

This section outlines the high-level design considerations for the CRS-API component. The scope of work is part of the overall scope of work of the IEDR BPR Development Project, and all details should be taken in that context. The goal of the scope of change for the CRS-API is to make the changes necessary to accommodate the proposed new billing process and policies. This is expected to result in a simplification of both the business processes and the API itself.

Assumptions

It is assumed that there will be a concerted deployment initiative ala big-bang. That is to say that we do not intend to offer backward compatibility with the older billing logic.

Further to the stated requirement of a protocol version number increment, it is assumed that we do not offer backward compatibility for 1.2 as this would make the deployment and development overtly complex and costly.

The CRS-API will form part of the implementation of the Domain State Machine. Specifically, it will trigger the following events:

[need to conclusively determine the list of DSM events triggered by API...]

Principal Requirements

We are proposing that we significantly simply and reduce the range of payment commands which we offer to Registrars which are paying for domains in various states. This reduction in command availability is logically in parallel with the simplification proposed by the BPR project owners. That is to say, that the BPR initiative intends by its design, to withdraw the option of variable time selection for payment of new registrations and transfers from a Registrars' discretionary implementation. Rather, the BPR mandate of payment for both new registrations and transfers at the point of Hostmaster acceptance significantly simplifies the necessity for status quo payment types.

The introduction of the above billing principle means that the remainder of billable domains can only be renewals which are either being renewed before their renewal date or after their renewal date. Consequently, it is vastly more simple for us to rationalise the payment commands available via the IE API such that we only offer ONE command. This is a hugely significant simplification of dot IE billing logic from an IE API perspective given that one command replaces FOUR deprecated commands which had many sub-types of invoice/payment states (currRenReg, futRen, Xfer, XferPIA).

REQ#0064: The CRS-API must implement the appropriate DSM events.

«Constraint» ***Requirement:***
The CRS-API is responsible for affecting a subset of the transitions in the DSM. The following events are to be handled by the CRS-API:

[complete list to be determined]

REQ#0065: The CRS-API should deprecate any redundant commands.

«Sup/Maint» ***Requirement:***
Description

1. The account:payOffline command is deprecated.
2. The account:payOnline command is deprecated.
3. The account:payFromDeposit command is deprecated.

4. The account:msdReActivation command is deprecated.

Rationale

The new IEDR billing process will effectively make these commands redundant.

REQ#0066: A new account:pay command must be implemented.

«Soln. Idea» Requirement:
Description

An entity for the CVV value on card payments will be included for future-proofing purposes, although at this time, it will not be used.
[TODO: modify example to include CVV]

An account:pay XML should work as follows for a credit card:

```
<ieapi xsi:schemaLocation="http://www.domainregistry.ie/ieapi-1.3  ieapi-1.3.xsd">
<command>
<pay>
<account:pay xsi:schemaLocation="http://www.domainregistry.ie/ieapi-account-1.3     ieapi-account1.3.xsd" test="false">
<account:domain>example1.ie</account:domain>
<account:domain>example2.ie</account:domain>

<account:method>
<account:card>
<account:cardHolderName>JohnDoe</account:cardHolderName>
<account:cardNumber>12341234123412</account:cardNumber>
<account:expiryDate>2010-01</account:expiryDate>
<account:cardType>VISA</account:cardType>
</account:card>
</account:method>

<period>10</period>

</account:pay>
</pay>
</command>
</ieapi>
```

An account:pay XML should work as follows for a deposit account:

```
<ieapi xsi:schemaLocation="http://www.domainregistry.ie/ieapi-1.3  ieapi-1.3.xsd">
<command>
<pay>
<account:pay xsi:schemaLocation="http://www.domainregistry.ie/ieapi-account-1.3     ieapi-account1.3.xsd" test="false">
<account:domain>example1.ie</account:domain>
<account:domain>example2.ie</account:domain>

<account:method>
   <deposit/>
</account:method>
<period>10</period>
</account:pay>
```

```
</pay>
</command>
</ieapi>
```

REQ#0067: There must be a 'test mode' for the new payment command.

«Functional» Requirement:
Description

An optional element called "test" may be included. The test element may have a value of "true" or "false". False implies that a test will not be run, i.e. the payment command will be fully processed. Whereas true implies that the Registrar wishes to run the payment command in test mode.

REQ#0068: The account:pay command must fail on insufficient funds.

«Functional» Requirement:
If there are insufficient available (net of reservations) funds, the usual constraints must persist. i.e. the command should not complete and an appropriate message should be returned to the Registrar client system.

REQ#0069: Determination of renewal versus registration.

«Functional» Requirement:
A domain which a Registrar is attempting to pay for, which has no corresponding ticket and which has a renewal date greater or equal to today&rsquo;s date is assumed to be a renewal. Accordingly, the renewal date must be incremented by the years detailed in the account:pay period element value, upon successful financial processing using either of the payment methods permissable; deposit or card.

REQ#0070: Determination of renewal period.

«Assumption» Requirement:
An account:pay command must have an optional period value which implies multi-year-renewal. If the optional period element is omitted, a period of 1 year is assumed.

REQ#0071: Calculation of sufficient funds.

«Functional» Requirement:
Sufficient funds are calculated, for each domain, based upon the period selected or assumed, which corresponds to the product renewal fees as defined in the phoenixdb.Product table. The subtotal is calculated for the batch of domains.

REQ#0072: Calculation of VAT.

«Functional» Requirement:
VAT should be calculated as a percentage of the subtotal, depending on the VAT status of the Registrar. VAT logic is not changing and could be replicated from the status quo CRS-API billing logic.

[PD: This needs to be reviewed as it may now be contradicted by superceding requirements.]

REQ#0073: History records must be recorded.

«Functional» Requirement:
History must be recorded as per the status quo CRS-API approach.

REQ#0074: Receipts must have an auto incremented ID value.

«Functional» <u>Requirement</u>:
Receipts should be created as per the CRS-API approach with the exception of the introduction of a &ldquo;id&rdquo; value within the Receipts database table. It is proposed that we introduce an integer auto_increment receipt id value, which will allow for multiple payments per domain at any time. This would allow a registrar to renew a domain as many times as they like at any time, without causing duplicate receipt errors as would arise with the status quo table structure.

REQ#0086: The API must support placing domains voluntarily into NRP.

«Functional» <u>Requirement</u>:
The API will need to accommodate the action of a customer attempt to place a domain into voluntary NRP.

REQ#0153: The account:query API command will be redefined.

«Functional» <u>Requirement</u>:
The existing account:query command is used to retrieve sets of domains in various states relative to the MSD process. As the MSD process is to be replaced, the account:query command will need to be redefined to accommodate.

[TENTATIVE]

The assumptions are that registrars will want to be able to:

determine the set of domains in NRP
determine the set of domains will have renewal dates upcoming

To this end, the new account:query command will have two forms:

- an NRP form which will allow them to query what domains are currently in NRP. The response should allow them to determine, for each domain, what and when the next NRP stage is (Suspended, deleted).
- a renewal date form which will take a 'start' and 'end' date and return all domains whose renewal date is in that range.

[The details of this need to be made concrete, and the XML structure determined.]

REQ#0154: The domain:status entity must conform to the DSM.

«Functional» <u>Requirement</u>:

- domains currently active | suspended | deleted
- this will actually remain the case, so no fundamental change
- however, since this domain state is encoded in the DSM state, the DSM state table could have a lookup field for each state to determine the return value (as per 'publish')
- any reference to MSD needs to be removed. NRP is the new process.

REQ#0155: The domain:billingStatus entity must conform to the DSM.

«Functional» <u>Requirement</u>:

- currently values are billable | notBillable | MSD
- will need to refer to NRP instead of MSD
- as per domain:status - there will be a lookup field in DSM state table, which will define the response for any given domain state.

REQ#0156: domain:renew and domain:autorenew must trigger the associated DSM event.

«Functional» Requirement:
[Dependant on DSM specification.]

REQ#0157: domain:msd must be replaced with domain:nrp

«Functional» Requirement:
This will trigger the voluntary NRP DSM event.

REQ#0158: The account:checkDeposit must return the available deposit balance.

«Functional» Requirement:
Using the proposed reservation system on deposit accounts, this must return the AVAILABLE (ACTUAL - RESERVED) balance.

REQ#0159: Updates to Reason Codes.

«Functional» Requirement:
Codes to be added:
[TBD]

Codes to be deprecated:
[TBD]

Codes that have altered meaning:
[TBD]

====================
102 - obsolete: no locked domains
204 / 205 - repurposed for NRP?
206 - obsolete, no unpaid invoices
266 - changed (probably) - can renew in advance of renewal date. used when domain is 'DELETED'
268 - ?
810 - may be used in different circumstances (e.g., if domain is non-billable, charity...)
817 - may be used in different circumstances (e.g., if deleted)
819 - changed?
820 - obsolete?

CRS-WEB

CRS-WEB-DNSCHECK

The CRS-WEB-DNSCHECK (DNSCheck) scope of work is part of the overall scope of work for the IEDR BPR Development project. The primary objective is to re-implement the existing DNS Check function, which is not part of CRS-WEB and which validates the nameserver details of domain registration tickets, to bring it within CRS-WEB and in line with the MVC architectural design. In addition, some usability enhancements will be made for the benefit of Registrars and IEDR users.

The existing system DNS check process confirms whether specific nameservers, whose details have been supplied by customers, have been correctly configured for specific domains.

If the nameserver details supplied do not validate, the Customer is notified and the Ticket is not progressed to the next stage, which is the check of financials.

The configuration of nameservers is a customer responsibility and outside the remit of the IEDR. However, in order for domains to be registered, the nameservers must be validated when processing the associated Ticket.

The scope of change for this process is to re-implement the logic as part of the Core, while enhancing and refining some of the associated functionality.

There are two variations of DNS check at present, each triggered by a cron job on the www-deg server. They are
Full DNS check and Partial DNS check.

This diagram shows the activity flow for the current full DNS check. The main sequence of events is:

The cron job for the full DNS check on www-deg is initiated. The full DNS check is currently scheduled to run 30 minutes prior to each Rebuild of the .ie zone file but is not in fact part of the Rebuild process. The Rebuild process is currently performed at 10:00, 12:00, 15:00, 17:00 and 22:00.
A log file is created for the execution of the job.

A query is run against Phoenix which selects all tickets which have:
- a Tech Status of New, Stalled or Renew, as these are to be checked
- an Admin Status of Passed, as tickets must be Admin-PASSED before being DNS checked
- a Ticket Type NOT of type Deletion (D), as these Tickets are no longer relevant in the system

For each such ticket:
The IP (if any) and DNS Name for the nameservers are selected.
If there is an IP supplied, the nameserver and IP are passed as parameters to the ckdns script. This is done when the nameserver is in bailiwick. If an IP is not supplied for the nameserver, the nameserver and domain name are passed as parameters to the ckdns script. This script will determine whether the nameserver is configured for the domain, and stream output accordingly.

If any of the checks for the nameservers on the ticket fail, the ticket Tech Status is set to STALLED, and a notification email is sent to the Tech-C for the ticket. The appropriate Ticket Failure Code is updated for the relevant field for the failure. For example, if the IP check failed for the second nameserver entry, then DNSIP2_Fail_Cd would be updated with the correct code.

If all checks on nameservers for the ticket pass, then the tech status of the Ticket is set to PASSED. At this point, as part of the CRS-WEB-IE-Beo scope of work, financial checks for the ticket can then proceed.

Once all tickets are processed, the log file is emailed to asd@domainregistry.ie.

Note: at all points where the Ticket record is updated, the TicketHist table is appended.

A partial DNS check is similar with the following differences:

- It is run every 15 minutes and every 45 minutes past the hour from 09:00 to 21:00.
- For the partial check, the DNS check program selects all Tickets which are Admin PASSED but have never had a DNS check performed on them. The partial DNS check will not be required for the new scope.

Motivation for Change

There are several issues associated with this process as it currently stands:

1) There is unnecessary lag between the time a Ticket is Admin-PASSED to the time the DNScheck is performed. This can result in needless delay in notification to Customers that there is an issue with a nameserver.

2) When a DNS check fails for a particular nameserver, the Customer will receive a notification email. This can result in a notification email for each rebuild (if the check keeps failing). Registrars have expressed interest in being able to configure the frequency at which they receive the notification emails. The system does not currently support this.

3) The DNS check system interacts with Phoenix directly, and is implemented outside of the Core. This violates the MVC architecture, which is an overarching technical objective for the IEDR.

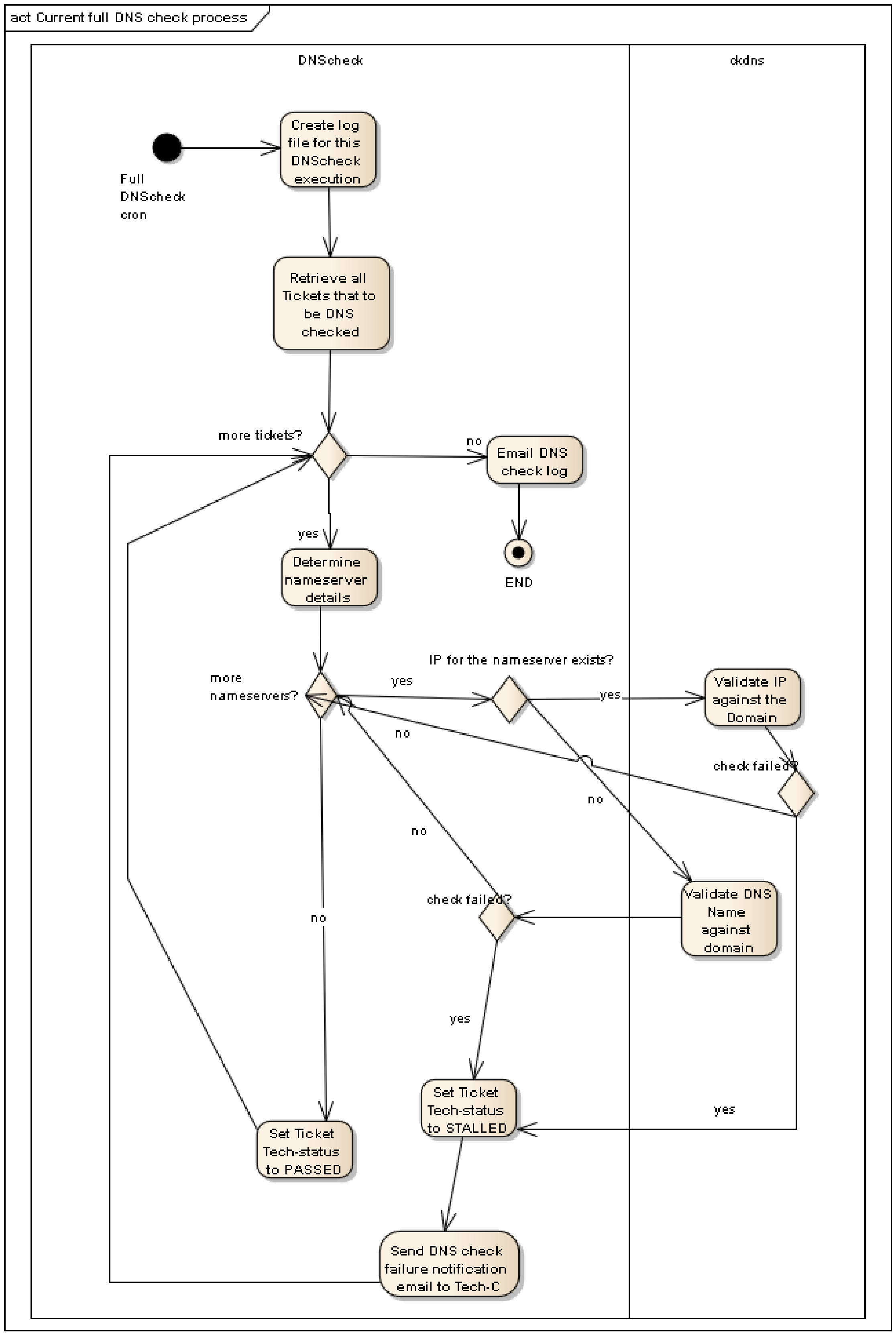

act Current full DNS check process
DNScheck
ckdns
Full DNScheck cron
Create log file for this DNScheck execution
Retrieve all Tickets that to be DNS checked
more tickets?
no
Email DNS check log
END
yes
Determine nameserver details
more nameservers?
yes
IP for the nameserver exists?
yes
Validate IP against the Domain
check failed?
no
no
no
Validate DNS Name against domain
check failed?
no
yes
Set Ticket Tech-status to STALLED
yes
Set Ticket Tech-status to PASSED
Send DNS check failure notification email to Tech-C

Figure 3 : Current full DNS check process

REQ#0163: Triggers for the DNS check process.

«Functional» Requirement:
The following events must trigger the DNS check:

1. When a registration ticket is ADMIN-PASSED, the DNS check will immediately be invoked on the ticket.
2. When a modification ticket is raised to change either the ADMIN-C or TECH-C on the domain, and the modification ticket has just been ADMIN-PASSED.
3. When the full DNS check process is scheduled to run, it will be performed on all tickets that are ADMIN-PASSED but not TECH-PASSED.
4. When a privileged IEDR user specifically invokes it via the CRS-WEB interface.

REQ#0009: The DNS check logic must be invoked from the core.

«Constraint» Requirement:
This is in keeping with the MVC architectural principle.

The existing ckdns Perl script which performs the check will be invoked from the core as required.

REQ#0053: Full DNS check batch runs must produce a log file.

«Functional» Requirement:
Each execution of the full DNS check batch job will produce a log file which will log the following information:

- Date & time of execution
- Ticket failures, and the specific details of the failure.

REQ#0090: The DNS check log must be mailed on error of execution only.

«Functional» Requirement:
If there is an exceptional error during execution of the Full DNS check process, the log file should be emailed to a configurable email address, initially configured as dnscheck@domainregistry.ie.

REQ#0055: Privileged IEDR users must be able to configure the schedule of the full DNS check.

«Usability» Requirement:
Privileged IEDR users can configure the schedule of the full DNS check.

REQ#0007: Frequency of DNS Check failure notification emails to registrars should be configurable.

«Functional» Requirement:
Registrars can configure the frequency of DNS check failure notification emails and their target email addresses.

The full DNS check process is performed several times daily. At present, if a domain nameserver fails this check, an email is sent to the associated Tech-C each time.

Registrars have expressed a desire to manage the frequency of these mails, for example, by receiving a single DNS failure notification email once per day, and also to specify particular email addresses for each type of email.

REQ#0056: Registrar email notification from DNS check schedule should be separated.

«Soln. Idea» ***<u>Requirement</u>:***
This is dependent on REQ#7.

The schedule, configured by each individual registrar, on which they receive DNS check failure notification emails, must not be tied to the separate Full DNS check schedule.

One potential approach to this would be to accumulate full DNS check failure notifications for a given registrar in a queue, and then to clear the queue when the registrar has specified that they receive the notifications. This is open for discussion.

REQ#0011: A DNS check must be triggered for a domain when a Registrar changes the associated Admin-C or Tech-C.

«Functional» ***<u>Requirement</u>:***
If the Admin-C or Tech-C is changed by a Registrar then a modification ticket is raised. If this ticked is subsequently Admin-Passed then the DNS check will be performed on the ticket.

REQ#0004: The DNS check for a domain must be performed as soon as a ticket is admin-passed.

«Functional» ***<u>Requirement</u>:***
By having the DNS check performed on a ticket as soon as it is Admin-PASSED, the average duration between registration and go-live for a domain is expected to be reduced.

REQ#0006: Priviledged IEDR users must be able to manually force DNS check on selected domains via CRS.

«Functional» ***<u>Requirement</u>:***
It may be the case that IEDR users will need to force the DNS check to be performed on selected domains outside of the Registration / Full DNS check processes. Privileged IEDR users will therefore need an interface mechanism to select domains and then force the DNS check process to execute on the selected domains.

CRS-WEB-IE-Beo

Introduction

The CRS-WEB-IE-Beo (IE-Beo) scope of work is part of the overall scope of work of the IEDR BPR Development Project. The goal of IE-Beo is to optimise the Domain Registration Process.

Currently, when a Registrar registers a domain, the associated ticked is created and checked by Registration Services. Once Registration Services have validated the request from an IEDR .ie Domain Policy perspective, the ticket's Admin status is PASSED and the ticket is then queued for technical validation of the nameserver details. This stage currently occurs in a batch DNS check process 30 minutes prior to each Rebuild job. The scope CRS-WEB-DNSCHECK addresses changes to this process separately. Once the proposed domain's technical details are validated by this process, the Domain object is then created in the system during the next Rebuild job.

Invoices for new registrations are then issued on the last calendar day of the month of registration, with 30 days credit terms offered. Registrars can pay off-line, online or from their deposit accounts, and the renewal date is then rolled forward by one year. After 30 days, on the last calendar day of the month, domains on unpaid invoices follow the Cleardown Process.

The scope of IE-Beo is to implement the re-engineering of this process so that the financial checks are performed as soon as possible, following Admin and Tech validation of the ticket, and the domain goes live at the earliest possible time, assuming no failures prevent it doing so.

REQ#0032: CRS must generate invoices.

«Functional» Requirement:
CRS will be the prime generator of Invoices. Invoice data will be transferred over to the G/L system.

REQ#0043: Deposit accounts must be maintained in CRS.

«Constraint» Requirement:
Deposit accounts will be maintained in CRS and should match the G/L.

REQ#0044: Corrections to deposit accounts must be made in CRS and reflected in th G/L system.

«Functional» Requirement:
Rationale

CRS will be the 'source of truth' for financial data, so the corrections will need to be made on the CRS side and imported to the G/L system.

REQ#0076: The financial checks on a registration ticket must be performed after being Admin / Tech passed, and prior to the creation of the domain.

«Functional» Requirement:
The financial checks are performed as soon as possible, following Admin and Tech validation of the ticket.

The process will be:

- Registrar issues the domain:create command and a registration ticket is created.
- Registration Services vet the ticket and set the ticket Admin Status to PASSED.
- DNS Check process will validate the nameservers supplied by the Registrar for the domain, and then set the ticket

Tech Status to PASSED.
- The (proposed) IE-Beo process will then perform the necessary financial checks and operations for the request, resulting in the Registrar's deposit A/c having the appropriate reservations placed against it and the available amount reduced by the appropriate amount
- The ticket history is created, the ticket removed, and the domain object is then created in the system.
- When Finance perform the Invoice run via CRS-WEB, all the reservations pending against the deposit account are accumulated and 'cleared' - and a single receipt / invoice pair created in the system against them.
- The invoice will detail each of the reservations (as line items).

The benefits of this proposal are:

It will accommodate the new IEDR Billing Process.
It will alleviate the need for manual intervention by IEDR Finance Team members.
Domains are only created after all checks, including financial, are successfully completed.

Financial Checks

At the point in the above process where the financial aspects are to be addressed, the sequence of events is:

- Check that a deposit A/c exists for the Bill-C of the Ticket.
- Check that there are sufficient available funds in the deposit A/c. (actual net of reservation). Include calculation of VAT (if applicable) as normal and compare with closing balance in the Deposit A/c.
- Create the reservation
- Update the renewal date.

REQ#0083: The system must accommodate GIBO registrations.

«Functional» <u>Requirement</u>:

As part of the GIBO (Getting Irish Businesses Online) initiative, some registrars are permitted to register new domains and have them go-live without being ADMIN-PASSED or TECH-PASSED at that time.

The current process is:

- Registrar issues the domain:autoCreate command
- No registration ticket is created. Rather, the domain object is created immediately.
- Thereafter, registration services validate the domain details, and subsequently payment is processed.

[The details of how GIBO will be handled in the new business processes is TBD]

REQ#0160: Triggers of financial checks on regsitration tickets

«Functional» <u>Requirement</u>:
For registration tickets that are ADMIN-PASSED, TECH-PASSED but not FINANCIAL-PASSED, the following will trigger the financial checks on the ticked:

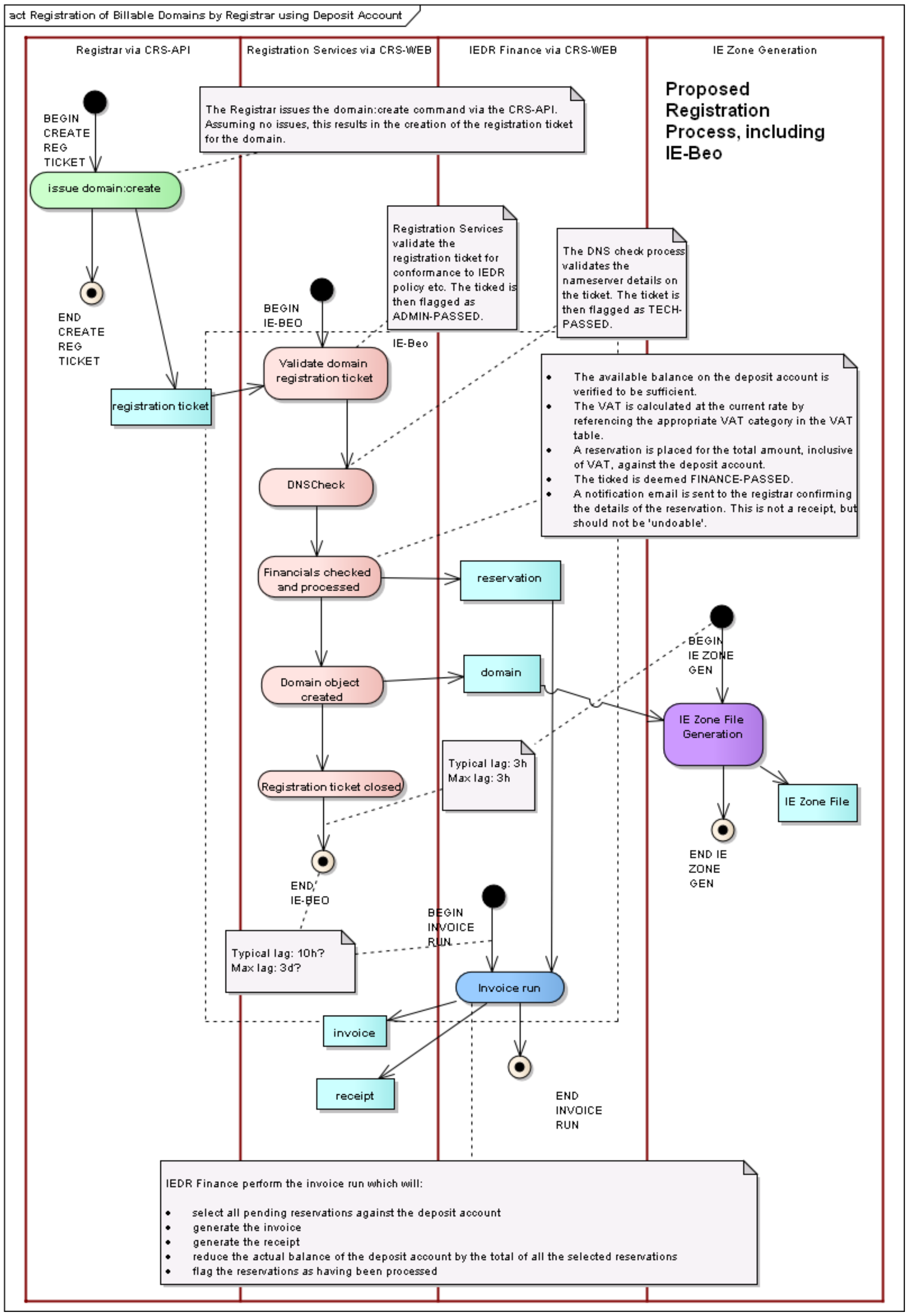
1. The DNS Check has just succeeded (TECH-PASSED) and the ticket has not yet been financially checked. The financial check will be attempted immediately.

2. The Registrar has topped up the deposit account. This will trigger the re-try of financial checks against any tickets that are financially failed.

3. A scheduled process is run which triggers the financial check on all such tickets. The schedule will be configurable by privileged IEDR personnel.

Registration using Deposit Account

The sequence of events that occurs when a Registrar registers a domain via the API for a billable domain using a deposit account without issues.



The Registrar issues the domain:create command via the CRS-API. Assuming no issues, this results in the creation of the registration ticket for the domain.

Registration Services validate the registration ticket for conformance to IEDR policy etc. The ticked is then flagged as ADMIN-PASSED.

The DNS check process validates the nameserver details on the ticket. The ticket is then flagged as TECH-PASSED.

- The available balance on the deposit account is verified to be sufficient.
- The VAT is calculated at the current rate by referencing the appropriate VAT category in the VAT table.
- A reservation is placed for the total amount, inclusive of VAT, against the deposit account.
- The ticked is deemed FINANCE-PASSED.
- A notification email is sent to the registrar confirming the details of the reservation. This is not a receipt, but should not be 'undoable'.

Typical lag: 3h
Max lag: 3h

Typical lag: 10h?
Max lag: 3d?

IEDR Finance perform the invoice run which will:

- select all pending reservations against the deposit account
- generate the invoice
- generate the receipt
- reduce the actual balance of the deposit account by the total of all the selected reservations
- flag the reservations as having been processed

Figure 4 : Registration of Billable Domains by Registrar using Deposit Account

CRS-WEB-PushQ

Introduction

The CRS-WEB-PushQ (PushQ) scope of work is part of the overall scope of work for the IEDR BPR Development project. As part of the BPR project, the current MSD ('Mailed / Suspended / Deleted') process will be deprecated and replaced with the NRP (Non-Renewal Process). The NRP will be the process by which domains which are past their renewal dates are placed into quarantine, and ultimately deleted if not renewed within the mandated period. A new process to automate the updating of domain states (as per the DSM Specification) to place them into quarantine from within the CRS-WEB application is proposed called PushQ ('Push into Quarantine'). The scope of work of PushQ is to implement and test this process change to CRS-WEB.

REQ#0078: The PushQ process must be implemented as part of the Core.

«Constraint» Requirement:
The PushQ process is required to be implemented as part of the existing CRS-WEB component. The implementation will be in line with the MVC architectural principle.

REQ#0038: Forced PushQ execution.

«Functional» Requirement:
Privileged IEDR users must be able to force the execution of the Push Q process on a selected set of domains.

REQ#0080: PushQ process will be scheduled.

«Functional» Requirement:
The PushQ process will execute on a set schedule.

For usability, the PushQ process schedule will be configurable via a CRS-WEB interface by privileged users.

REQ#0039: Priviledged IEDR users must be able to configure the schedule of the Push Q process.

«Usability» Requirement:
The PushQ schedule will be configurable by privileged IEDR users.

REQ#0079: The PushQ process must implement the specified events of the DSM.

«Functional» Requirement:
The PushQ process will comprise part of the Domain State Machine. Specifically, it will implement the following events in the state machine:

RenewalDatePasses: The PushQ process will select all domains which are in a state that respond to the RenewalDatePasses event. For each such domain, the PushQ process will check the renewal date of the domain to see if it is less than the current date. If it is, then the associated transition and action is taken.

SuspensionDatePasses: The NPR process mandates that 14 days after a domain enters the NRP (either voluntarily or involuntarily) the domain is suspended, and will not be included in following Rebuilds of the dot IE zone file. The PushQ process will select all domains which are in a state that reacts to the SuspensionDatePasses event. For each such domain, the PushQ process will check the suspension date of the domain to see if it is less than the current date. If it is, then the associated transition and action is taken.

DeletionDatePasses: The PushQ process will select all domains which are in a state that respond to the DeletionDatePasses event. For each such domain, the PushQ process will check the deletion date of the domain to see if is less than the current date. If it is, then the associated transition and action is taken.

DeletedDomainCleanup; The PushQ process will select all domains which are in a state that respond to the DeletedDomainCleanup event. Each such domain will be removed, after updating history.

Refer to the Domain State Machine specification for the specifics of states that respond to these events, and the actions that are taken for any transition.

EPP

The scope of EPP is to be estimated separately.

We would ask you to estimate the effort and cost involved in offering the extensible provisioning protocol (EPP) as a channel by which Registrars can interact with the Registry.

We do not intend to deprecate the IE API protocol. Moreover, we would be interested in offering EPP in tandem with the IE API protocol.

As such, can you please consider the technical implications of offering EPP as a protocol which would work with the CRS-API application.

Please offer a quote and project plan for this as a separate piece of work from the BPR project.

Miscellaneous Technical Requirements

REQ#0024: Pricing tables implemented to future proof against arbitrary rate changes.

«Soln. Idea» Requirement:
[detail]

REQ#0061: Passwords must not be displayed, emailed or stored in plaintext.

«Security» Requirement:
[detail]

REQ#0151: Required changes to existing documentation should be provided.

«Doc/Training» Requirement:
Updated revisions to the following are required to be supplied prior to go-live:

- CRS API protocol specification

Transition

REQ#0147: ETL process for data migration to new schema.

«Migration» Requirement:
Proposed approach for testing and go-live:

- separate, empty db, new system (CRS) etc.
- ETL process to populate new db from old, transforming data as required.
- this can be used for QA and UAT
- once ready to go live, ETL can be performed again, and made production
- test system needed to be firewalled to prevent outside access / communication

An ETL process / script is required to perform the data migration.

REQ#0148: Migration of Domain State and Domain State History

«Functional» Requirement:
Issue: Domain history will have different structure

- potential solution is to:

a) retain but deprecate unused domain table fields
b) don't overload existing fields but use new field for state
c) for historic records, new state field can have 'historic' state value. Historic state will be an 'island' - can't get in or out of it.

REQ#0149: Separate enviroment is required for transition to go-live

«Migration» Requirement:
Proposed approach for go-live:

- separate, empty db, new system (CRS) etc.
- ETL process to populate new db from old, transforming data as required.
- once ready to go live, ETL can be performed again, and made production
- system needed to be firewalled to prevent outside access / communication until go-live

Minimum 4 machines required for new environment:
- DB
- CRS (web, api, ws-api)
- Webserver for NRC (PHP)
- webserver for API reverse proxy

but if UAT mandates availability of intranet, cp, acp etc. on new env. then more machines required.

Waiting Room

These requirements are currently out of scope but are listed here for context of potential future requirements, which may affect current implementation decisions.

It is assumed that the requirements listed here WON'T be implemented.

REQ#0025: Customised emails per email type per registrar.

«Waiting Room» Requirement:
[detail]

REQ#0036: Deposit a/c top-ups and other transactions implemented in similar way to Invoicing.

«Waiting Room» Requirement:
For future development.

REQ#0059: Domains will have unique IDs for each new registration.

«Waiting Room» Requirement:
[detail]

REQ#0075: AUTH codes for transfer between Regs / change of Bill-C

«Functional» Requirement:
See doc. from DC, 5 dec.

REQ#0082: Deleted domains may need to be pulled back into previous state

«Functional» Requirement:
Low priority, probably not needed. Source: BG.

REQ#0093: Unique Domain IDs

«Waiting Room» Requirement:

REQ#0094: Secondary market for domains

«Waiting Room» Requirement:
sedo.com

REQ#0096: No Deletion of Domain Objects from System

«Waiting Room» Requirement:
- They are simply in some kind of 'available for registration' state.