LOIS, DECRETS, ORDONNANCES ET REGLEMENTS
WETTEN, DECRETEN, ORDONNANTIES EN VERORDENINGEN

SERVICE PUBLIC FEDERAL TECHNOLOGIE DE L'INFORMATION ET DE LA COMMUNICATION

F. 2007 — 145 [C − 2006/12614]

7 DECEMBRE 2006. — Arrêté royal fixant les spécifications et la procédure d'enregistrement des appareils de lecture pour la carte d'identité électronique et modifiant l'arrêté royal du 13 février 1998 relatif aux spécifications des appareils de lecture de la carte d'identité sociale

RAPPORT AU ROI

Sire,

Le présent projet d'arrêté royal que nous avons l'honneur de présenter à Votre Majesté, vise à exécuter l'article 6quinquies de la loi du 19 juillet 1991 relative aux registres de la population et aux cartes d'identité et modifiant la loi du 8 août 1983 organisant un Registre national des personnes physiques, inséré par la loi du 25 mars 2003. L'article précité stipule que les normes et les spécifications techniques et fonctionnelles auxquelles doivent satisfaire les appareils (le lecteur de cartes) et les applications qui rendent possible la lecture et la mise à jour des données reprises de manière électronique sur la carte d'identité peuvent être déterminées.

La détermination de normes et d'exigences techniques et fonctionnelles est souhaitable dans l'intérêt des utilisateurs de la carte d'identité électronique et des diverses applications. Si l'on veut encourager l'utilisation de la carte d'identité électronique, il faut veiller à ce qu'il y ait suffisamment de lecteurs de cartes en circulation capables de lire la carte d'identité électronique.

Afin d'éliminer dans le chef de l'utilisateur le doute quant à la compatibilité des lecteurs avec la carte d'identité électronique, une procédure d'enregistrement libre est prévue pour les fournisseurs de lecteurs compatibles. Cette procédure permet de contrôler la conformité des appareils aux normes et spécifications déterminées afin de pouvoir garantir à l'utilisateur et à l'acheteur potentiel que les appareils fonctionnent correctement. La conformité d'un lecteur de cartes aux standards précités est confirmée au moyen d'un label.

Enfin, la procédure d'enregistrement vise à éviter, par le biais de l'attribution d'un label, l'utilisationd'appareils susceptibles d'endommager la carte d'identité électronique.

L'utilisation du label, déposé comme marque, par les fournisseurs d'appareils de lecture est soumise à certaines exigences minimales. Les spécifications relatives au label ainsi que ses conditions d'utilisation sont reprises en annexe au présent projet d'arrêté.

Il convient de viser une interopérabilité maximale des divers lecteurs de cartes; autrement dit, les autorités doivent faire en sorte que les lecteurs de cartes utilisés pour la carte d'identité électronique soient également compatibles avec d'autres types de cartes utilisés dans la relation avec les autorités, et en particulier avec la carte d'identité sociale. C'est la raison pour laquelle on choisit d'utiliser la même procédure d'enregistrement que celle utilisée actuellement pour les fournisseurs d'appareils de lecture de la carte d'identité sociale. Dans un souci de clarté, on choisit dès lors de régler les deux procédures d'enregistrement par le biais d'un seul et même arrêté. A cette fin, le présent arrêté modifie l'arrêté royal du 13 février 1998 relatif aux spécifications des appareils de lecture de la carte d'identité sociale.

FEDERALE OVERHEIDSDIENST INFORMATIE- EN COMMUNICATIETECHNOLOGIE

N. 2007 — 145 [C − 2006/12614]

7 DECEMBER 2006. — Koninklijk besluit tot vaststelling van de specificaties en registratieprocedure van de leesapparatuur voor de elektronische identiteitskaart en tot wijziging van het koninklijk besluit van 13 februari 1998 houdende specificaties van de leesapparatuur voor de sociale identiteitskaart

VERSLAG AAN DE KONING

Sire,

Het ontwerp van koninklijk besluit dat wij de eer hebben Uwe Majesteit voor te leggen, strekt ertoe artikel 6quinquies van de wet van 19 juli 1991 betreffende de bevolkingsregisters en de identiteitskaarten en tot wijziging van de wet van 8 augustus 1983 tot regeling van een Rijksregister van de natuurlijke personen, ingevoegd bij wet van 25 maart 2003 uit te voeren. Voormeld artikel voorziet dat normen en functionele en technische specificaties kunnen worden vastgelegd waaraan de apparatuur (de kaartlezer) en de toepassingen moeten voldoen teneinde het uitlezen en het bijwerken van de elektronisch op de identiteitskaart opgeslagen gegevens mogelijk te maken.

Het vastleggen van normen en van technische en functionele vereisten is, in het belang van de gebruikers van de elektronische identiteitskaart en de diverse toepassingen, wenselijk. Wanneer men het gebruik van de elektronische identiteitskaart wil stimuleren dan dient men er voor te zorgen dat er voldoende kaartlezers in omloop zijn die de elektronische identiteitskaart kunnen lezen.

Om bij de gebruiker geen twijfel te laten bestaan welke lezers verenigbaar zijn met de elektronische identiteitskaart wordt er een vrije registratieprocedure voorzien voor leveranciers van verenigbare leesapparatuur. Via de procedure kan de conformiteit van de apparatuur met de vastgestelde normen en specificaties worden gecontroleerd zodat aan de gebruiker en de potentiële koper ervan de waarborg kan worden geboden dat de apparatuur juist functioneert. Via een label wordt de conformiteit van een kaartlezer aan de voormelde standaarden bevestigd.

Tenslotte beoogt de registratieprocedure via de toekenning van een label te vermijden dat apparatuur wordt gebruikt die de elektronische identiteitskaart zou kunnen beschadigen.

Het gebruik van het label, als merk gedeponeerd, door leveranciers van leesapparatuur wordt aan bepaalde minimale voorwaarden onderworpen. De specificaties van het label evenals de gebruiksvoorwaarden die daarop betrekking hebben, worden als bijlage bij het ontwerp van dit besluit opgenomen.

Er moet worden gestreefd naar een maximale interoperabiliteit van de diverse kaartlezers; anders gesteld dient de overheid ervoor te zorgen dat de kaartlezers die de elektronische identiteitskaart kunnen lezen ook andere kaarttypes die worden gebruikt in de relatie met de overheid, in het bijzonder de sociale identiteitskaart, kunnen lezen. Vandaar dat ervoor wordt gekozen om dezelfde registratieprocedure te hanteren als degene die momenteel gebruikt wordt voor de leveranciers van leesapparatuur voor de sociale identiteitskaart. Omwille van duidelijkheidsredenen wordt er daarom geopteerd om beide registratieprocedures via één en hetzelfde besluit te regelen. Hiertoe wordt via dit besluit het koninklijk besluit van 13 februari 1998 houdende specificaties van de leesapparatuur voor de sociale identiteitskaart gewijzigd.

La procédure d'enregistrement peut être résumée comme suit :

Le fournisseur d'appareils de lecture doit être en mesure de montrer aux autorités la conformité de l'appareil aux spécifications fonctionnelles et aux normes de standardisation, telles que reprises en annexe au présent arrêté royal.

Si cette conformité est prouvée, le fournisseur est enregistré par les autorités et pourvoit ensuite son appareil de lecture d'un numéro d'enregistrement électronique et appose sur l'appareil le label destiné à cet effet.

La liste des appareils de lecture conformes, ainsi que les spécifications relatives à ces lecteurs, peuvent également être consultées sur le site web de l'autorité compétente.

Nous avons l'honneur d'être,
Sire,
De Votre Majesté
le très respectueux
et très fidèle serviteurs,

Le Ministre de l'Intérieur,
P. DEWAEL

Le Ministre de l'Economie,
M. VERWILGHEN

Le Ministre des Affaires sociales,
R. DEMOTTE

La Ministre des Classes moyennes et de l'Agriculture,
Mme S. LARUELLE

Le Ministre de l'Emploi,
P. VANVELTHOVEN

---

7 DECEMBRE 2006. — Arrêté royal fixant les spécifications et la procédure d'enregistrement des appareils de lecture pour la carte d'identité électronique et modifiant l'arrêté royal du 13 février 1998 relatif aux spécifications des appareils de lecture de la carte d'identité sociale

ALBERT II, Roi des Belges,
A tous, présents et à venir, Salut.

Vu la loi du 19 juillet 1991 relative aux registres de la population et aux cartes d'identité et modifiant la loi du 8 août 1983 organisant un Registre national des personnes physiques, notamment l'article 6*quinquies*, inséré par la loi du 25 mars 2003;

Vu la loi du 26 juillet 1996 portant modernisation de la sécurité sociale et assurant la viabilité des régimes légaux des pensions, notamment les articles 41 et 49;

Vu l'arrêté royal du 18 décembre 1996 portant des mesures en vue d'instaurer une carte d'identité sociale à l'usage de tous les assurés sociaux, en application des articles 38, 40, 41 et 49 de la loi du 26 juillet 1996 portant modernisation de la sécurité sociale et assurant la viabilité des régimes légaux des pensions, sanctionné par la loi du 26 juin 1997, notamment l'article 4, alinéa trois;

Vu l'arrêté royal du 13 février 1998 relatif aux spécifications des appareils de lecture de la carte d'identité sociale;

Vu l 'avis de l'Inspection des Finances, donné le 14 mai 2006;

Vu l'accord de Notre Ministre du Budget, donné le 30 juin 2006;

Vu l'avis de la Banque Carrefour de la Sécurité sociale, donné le 26 septembre 2006;

De registratieprocedure kan kort worden omschreven als volgt :

De leverancier van de leesapparatuur dient tegenover de overheid de verenigbaarheid te kunnen aantonen met de functionele specificaties en de normalisatienormen, die als bijlage bij dit koninklijk besluit zijn opgenomen.

Bij bewijs van verenigbaarheid wordt de leverancier door de overheid geregistreerd en voorziet de leverancier vervolgens zijn leesapparatuur van een elektronisch registratienummer en een daartoe bestemd label.

De lijst van verenigbare leesapparatuur, alsook de specificaties betreffende deze lezers, is tevens op de website van de bevoegde overheid te raadplegen.

Wij hebben de eer te zijn,
Sire,
van Uwe Majesteit
de zeer eerbiedige
en zeer getrouwe dienaren,

De Minister van Binnenlandse Zaken,
P. DEWAEL

De Minister van Economie,
M. VERWILGHEN

De Minister van Sociale Zaken,
R. DEMOTTE

De Minister van Middenstand en Landbouw,
Mevr. S. LARUELLE

De Minister van Werk,
P. VANVELTHOVEN

---

7 DECEMBER 2006. — Koninklijk besluit tot vaststelling van de specificaties en registratieprocedure van de leesapparatuur voor de elektronische identiteitskaart en tot wijziging van het koninklijk besluit van 13 februari 1998 houdende specificaties van de leesapparatuur voor de sociale identiteitskaart

ALBERT II, Koning der Belgen,
Aan allen die nu zijn en hierna wezen zullen, Onze Groet.

Gelet op de wet van 19 juli 1991 betreffende de bevolkingsregisters en de identiteitskaarten en tot wijziging van de wet van 8 augustus 1983 tot regeling van een Rijksregister van de natuurlijke personen, inzonderheid op artikel 6*quinquies*, ingevoegd bij wet van 25 maart 2003;

Gelet op de wet van 26 juli 1996 tot modernisering van de sociale zekerheid en tot vrijwaring van de leefbaarheid van de wettelijke pensioenstelsels, inzonderheid op artikelen 41 en 49;

Gelet op het koninklijk besluit van 18 december 1996 houdende maatregelen met het oog op de invoering van een sociale identiteitskaart ten behoeve van alle sociaal verzekerden, met toepassing van de artikelen 38, 40, 41 en 49 van de wet van 26 juli 1996 houdende de modernisering van de sociale zekerheid en tot vrijwaring van de wettelijke pensioenstelsels, bekrachtigd bij de wet van 26 juni 1997, inzonderheid op artikel 4, derde lid;

Gelet op het koninklijk besluit van 13 februari 1998 houdende specificaties van de leesapparatuur voor de sociale identiteitskaart;

Gelet op het advies van de Inspectie van Financiën, gegeven op 14 mei 2006;

Gelet op de akkoordbevinding van Onze Minister van Begroting gegeven op 30 juni 2006;

Gelet op het advies van de Kruispuntbank van de sociale zekerheid, gegeven op 26 september 2006;

Vu l'accomplissement des formalités prescrites par la directive du Parlement Européenne et le Conseil 98/34/EG du 22 juin 1998 prévoyant une procédure d'information dans le domaine des normes et réglementations techniques;

Vu avis 40.903/1/V du Conseil d'Etat, donné le 3 août 2006, en application de l'article 84, § 1er, alinéa 1er, 1°, des lois coordonnées sur le Conseil d'Etat;

Sur la proposition de Notre Ministre de l'Intérieur, de Notre Ministre de l'Economie, de Notre Ministre des Affaires sociales et de la Santé publique, de Notre Ministre des Classes moyennes et de l'Agriculture et de Notre Ministre de l'Emploi, et de l'avis de Nos Ministres qui en ont délibéré en Conseil,

Nous avons arrêté et arrêtons :

Article 1er. L'intitulé de l'arrêté royal du 13 février 1998 relatif aux spécifications des appareils de lecture de la carte d'identité sociale, est remplacé par l'intitulé suivant : "Arrêté royal fixant les spécifications et la procédure d'enregistrement des appareils de lecture pour la carte d'identité électronique et la carte d'identité sociale".

Art. 2. A l'article 1er du même arrêté sont apportées les modifications suivantes:

a) au deuxième tiret, les mots "-"fournisseur": toute personne qui fabrique, distribue directement ou indirectement, ou assure la maintenance de tout ou partie d'un appareil de lecture destiné à lire ou écrire des données figurant au sein de la carte d'identité sociale;" sont remplacés par les mots "-"fournisseur" : toute personne qui fabrique, distribue directement ou indirectement, ou assure la maintenance de tout ou partie d'un appareil de lecture destiné à lire ou écrire des données figurant au sein de la carte d'identité sociale et/ou de la carte d'identité électronique";

b) un troisième tiret est ajouté, rédigé comme suit: "-loi" : Loi du 19 juillet 1991 relative aux registres de la population et aux cartes d'identité et modifiant la loi du 8 août 1983 organisant un Registre national des personnes physiques."

Art. 3. L'article 2 du même arrêté est remplacé comme suit : "Les appareils de lecture visés à l'article 4, alinéa 3, de l'arrêté royal ou visés à l'article 6quinquies de la loi, doivent respecter les spécifications fonctionnelles applicables et les normes de standardisation applicables décrites en annexe, pour que les cartes d'identité puissent être utilisées."

Art. 4. Dans la première phrase de l'article 3 du même arrêté, les mots "pour l'emploi de la carte d'identité sociale" sont insérés entre les mots "des personnes habilitées" et les mots "peuvent obtenir la documentation suivante".

Art. 5. Un article 3*bis*, rédigé comme suit, est inséré dans le même arrêté : "Les fournisseurs souhaitant mettre à disposition ces appareils de lecture pour l'emploi de la carte d'identité électronique peuvent obtenir auprès du Service public fédéral Technologie de l'Information et de la Communication la documentation suivante :

1° spécifications du système de la carte d'identité électronique,

2° descriptif des commandes nécessaires à la lecture et à l'écriture de la carte d'identité électronique,

3° descriptif de la procédure pour tester en ligne la carte d'identité électronique,

4° spécimens de la carte d'identité électronique."

Art. 6. A l'article 4 du même arrêté sont apportées les modifications suivantes:

a) dans la première phrase, les mots "des personnes habilitées" et "auprès de la Banque Carrefour de la sécurité sociale" sont supprimés;

b) dans la disposition au 2°, les mots "concernant d'une part le respect de la norme ISO 7816 - 1 à 4 et d'autre part le respect des spécifications fonctionnelles ainsi que des spécifications techniques et critères de qualité et de performance figurant en annexe" sont remplacés par les mots "concernant d'une part le respect des parties pertinentes de la norme ISO 7816 et d'autre part le respect des spécifications fonctionnelles ainsi que des spécifications techniques applicables et critères de qualité et de performance applicables dans l'annexe I au présent arrêté".

Gelet op het feit dat is voldaan aan de formaliteiten voorgeschreven door Richtlijn 98/34/EG van 22 juni 1998 van het Europees parlement en de Raad betreffende de informatieprocedure op het gebied van normen en technische voorschriften;

Gelet op advies 40.903/1/Vvan de Raad van State, gegeven op 3 augustus 2006, met toepassing van artikel 84, § 1, eerste lid, 1°, van de gecoördineerde wetten op de Raad van State;

Op de voordracht van Onze Minister van Binnenlandse Zaken, Onze Minister van Economie, Onze Minister van Sociale Zaken en Volksgezondheid, Onze Minister van Middenstand en Landbouw en van Onze Minister van Werk, en op het advies van Onze in Raad vergaderde Ministers,

Hebben Wij besloten en besluiten Wij :

Artikel 1. Het opschrift van het koninklijk besluit van 13 februari 1998 houdende specificaties van de leesapparatuur voor de sociale identiteitskaart wordt vervangen als volgt : "Koninklijk besluit tot vaststelling van de specificaties en de registratieprocedure van de leesapparatuur voor de elektronische identiteitskaart en de sociale identiteitskaart".

Art. 2. In artikel 1 van hetzelfde koninklijk besluit worden de volgende wijzigingen aangebracht :

a) in het tweede streepje worden de woorden : " -"leverancier" : iedere persoon die leesapparatuur fabriceert, rechtstreeks of onrechtstreeks verdeelt, of die instaat voor het onderhoud van het hele of een deel van het leesapparaat dat dient om gegevens uit te lezen of te schrijven op de sociale identiteitskaart" vervangen door de woorden " -"leverancier" : iedere persoon die leesapparatuur fabriceert, rechtstreeks of onrechtstreeks verdeelt, of die instaat voor het onderhoud van het hele of een deel van het leesapparaat dat dient om gegevens uit te lezen of te schrijven op de sociale identiteitskaart en/of op de elektronische identiteitskaart;";

b) een derde streepje wordt toegevoegd, luidend als volgt : "- "wet" : Wet van 19 juli 1991 betreffende de bevolkingsregisters en de identiteitskaarten en tot wijziging van de wet van 8 augustus 1983 tot regeling van een Rijksregister van de natuurlijke personen. »

Art. 3. Artikel 2 van hetzelfde besluit wordt vervangen als volgt : "De leesapparatuur bedoeld in artikel 4, derde lid, van het koninklijk besluit of in artikel 6quinquies van de wet moet voldoen aan de toepasselijke functionele specificaties en aan de toepasselijke in bijlage beschreven normalisatienormen, opdat de identiteitskaarten zouden kunnen worden gebruikt".

Art. 4. In artikel 3 van hetzelfde besluit worden in de eerste zin de woorden "voor gebruik van de sociale identiteitskaart" toegevoegd tussen de woorden "van de bevoegde personen," en "kunnen de volgende documentatie bekomen".

Art. 5. Een artikel 3*bis* wordt in hetzelfde besluit ingevoegd, luidend als volgt : "De leveranciers die deze leesapparatuur ter beschikking wensen te stellen voor gebruik van de elektronische identiteitskaart, kunnen de volgende documentatie bekomen bij de Federale overheidsdienst Informatie- en Communicatietechnologie :

1° specificaties van het systeem van de elektronische identiteitskaart,

2° beschrijving van de commando's nodig voor het lezen en schrijven van de elektronische identiteitskaart,

3° beschrijving van de procedure voor het online testen van de elektronische identiteitskaart,

4° proefexemplaren van de elektronische identiteitskaart."

Art. 6. In artikel 4 van hetzelfde besluit worden de volgende wijzigingen aangebracht :

a) in de eerste zin worden de woorden "van de bevoegde personen" en "bij de Kruispuntbank van de sociale zekerheid" geschrapt;

b) in de bepaling onder 2° worden de woorden "met betrekking tot de naleving van de ISO-norm 7816 - 1 tot 4, en anderzijds met betrekking tot de naleving van de technische en functionele specificaties en van de kwaliteits- en performantiecriteria die in bijlage worden vermeld" vervangen door de woorden "met betrekking tot de naleving van de relevante delen van de ISO-norm 7816, en anderzijds met betrekking tot de naleving van de toepasselijke technische en functionele specificaties en van de toepasselijke kwaliteits- en performantiecriteria in bijlage I bij dit besluit".

Art. 7. Un article 4bis, rédigé comme suit, est inséré dans le même arrêté :

« Art. 4*bis*. § 1er. Le fournisseur dépose son dossier de référence :

1° auprès de la Banque Carrefour de la sécurité sociale pour autant que seul un numéro d'enregistrement soit demandé pour un appareil de lecture destiné à lire et/ou écrire des données figurant au sein de la carte d'identité sociale;

2° auprès du Service public fédéral Technologie de l'Information et de la Communication pour autant que seul un numéro d'enregistrement soit demandé pour un appareil de lecture destiné uniquement à lire et/ou écrire des données figurant au sein de la carte d'identité électronique.

§ 2. Si un numéro d'enregistrement est demandé simultanément pour un appareil de lecture destiné à lire et/ou écrire des données figurant au sein de la carte d'identité sociale et la carte d'identité électronique, le fournisseur dépose les deux dossiers de référence distincts et spécifie à chaque fois qu'il s'agit d'une procédure d'enregistrement simultanée.

La Banque Carrefour de la sécurité sociale et le Service public fédéral Technologie de l'Information et de la Communication s'informent sans délai du dépôt simultané par le fournisseur. »

§ 3. La demande d'enregistrement d'un appareil de lecture de la carte d'identité électronique est faite au moyen du modèle de formulaire établi dans l'annexe II du présent arrêté.

Art. 8. A l'article 5 du même arrêté sont apportées les modifications suivantes :

a) la première phrase est remplacée comme suit : "La Banque Carrefour de la sécurité sociale ou, après avis conforme du Service public fédéral Intérieur, le Service public fédéral Technologie de l'Information et de la Communication, attribue(nt) dans une période de 45 jours civils suivant la réception du dossier de référence un numéro d'enregistrement à chaque dossier de référence déposé, pour autant que ce dossier comprenne les pièces citées à l'article 4 et qu'il en ressorte qu'il satisfait aux conditions citées à l'article 2.";

b) la deuxième phrase est remplacée comme suit : "Le fournisseur qui dépose un dossier de référence s'engage, à incorporer électroniquement dans les appareils de lecture le numéro d'enregistrement attribué par la Banque Carrefour de la sécurité sociale et/ou le Service public fédéral Technologie de l'Information et de la Communication. En ce qui concerne les appareils de lecture utilisés pour la carte d'identité électronique, le fournisseur s'engage en outre, selon les spécifications prévues à l'annexe II du présent arrêté, à pourvoir de façon visible d'un label tout appareil de lecture enregistré."

Art. 9. Un article 5*bis*, rédigé comme suit, est inséré dans le même arrêté :

« Art. 5*bis*. La Banque Carrefour de la sécurité sociale, en ce qui concerne la carte d'identité sociale, et le Service public fédéral Technologie de l'Information et de la Communication, en ce qui concerne la carte d'identité électronique, disposent du droit de soumettre les appareils de lecture enregistrés à un contrôle approfondi en cas de plainte ou de soupçon de non-conformité de l'appareil de lecture au dossier de référence déposé. »

Art. 10. L'alinéa 2 de l'article 6 du même arrêté est remplacée comme suit : "Le fournisseur qui a déposé un dossier de référence s'engage à adapter les appareils de lecture qu'il a installés en cas d'évolution des spécifications fonctionnelles et des normes de standardisation qui lui sont communiquées par la Banque Carrefour de la sécurité sociale et/ou le Service public fédéral Technologie de l'Information et de la Communication."

Art. 11. A l'article 6*bis* du même arrêté sont apportées les modifications suivantes :

a) le première alinéa est remplacée comme suit : "Chaque modification des spécifications fonctionnelles qui est communiquée par la Banque carrefour de la sécurité sociale et/ou le Service public fédéral Technologie de l'Information et de la Communication, est enregistrée auprès de la Banque carrefour de la sécurité sociale, en ce qui concerne les appareils de lecture de la carte d'identité sociale et/ou auprès du Service public fédéral Technologie de l'Information et de la Communication, en ce qui concerne les appareils de lecture de la carte d'identité électronique.";

b) l'alinéa 2 est remplacée comme suit : "Le délai entre la fourniture des nouvelles spécifications fonctionnelles par la Banque carrefour de la sécurité sociale et/ou le Service public fédéral Technologie de l'Information et de la Communication et la possibilité de décharger les adaptations auprès des utilisateurs des appareils de lecture ne peut excéder trois mois.";

Art. 7. Een artikel 4*bis* wordt in hetzelfde besluit ingevoegd, luidend als volgt :

« Art. 4*bis*. § 1. De leverancier dient zijn referentiedossier in :

1° bij de Kruispuntbank van de sociale zekerheid voor zover enkel een registratienummer wordt aangevraagd voor een leesapparaat dat dient om gegevens uit te lezen en/of te schrijven op de sociale identiteitskaart;

2° bij de Federale overheidsdienst Informatie- en Communicatietechnologie voor zover enkel een registratienummer wordt aangevraagd voor een leesapparaat dat dient om uitsluitend gegevens uit te lezen en/of te schrijven op de elektronische identiteitskaart.

§ 2. Indien gelijktijdig een registratienummer wordt aangevraagd voor een leesapparaat dat dient om zowel op de sociale identiteitskaart als op de elektronische identiteitskaart gegevens uit te lezen en/of te schrijven, dient de leverancier twee aparte referentiedossiers in en duidt telkens aan dat het gaat om een gelijktijdige registratieprocedure.

De Kruispuntbank van de sociale zekerheid en de Federale overheidsdienst Informatie- en Communicatietechnologie informeren elkaar onverwijld van de gelijktijdige neerlegging door de leverancier. »

§ 3. De aanvraag tot registratie van een leesapparaat voor de elektronische identiteitskaart geschiedt aan de hand van het modelformulier vastgesteld in bijlage II bij dit besluit.

Art. 8. In artikel 5 van hetzelfde besluit worden de volgende wijzigingen aangebracht :

a) de eerste zin wordt vervangen als volgt : "De Kruispuntbank van de sociale zekerheid of de Federale overheidsdienst Informatie- en Communicatietechnologie na eensluidend advies van de Federale Overheidsdienst Binnenlandse zaken, kent binnen een periode van 45 kalenderdagen na ontvangst van het referentiedossier een registratienummer toe aan elk ingediend referentiedossier, voor zover het de in artikel 4 vermelde stukken omvat en daaruit blijkt dat aan de in artikel 2 vermelde voorwaarden is voldaan";

b) de tweede zin wordt vervangen als volgt : "De leverancier die een referentiedossier indient, verbindt er zich toe, het door de Kruispuntbank van de sociale zekerheid en/of de Federale overheidsdienst Informatie- en Communicatietechnologie toegekend registratienummer op elektronische wijze in de leesapparatuur te brengen. Voor wat betreft de leesapparatuur voor gebruik van de elektronische identiteitskaart verbindt de leverancier zich er bovendien toe, volgens de in bijlage II bij dit besluit voorziene specificaties, op een zichtbare wijze elke geregistreerd leesapparaat van een label te voorzien."

Art. 9. Een artikel 5*bis* wordt in hetzelfde besluit ingevoegd, luidend als volgt :

« Art. 5*bis*. De Kruispuntbank van de sociale zekerheid, wat betreft de sociale identiteitskaart, en de Federale overheidsdienst Informatie- en Communicatietechnologie, wat betreft de elektronische identiteitskaart, beschikken over het recht om de geregistreerde leesapparatuur aan een grondige controle te onderwerpen bij klacht of vermoeden van niet-overeenstemming van het leesapparaat met het ingediende referentiedossier. »

Art. 10. In artikel 6 van hetzelfde besluit wordt het tweede lid vervangen als volgt "De leverancier die een referentiedossier heeft ingediend, verbindt zich ertoe de door hem geïnstalleerde leesapparatuur aan te passen ingeval de functionele specificaties en de normalisatienormen die hem door de Kruispuntbank van de sociale zekerheid en/of de Federale overheidsdienst Informatie- en Communicatietechnologie worden meegedeeld, evolueren."

Art. 11. In artikel 6*bis* van hetzelfde besluit worden de volgende wijzigingen aangebracht :

a) het eerste lid wordt vervangen als volgt : "Elke wijziging van de functionele specificaties, die wordt meegedeeld door de Kruispuntbank van de sociale zekerheid en/of de Federale overheidsdienst Informatie- en Communicatietechnologie, wordt geregistreerd bij de Kruispuntbank van de sociale zekerheid, wat betreft de leesapparaten voor de sociale identiteitskaart, en/of de Federale overheidsdienst Informatie- en Communicatietechnologie, wat betreft de leesapparaten voor de elektronische identiteitskaart.";

b) het tweede lid wordt vervangen als volgt : "De termijn tussen de levering van de nieuwe functionele specificaties door de Kruispuntbank van de sociale zekerheid en/of de Federale overheidsdienst Informatie- en Communicatietechnologie en de mogelijkheid om de aanpassingen af te laden bij de gebruikers van de leesapparatuur mag drie maanden niet overschrijden.";

c) le troisième alinéa est remplacée comme suit : ″Le dossier de référence relatif aux nouvelles spécifications fonctionnelles est soumis selon les conditions visées à l'article 4bis du présent arrêté par le déposant du dossier d'enregistrement original.″;

d) dans le quatrième alinéa les mots ″, en ce qui concerne la Banque carrefour de la sécurité sociale, ″ sont insérés entre les mots ″comprend″ et ″les éléments suivants″;

e) un cinquième alinéa est ajouté, rédigée comme suit : ″Le dossier de référence comprend, en ce qui concerne le Service public fédéral Technologie de l'Information et de la Communication, les éléments suivants qui relèvent de la responsabilité du déposant :

1. documentation des procédures de déchargement des nouvelles spécifications fonctionnelles sur les appareils de lecture;

2. manuel d'utilisation permettant à l'utilisateur de l'appareil de lecture de décharger en toute sécurité les nouvelles spécifications fonctionnelles sur son appareil de lecture;

3. contrats de maintenance modèles;

4. démonstration du déchargement de la nouvelle version sur chaque type d'appareils de lecture enregistrés.″.

Art. 12. L'article 7 du même arrêté est remplacé comme suit :

« Art. 7. Le fournisseur qui a reçu un numéro d'enregistrement s'engage par tous les moyens appropriés à aider la Banque Carrefour de la sécurité sociale et/ou le Service public fédéral Technologie de l'Information et de la Communication à pouvoir dépister l'origine d'éventuels problèmes de fonctionnement du lecteur de cartes enregistré et/ou de son logiciel ou d'éventuels problèmes de falsification des cartes d'identité sociale et/ou des cartes d'identité électroniques. »

Art. 13. L'article 8 du même arrêté est remplacé comme suit :

« Art. 8. Toute personne habilitée à utiliser la carte d'identité sociale et/ou la carte d'identité électronique peut consulter auprès de la Banque Carrefour, en ce qui concerne les appareils de lecture pour la carte d'identité sociale, et auprès du Service public fédéral Technologie de l'Information et de la Communication, en ce qui concerne les appareils de lecture pour la carte d'identité électronique, la liste des dossiers de référence ainsi que les dossiers de référence des modèles d'appareils de lecture tels que déposés par les fournisseurs, en ce compris les indications d'équivalence par rapport aux spécifications techniques, aux critères de qualité ou de performance. »

Art. 14. A l'article 9 du même arrêté les mots ″, Notre Ministre des Affaires sociales et de la Santé publique, Notre Ministre de l'Intérieur, Notre Ministre de l'Economie, Notre Ministre des Classes moyennes et de l'Agriculture et notre Ministre de l'Emploi, sont, chacun en ce qui les concerne,″ sont insérés entre les mots ″Affaires″ et ″chargés″.

Art. 15. L'annexe I^re^ jointe au présent arrêté remplace l'annexe à l'arrêté royal du 13 février 1998 relatif au spécifications des appareils de lecture de la carte d'identité sociale.

Les annexes II et III jointes au présent arrêté sont ajoutées commes annexes II et III au même arrêté royal.

Art. 16. Le présent arrêté entre en vigueur le jour de sa publication au *Moniteur belge*.

Art. 17. Notre Ministre de l'Intérieur, Notre Ministre de l'Economie, Notre Ministre des Affaires sociales et de la Santé publique, Notre Ministre des Classes moyennes et de l'Agriculture et Notre Ministre de l'Emploi sont, chacun en ce qui les concerne, chargés de l'exécution du présent arrêté.

Donné à Bruxelles, le 7 décembre 2006.

ALBERT

Par le Roi :

Le Ministre de l'Intérieur,
P. DEWAEL

Le Ministre de l'Economie,
M. VERWILGHEN

Le Ministre des Affaires sociales,
R. DEMOTTE

La Ministre des Classes moyennes et de l'Agriculture,
Mme S. LARUELLE

Le Ministre de l'Emploi,
P. VANVELTHOVEN

c) het derde lid wordt vervangen als volgt : ″Het referentiedossier met betrekking tot de nieuwe functionele specificaties wordt voorgelegd op de wijze zoals bepaald in artikel 4*bis* van dit besluit door de indiener van het oorspronkelijke registratiedossier.″;

c) In het vierde lid worden de woorden ″, wat de Kruispuntbank voor de sociale zekerheid betreft,″ ingevoegd tussen de woorden ″omvat″ en ″de volgende elementen″;

e) een vijfde lid wordt toegevoegd, luidend als volgt : ″Het referentiedossier omvat, wat de Federale overheidsdienst Informatie- en Communicatietechnologie betreft, de volgende elementen die tot de verantwoordelijkheid van de indiener behoren :

1. documentatie van de procedures tot aflading van de nieuwe functionele specificaties op de leesapparatuur;

2. gebruikershandboek aan de hand waarvan de gebruiker van de leesapparatuur de nieuwe functionele specificaties in alle veiligheid kan afladen op zijn leesapparatuur;

3. modelonderhoudscontracten;

4. demonstratie van de aflading van de nieuwe versie op elk type geregistreerde leesapparatuur.″

Art. 12. Artikel 7 van hetzelfde besluit wordt vervangen als volgt :

« Art. 7. De leverancier die een registratienummer heeft ontvangen, verbindt zich ertoe met alle passende middelen de Kruispuntbank van de sociale zekerheid en/of de Federale overheidsdienst Informatie- en Communicatietechnologie bij te staan, teneinde de oorzaak van eventuele werkingsproblemen van de geregistreerde kaartlezer en/of de daarbij horende software of van vervalsing van de sociale identiteitskaarten en/of elektronische identiteitskaarten te kunnen opsporen. »

Art. 13. Artikel 8 van hetzelfde besluit wordt vervangen als volgt :

« Art. 8. Iedere persoon die gemachtigd is om de sociale identiteitskaart en/of de elektronische identiteitskaart te gebruiken, kan bij de Kruispuntbank, wat betreft de leesapparaten voor de sociale identiteitskaart, en bij de Federale overheidsdienst Informatie- en Communicatietechnologie, wat betreft de leesapparaten voor de elektronische identiteitskaart, de lijst van de referentiedossiers alsook de referentiedossiers van de modellen van leesapparatuur raadplegen zoals die door de leveranciers werden ingediend, met inbegrip van de vermeldingen van gelijkwaardigheid ten opzichte van de technische specificaties, de performantie- of kwaliteitscriteria. »

Art. 14. Artikel 9 van hetzelfde besluit wordt gewijzigd als volgt : ″ Onze Minister van Sociale Zaken en Volksgezondheid, Onze Minister van Binnenlandse zaken, Onze Minister van Economie, Onze Minister van Middenstand en Landbouw en onze Minister van Werk zijn, elk wat hen betreft, belast met de uitvoering van dit besluit.″

Art. 15. De bij dit besluit gevoegde bijlage I vervangt de bijlage bij het koninklijk besluit van 13 februari 1998 houdende specificaties van de leesapparatuur voor de sociale identiteitskaart.

De bij dit besluit gevoegde bijlagen II en III worden toegevoegd als bijlagen II en III aan hetzelfde koninklijk besluit.

Art. 16. Dit besluit treedt in werking de dag waarop het in het *Belgisch Staatsblad* wordt bekendgemaakt.

Art. 17. Onze Minister van Binnenlandse Zaken, Onze Minister van Economie, Onze Minister van Sociale Zaken en Volksgezondheid, Onze Minister van Middenstand en Landbouw en Onze Minister van Werk, zijn, elk wat hen betreft, belast met de uitvoering van dit besluit.

Gegeven te Brussel, 7 december 2006.

ALBERT

Van Koningswege :

De Minister van Binnenlandse Zaken,
P. DEWAEL

De Minister van Economie,
M. VERWILGHEN

De Minister van Sociale Zaken,
R. DEMOTTE

De Minister van Middenstand en Landbouw,
Mevr. S. LARUELLE

De Minister van Werk,
P. VANVELTHOVEN

Annexe I^re^

Table des matières

Art. 2.

Spécifications des lecteurs

1. Compatibilité materielle avec la puce
2. Compatibilité logicielle avec la puce
    2.1. Tous les lecteurs
    2.2. Lecteurs avec clavier PIN
        2.2.1. Commandes de carte (APDU) à mettre en oeuvre
        2.2.2. Commandes de lecteur (APDU) à mettre en oeuvre
        2.2.3. Format PIN
3. Compatibilité logicielle avec le middleware
    3.1. Pilotes PC/SC
    3.2. Intégration au middleware
        3.2.1. SCR_Init ( )
        3.2.2. SCR_VerifyPIN ( )
        3.2.3. SCR_ChangePIN ( )
    3.3. Afficheur
        3.3.1. SCR_VerifyPIN ( )
        3.3.2. SCR_ChangePIN ( )

Art. 3.

Spécifications du middleware Identité

1. Introduction
    1.1. Matrice des environnements de développement
2. Spécifications fonctionnelles
    2.1. Gestion des versions et compatibilité
    2.2. Saisie du PIN
    2.3. Remarque qur la longueur maximale des paramètres
    2.4. Applications multifilières
    2.5. Organisation des API
    2.6. Fonctions d'initialisation et de terminaison
        2.6.1. BEID_Init
        2.6.2. BEID_Exit
    2.7. Fonctions d'identité
        2.7.1. BEID_GetID
        2.7.2. BEID_GetAddress
        2.7.3. BEID_GetPicture
        2.7.4. Fonctions d'identité hors ligne
            2.7.4.1. BEID_GetRawData
            2.7.4.2. BEID_SetRawData

2.8. Fonctions générales de haut niveau
2.8.1. BEID_BeginTransaction
2.8.2. BEID_EndTransaction
2.8.3. BEID_SelectApplication
2.8.4. BEID_ReadFile
2.8.5. BEID_WriteFile
2.8.6. BEID_VerifyPIN
2.8.7. BEID_ChangePIN
2.8.8. BEID_GetPINStatus
2.9. Fonctions de bas niveau
2.9.1. BEID_GetVersionInfo
2.9.2. BEID_SendAPDU
2.9.3. BEID_FlushCache
2.10. Identification PIN
2.10.1. Types de PIN
2.10.2. Utilisations du PIN
2.11. Paramètres de politique en entrée - OCSP et CRL
2.12. Statut des fonctions
2.12.1. Codes d'erreur généraux
2.13. Contrôle de signature et validation de certificat
2.13.1. Contrôle de signature
2.13.2. Résultat du contrôle et de la validation du certificate
2.13.3. Politiques OCSP et CRL appliqués
3. Interfaces de programmation
3.1. APIC
3.1.1. Structures
3.1.2. Fonctions
3.2. API Java
3.2.1. Data Classes
3.2.2. Main Classe
3.2.3. Applet Classe
3.3. API ActiveX
3.3.1. Définitions d'interface
3.3.2. Fonctions
4. Installation
4.1. Package d'installation
4.2. Runtime
4.3. Bibliothèque C
4.4. Java

Art. 3*bis*

Spécifications du middleware Crypto

1. But
2. Architecture
2.1. Interfaces
2.1.1. l'Interface Crypto API
2.1.1.1. CSP-Architecture de haut niveau
2.1.1.2. CSP-Architecture de bas niveau
2.1.2. l'Interface PKCS (11
2.1.2.1. PKCS (11-Architecture de haut niveau

Art. 1er. Procédure d'enregistrement d'un lecteur

1. Introduction

1.1. But des annexes

Ce document référence les spécifications et décrit les exigences à remplir pour obtenir le label " compatible eID belge " d'un lecteur.

Avant le processus d'enregistrement, le demandeur (fabricant, revendeur, distributeur ou intégrateur du lecteur) accomplira dans ses services internes un ensemble d'opérations/scénarios qui serviront de base aux certificats de conformité, aux données d'essai et à la validation finale :

1) Obtenir les certificats d'accréditation ISO, garantissant la conformité technique, auprès des instances ISO (voir le chapitre "Spécifications techniques").

2) Exécuter (dans ses services internes, sur son propre équipement) les scénarios de test de bas niveau et obtenir les données d'essai demandées (voir le chapitre "Scénarios de bas niveau").

3) Exécuter (en ligne, sur le site de test eID) les scénarios de validation de haut niveau et obtenir les données de confirmation nécessaires (voir le chapitre "Scénarios de haut niveau").

Le demandeur complétera ensuite le formulaire d'enregistrement (avec tous les détails demandés au sujet du produit) et le soumettra à l'instance d'enregistrement des lecteurs.

Note 1 : le formulaire d'enregistrement contient une déclaration de conformité du fabricant, certifiant que le demandeur a accompli toutes les étapes précédentes et que l'information fournie avec la déclaration est correcte.

Note 2 : l'instance d'enregistrement des lecteurs se réserve le droit d'évaluer plus en détail la conformité du lecteur (p.ex. en cas de réclamation). Trois exemplaires du produit visé dans le formulaire d'enregistrement accompagneront le formulaire, aux fins d'archivage et d'évaluation.

1.2. Portée

Le présent document porte principalement sur l'enregistrement des lecteurs. Par "lecteur", nous entendons tout dispositif servant d'interface SmartCard avec la carte eID (autonome ou non, incorporé ou non, propre à l'application ou non).

Le document ne couvre pas les besoins spécifiques liés aux lecteurs utilisés pour des opérations de gestion de carte (activation, (ré)initialisation de PIN/PUK, changement dans les données, etc.) qui sont propres à l'émetteur de la carte (p.ex. activation initiale de la carte), pour le blocage/déblocage de la carte (p.ex. si le nombre de tentatives de saisie du PIN atteint la limite), et/ou pour une modification de la carte (p.ex. quand il faut mettre à jour le fichier d'adresses). Ce type de lecteur nécessite des fonctions spéciales de fusion PIN/PUK, qui sortent du cadre de ce document.

1.3. Catégories des lecteurs

Les lecteurs sont répartis en catégories suivant les critères ci-dessous :

- connectable : le lecteur est équipé d'une interface hôte (p.ex. USB, RSR232, PCMCIA, etc.)

- affichage sécurisé : le lecteur possède un affichage réservé à la visualisation des messages

- clavier PIN sécurisé : le lecteur possède un clavier numérique PIN destiné à la saisie du numéro PIN

- application sécurisée : le lecteur met en place un canal de communication sécurisé avec l'application qui le commande

	Value Checker (lecteur de valeur)	Transparent Reader (lecteur transparent)	Secure Display Reader (lecteur à affichage sécurisé)	Secure pinpad Reader (lecteur à clavier PIN sécurisé)	Trusted Reader (lecteur de confiance)
Connectable		X	X	X	X
Affichage sécurisé			X		X
Clavier PIN sécurisé				X	X
Application sécurisée					X

- Value Checker (lecteur de valeur) : lecteur non connecté, conçu pour lire facilement le contenu de la puce, sans être capable d'accomplir des transactions cryptographiques (ne nécessite pas d'enregistrement).

- Transparent Reader (lecteur transparent) : lecteur connectable, sans affichage ni clavier PIN dédié (PIN et code d'état sont introduits et affichés sur des supports non dignes de confiance comme le clavier normal et le moniteur).

- Secured reader (lecteur sécurisé) : lecteur connectable avec affichage et/ou clavier PIN dédié (PIN et code d'état sont introduits directement sur le clavier PIN et/ou présentés sur l'affichage).

- Trusted Reader (lecteur de confiance) : lecteur sécurisé, avec applications de confiance, établissant un canal de communication sécurisé avec le lecteur.

1.4. Plates-formes

Le support de la carte eID étant lié à la plate-forme, l'enregistrement sera demandé par plate-forme (le lecteur peut naturellement faire l'objet de plusieurs enregistrements, pour des plates-formes différentes).

Les plates-formes suivantes sont actuellement prises en charge, mais l'instance d'enregistrement des lecteurs se réserve le droit d'ajouter ou de supprimer des plates-formes en fonction de la demande du marché, du support des fournisseurs, etc.

Microsoft	**Macintosh**	**Linux (2.x)**	
Windows 98	MacOSX10.1	I386	
Windows ME	MacOSX10.2		
Windows 2000	MacOSX10.3		
Windows XP			

En raison du grand nombre de variantes dans les plates-formes, nous supposons que les tests/scénarios sont exécutés sur la version la plus légère, nue, de la plate-forme en question (p.ex. édition familiale plutôt que professionnelle, édition personnelle plutôt que serveur, etc.), après installation de tous les correctifs (de sécurité) disponibles. Si le lecteur nécessite une variante ou un correctif spécifique, le fait doit être précisé clairement dans le formulaire d'enregistrement et sera mentionné sur le site web listant les lecteurs enregistrés.

Pour mettre en place une interface commune avec les applications, le lecteur doit comporter un ensemble d'API (dans le package d'installation ou directement préinstallé). Certaines de ces API sont propres à une plate-forme (p.ex. Microsoft CryptoAPI), d'autres sont spécifiquement belges (p.ex. eID runtime) - voir le chapitre "Spécifications".

L'instance d'enregistrement des lecteurs se réserve le droit d'adapter le scénario de test à l'évolution technique des interfaces et des normes propres aux plates-formes.

1.5. Environnement

Les impératifs de sécurité de la carte eID étant propres à l'environnement, l'enregistrement fera par ailleurs une distinction entre les lecteurs destinés à un environnement domestique/professionnel/mobile et les lecteurs destinés à un environnement public (voir le chapitre "Impératifs de sécurité").

En effet, les lecteurs à installer dans des endroits publics, sans être soumis au contrôle de citoyens belges, présentent des impératifs de sécurité spéciaux, que l'on peut résumer ainsi :

- disponibilité d'un clavier PIN sécurisé, pour garantir la confidentialité du code PIN

- disponibilité d'un affichage sécurisé pour garantir l'exactitude des messages.

Deux logos " compatible eID belge " distingueront ces deux catégories de lecteurs.

2. Besoins et spécifications

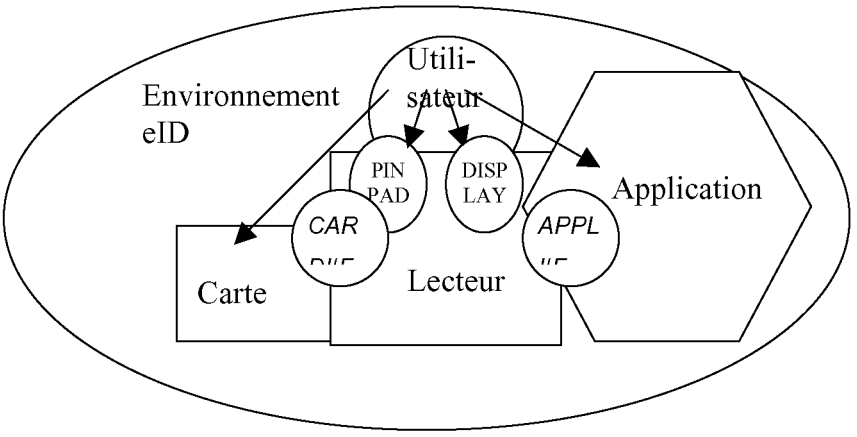
2.1. Besoins et termes de modèle et de sécurité

Un environnement compatible avec la carte eID se compose d'un utilisateur final en interaction avec un système compatible eID. A son tour, le système eID compte trois parties :

- la carte eID elle-même

- le lecteur compatible avec la carte eID

- l'application compatible avec la carte eID.



L'application compatible eID contient tous les composants propres à l'application, nécessaires notamment pour la présentation des documents, la visualisation des données/attributs, la mise en forme de la signature, etc. Plusieurs instances de l'application peuvent être présentes dans la même unité physique et partager le même lecteur.

Le lecteur compatible eID doit comporter tous les composants génériques (matériels et/ou logiciels) nécessaires à l'interfaçage de la carte eID conformément aux impératifs de sécurité décrits dans [CEN/CWA 14170] et traduits en spécifications techniques dans [EID/READERS 2.7.3], [EID/CRYPTO 1.4.0] et [EID/IDENTITY 1.0.0]. En particulier :

- PINPAD (User Input & Authentication Component - composant de saisie utilisateur et authentification) : ce composant assure les fonctions d'authentification sécurisée via lesquelles le code PIN de l'utilisateur est introduit et comparé au code PIN contenu dans la carte. L'on distingue généralement les lecteurs munis d'un clavier PIN spécifique et ceux qui font usage du clavier générique (numpad). Ce composant doit être conforme aux impératifs de sécurité de [CEN/CWA 14170 - Chapitre 13], traduits en spécifications techniques dans [EID/READERS 2.7.3 - Chapitres 2 & 3].

- DISPLAY (User Output & Control Component - composant d'affichage utilisateur et contrôle) : ce composant se charge des interactions sécurisées entre l'utilisateur et l'application qui permettent au premier de contrôler l'exécution et qui renvoient codes d'erreur et messages d'état. L'on distingue généralement les lecteurs munis d'un afficheur dédié et ceux qui font usage de l'écran générique. Ce composant doit être conforme aux impératifs de sécurité de [CEN/CWA 14170 - Chapitre 12], traduits en spécifications techniques dans [EID/READERS 2.7.3 - Chapitres 2 & 3].

- APPLI/F (Application Input/Output Interface Component - composant d'interface d'entrée/de sortie de l'application) : ce composant met en place les fonctions de manipulation sécurisée des données qui permettent à l'application d'interagir avec le lecteur eID pour envoyer et recevoir les données de façon normalisée et sécurisée. Il constitue généralement une interface générique avec l'application. Il doit être conforme aux impératifs de sécurité de [CEN/CWA 14170 - Chapitre 15], traduits en spécifications techniques dans [EID/CRYPTO 1.4.0] et [EID/IDENTITY 1.0.0].

- CARDI/F (Card Input/Output Interface Component - composant d'interface d'entrée/de sortie de la carte) : ce composant met en place les fonctions d'interaction qui permettent à l'application d'échanger [ISO/IEC 7816] des commandes avec la carte. Il doit être conforme aux impératifs de sécurité de [CEN/CWA 14170 - Chapitre 16], traduits en spécifications techniques dans [EID/READERS 2.7.3 - Chapitres 1 et 2.1].

2.2 Besoins en termes d'interface carte

L'interface carte [ISO7816] sera conforme aux paramètres des spécifications [EID/READER 2.7.3, chapitres 1er et 2.1].

Elle respectera aussi :

- les impératifs de sécurité décrits dans [EN60950]

- les critères génériques d'émissions électromagnétiques décrits dans [EN50081]

- les critères génériques d'immunité électromagnétique décrits dans [EN50081]

- les critères génériques de perturbations radio-électriques décrits dans [EN55022].

Note : si le lecteur possède une interface à deux slots, il devra être conforme aux spécifications des lecteurs de cartes SIS/SAM (voir www.ksz-bcss.fgov.be).

2.3 Besoins en termes d'interface utilisateur

Le processus d'interaction avec l'utilisateur peut trouver place dans deux types d'environnement physique, où le système eID est contrôlé par des organismes différents :

1. Lieu public : le système eID est installé dans un lieu public (gare, agence de banque, bureau de poste ou tout autre endroit exploité par un prestataire de services qui n'est pas nécessairement lié à l'utilisateur final ou sous son contrôle). Sans autres mesures de sécurité techniques, ce type d'environnement est exposé à une série de risques (p.ex. remplacement par un système eID contrefait). En conséquence, dans les lieux publics, les impératifs techniques des systèmes eID seront nécessairement plus stricts.

2. Domicile et lieu de travail : le système eID est installé dans un domicile ou sur un lieu de travail, généralement sous le contrôle direct de l'utilisateur (p.ex. téléphone mobile). Dans ce cas, les impératifs de sécurité peuvent être respectés par des méthodes d'organisation mises en place par l'utilisateur. Les moyens techniques nécessaires pour répondre aux impératifs de sécurité peuvent être moins stricts.

Selon son utilisation, le lecteur de carte devra se conformer à des besoins et spécifications différents, résumés ci-dessous :

- les lecteurs installés dans les lieux publics devront posséder un clavier PIN et un afficheur dédiés et sécurisés, intégrés dans l'appareil (clavier et écran standard ne peuvent servir aux interactions avec l'utilisateur)

les lecteurs installés dans un environnement domestique ou professionnel peuvent faire appel au clavier et à l'écran génériques pour les interactions avec l'utilisateur.

2.4 Besoins en termes d'interface application

Le gouvernement fédéral a développé un environnement runtime eID qui est mis gratuitement à la disposition de tout utilisateur final. Il constitue un ensemble de bibliothèques, outils et exécutables permettant de donner une interface générique aux applications qui doivent communiquer/s'interfacer avec une carte eID.

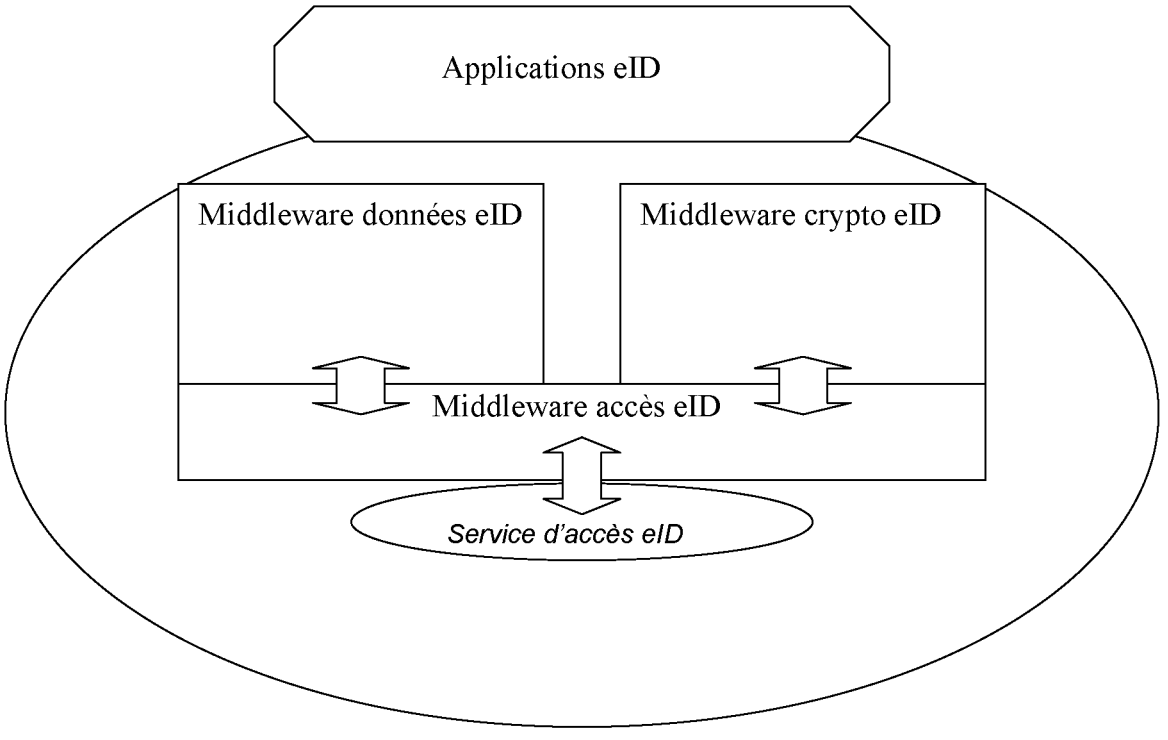
Cet environnement regroupe quatre composants :

- middleware (intergiciel) données eID : présente une interface d'extraction de données standardisée aux applications, pour leur permettre d'exploiter les fonctions de capture de données de la carte.

- middleware cryptographique eID : présente une interface de cryptographie standardisée aux applications, pour leur permettre d'exploiter les fonctions d'authentification et de signature de la carte.

- middleware d'accès eID : utilisé par les deux autres pour se connecter et communiquer de façon sécurisée avec un service d'accès eID

- le service d'accès eID utilisé par un middleware d'accès eID pour la connexion au lecteur physique (il peut s'agir d'un simple câble ou d'un réseau).



L'environnement runtime eID est naturellement lié à la plate-forme. Une version est disponible pour chaque plate-forme supportée. Plusieurs releases de l'environnement eID seront prévus pour suivre les releases de la carte eID dans le temps.

Note : selon la plate-forme, le middleware cryptographique pourra posséder plusieurs interfaces pour répondre aux impératifs propres au fournisseur :

- Linux : seul PKCS#11 est requis

- Microsoft : PKCS#11 et Microsoft CryptoAPI sont requis

- MacOS : PKCS#11 et Apple CryptoAPI sont requis.

Le gouvernement belge considère l'environnement runtime eID comme remplissant tous les impératifs à respecter par le composant interface d'application. De plus, l'environnement runtime eID porte une signature codée du gouvernement belge, qui garantit la diffusion des exemplaires légitimes uniquement. Un kit de développement logiciel est aussi disponible gratuitement, avec des exemples et des directives concernant l'interface avec l'environnement runtime eID.

L'environnement runtime eID étant considéré comme faisant partie du lecteur, il existe deux alternatives :

- le lecteur eID est acquis par l'utilisateur final en tant que composant séparé (p.ex. lecteur connectable); il doit alors inclure les outils d'installation du runtime eID et les guides d'installation/utilisation livrés avec le lecteur. Le package d'installation doit intégrer à la fois les pilotes propres au lecteur et l'environnement runtime eID (la partie concernant le runtime eID peut être rendue facultative pour que l'utilisateur puisse décider de l'installer ou non).

- le lecteur eID est acquis incorporé dans un autre produit (p.ex. un PC comprenant un lecteur de carte à puce). Dans ce cas, le runtime eID doit être préinstallé. Le package d'installation eID et les guides d'installation/utilisation doivent aussi être fournis (dans la perspective d'une réinstallation).

Note : s'il n'existe pas d'environnement runtime pour un lecteur particulier (plate-forme non supportée et/ou middleware identité/crypto directement mis en œuvre dans le firmware de l'appareil), le demandeur a le droit de développer le sien (ou d'adapter la version officielle à son environnement propriétaire). Dans un tel cas, l'instance d'enregistrement des lecteurs se réserve le droit d'évaluer la conformité et l'équivalence avec l'environnement runtime eID officiel.

2.5 Besoins spécifiques

L'utilisateur final doit pouvoir installer/réinstaller/supprimer/mettre à niveau l'environnement runtime eID quand une nouvelle version est publiée sur le site web eID. Pour cette raison, la présente procédure d'enregistrement ne prendra pas en charge les environnements runtime de tiers ni les environnements runtime modifiés, sauf accord contractuel avec l'instance d'enregistrement des lecteurs.

Après installation, l'utilisateur final doit pouvoir déconnecter/reconnecter physiquement le lecteur (mode plug and play) sans devoir redémarrer ou réamorcer le système. Cette exigence ne concerne que les lecteurs connectables utilisés dans le contexte domestique ou professionnel.

Il doit être facile de déterminer si le lecteur a été enregistré conformément à la procédure décrite dans ce document (p.ex. via un autocollant). L'on doit également pouvoir connaître sans difficulté la marque et le modèle de lecteur, surtout s'il est incorporé dans un autre appareil. La marque et le type doivent figurer clairement dans la documentation et sur les parties visibles de l'unité.

3. Scénarios de test de bas niveau

Ces ensembles de scénarios seront exécutés par chaque fournisseur de lecteur eID pour établir la compatibilité de son produit avec la carte eID. Ils sont organisés en quatre parties, suivant la plate-forme :

- La première partie se compose d'un jeu de commandes ISO7816 à exécuter et tester dans l'appel de fonction SendAPDU de l'environnement runtime eID.

- La deuxième partie se compose d'un jeu de fonctions Data Capture à exécuter et tester dans les appels de fonction GetXXX de l'environnement runtime eID.

- La troisième partie se compose d'un jeu de fonctions crypto à exécuter et tester dans le middleware crypto de l'environnement runtime eID (mise en œuvre PKCS#11).

- La quatrième partie (nécessaire seulement sur les plates-formes Microsoft et Apple) se compose d'un jeu de fonctions crypto à exécuter et tester dans le middleware crypto de l'environnement runtime eID (mise en œuvre CryptoAPI).

Les spécifications du contenu de la carte et des interfaces se trouvent dans :

- [EID/CRYPTO 1.4.0] pour les éléments cryptographiques (clés, certificats, PIN, etc.) et les spécifications du middleware crypto

- [EID/IDENTITY 1.0.0] pour les données (identité, adresse, photo) et spécifications du middleware données

- [EID/CARD 2.0.0] pour les commandes ISO7816 supportées et les spécifications de l'interface de carte eID.

3.1 Scénario ISO7816/APDU

Conditions préalables :

- Lecteur de carte à puce correctement installé et configuré

- Bibliothèque d'identité correctement installée et configurée

- Carte de test insérée

```
long handle = 0;
BEID_Pin pin = {0};
   pin.id = 0x01;
   pin.pinType = BEID_PIN_TYPE_PKCS15;
   pin.usageCode = BEID_USAGE_AUTH;
BEID_Bytes sendBytes = {(unsigned char *)"\x00\xA4\x08\x0C\x06\x3F\x00\xDF\x01\x40\x32", 11};
BEID_Bytes respBytes = {0};
…
assert (BEID_Init( NULL, 0, 0, &handle ).general==BEID_OK);                    // Init
assert (BEID_BeginTransaction().general==BEID_OK);                             // Begin transaction
assert (BEID_SendAPDU( &sendBytes, &pin, &respBytes ).general==BEID_OK); // Select SGNID
assert (BEID_EndTransaction().general==BEID_OK);                               // End transaction
assert (BEID_FlushCache().general==BEID_OK);                                   // Flush Cache
assert (BEID_Exit().general==BEID_OK);                                         // Exit
```

Conditions postérieures :

- assertion entièrement réussie

- les informations suivantes doivent figurer dans le formulaire d'enregistrement :

réponse de la carte au SGNID sélection.

3.2 Scénario Data Capture

3.2.1 Version carte et certificat RRN

Conditions préalables :

- Lecteur de carte à puce correctement installé et configuré

- Bibliothèque d'identité correctement installée et configurée

- Carte de test insérée

```
long handle = 0;
BEID_VersionInfo    version = {0};
BEID_Bytes          certRRN = {0};
BEID_Bytes          bytesBuffer = {0};
BEID_Bytes          tAID = { (unsigned char *)"\xA0\x00\x00\x01\x77\x50\x4B\x43\x53\x2D\x31\x35", 12 };
BEID_Bytes          tFileID = { (unsigned char *)"\x50\x3C", 2 };                    // RRN
…
assert (BEID_Init( NULL, 0, 0, &handle ).general==BEID_OK);                          // Init
assert (BEID_GetVersionInfo( &version, 0, &bytesBuffer ).general==BEID_OK); // Read Version
assert (BEID_SelectApplication( &tAID ).general==BEID_OK);                           // Select application
assert (BEID_ReadFile( &tFileID, &certRRN, 0x00 ).general==BEID_OK);         // Read RRN Cert
assert (BEID_Exit().general==BEID_OK);                                               // Exit
```

Conditions postérieures :

- assertion entièrement réussie

- les informations suivantes doivent figurer dans le formulaire d'enregistrement :

- version (numéro de série, code composant, numéro et version système d'exploitation, numéro et version softmask, version applet, version système d'exploitation global, version interface applet, support PKCS#1, version échange de clés, cycle de vie application, personnalisation graphique, personnalisation électronique, interface personnalisation électronique)

- Certificat RRN.

3.2.2 Identité, adresse et photo

Conditions préalables :

- Lecteur de carte à puce correctement installé et configuré

- Bibliothèque d'identité correctement installée et configurée

- Carte de test insérée

```
long handle = 0;
BEID_ID_Data        identityData = {0};
BEID_Address        addressData = {0};
BEID_Certif_Check   certifCheck = {0};
BEID_Bytes          pictureData = {0};
…
assert (BEID_Init( NULL, 0, 0, &handle ).general==BEID_OK);                          // Init
assert (BEID_GetID(&identityData, &certifCheck).general==BEID_OK);           // Read Identity Data
assert (BEID_GetAddress(&addressData, &certifCheck).general==BEID_OK);       // Read Address Data
assert (BEID_GetPicture( &pictureData, &certifCheck ).general==BEID_OK);     // Read Picture Data
assert (BEID_Exit().general==BEID_OK);                                               // Exit
```

Conditions postérieures :

- assertion entièrement réussie

- les informations suivantes doivent figurer dans le formulaire d'enregistrement :

- identité (numéro carte et puce, validité, commune, NRN, prénoms et nom de famille, lieu et date de naissance, nationalité, sexe, titre de noblesse)

- adresse (rue, numéro, boîte, code postal, localité, pays)

3.2.3 Contrôle PIN

Conditions préalables :

- Lecteur de carte à puce correctement installé et configuré

- Bibliothèque d'identité correctement installée et configurée

- Carte de test insérée

```
long handle = 0;
long triesLeft = 0;
BEID_Pin pin = {0};
   pin.id = 0x01;
   pin.pinType = BEID_PIN_TYPE_PKCS15;
   pin.usageCode = BEID_USAGE_AUTH;
…
assert (BEID_Init( NULL, 0, 0, &handle ).general==BEID_OK);                          // Init
assert (BEID_VerifyPIN(&pin, NULL, &triesLeft ).general==BEID_OK);           // Verify PIN
assert (BEID_Exit().general==BEID_OK);                                               // Exit
```

Conditions postérieures :

- assertion entièrement réussie
- les informations suivantes doivent figurer dans le formulaire d'enregistrement :
- si le lecteur possède un clavier PIN sécurisé, celui-ci doit servir à introduire le PIN

s'il s'agit d'un lecteur sans clavier PIN sécurisé, le PIN doit être introduit à l'écran.

3.3 Scénario Cryptoki/PKCS#11

3.3.1 Initialisation bibliothèque, slot et token

Conditions préalables :

- Lecteur de carte à puce correctement installé et configuré
- Bibliothèque Cryptoki/PKCS#11 correctement installée et configurée
- Carte de test insérée

```
CK_INFO             libInfo;                                    // PKCS#11 / Cryptoki Library Information
CK_ULONG                    slotCount;                          // Number of Slots available
CK_SLOT_ID_PTR      pSlotList;                          // List of Slots available
CK_SLOT_ID                  slotID;                                     // Identifier of Slot to be tested
CK_SLOT_INFO                slotInfo;                                   // Slot Information
CK_TOKEN_INFO       tokenInfo;                          // Token Information
…
assert (C_Initialize(NULL)==CKR_OK);                    // initialize the PKCS#11 / Cryptoki Library
assert (C_GetInfo(&libInfo)==CKR_OK);                   // Get PKCS#11 / Cryptoki Library Information
assert (C_GetSlotList(CK_FALSE, NULL_PTR, &slotCount)==CKR_OK); // Get List of Slots / Readers
pSlotList = (CK_SLOT_ID_PTR) malloc(slotCount*sizeof(CK_SLOT_ID));
assert (C_GetSlotList(CK_TRUE, pSlotList, &slotCount)==CKR_OK);   // Get List of Slots with token
slotID = pSlotList[0];                                  // Slot to be tested (with card inserted)
assert (C_GetSlotInfo(slotID, &slotInfo)==CKR_OK);      // Get selected slot information
assert (C_GetTokenInfo(slotID, &tokenInfo)==CKR_OK);  // Get selected token information
…
```

Conditions postérieures :

- assertion entièrement réussie
- les informations suivantes doivent figurer dans le formulaire d'enregistrement :
- bibliothèque (version cryptoki, id. fabricant, version et description bibliothèque)
- slot/lecteur (description slot, id. fabricant, version matériel, version firmware)
- token/carte (label et modèle token, id. fabricant, numéro de série, version matériel, version firmware).

3.3.2. Déclenchement signature avec clé d'authentification

Conditions préalables : test précédent réussi (initialisation bibliothèque, slot et token)

```
CK_SESSION_HANDLE       hSession;
CK_OBJECT_HANDLE        hObject, hAuthKey, hAuthCert, hIssuerCert, hRootCert;
CK_ATTRIBUTE            authKeyTemplate[], authCertTemplate[], issuerCertTemplate[], rootCertTemplate[];
CK_BYTE                 application, digest[20], signature[128], data[] = {'S','A','M','P','L','E'};
CK_ULONG                objectCount, digestLen, signatureLen;
CK_MECHANISM            mecHash = {CKM_SHA_1,NULL_PTR,0},
                mecSign = {CKM_SHA1_RSA_PKCS,NULL_PTR,0};
…
assert (C_OpenSession(slotID, CKF_SERIAL_SESSION,               // open authentication session
                (CK_VOID_PTR) &application, NULL_PTR,&hSession)==CKR_OK);
assert (C_FindObjectsInit(hSession, NULL_PTR, 0)==CKR_OK);  // find certificates and keys
while (C_FindObjects(hSession, &hObject, 1, &ObjectCount)==CKR_OK) {
        if (ObjectCount == 0) break;
        if (C_GetAttributeValue(hSession, hObject, authKeyTemplate, 1)==CKR_OK) hAuthKey=hObject;
        if (C_GetAttributeValue(hSession, hObject, authCertTemplate, 1)==CKR_OK) hAuthCert=hObject;
        if (C_GetAttributeValue(hSession, hObject, issuerCertTemplate, 1)==CKR_OK) hIssuerCert=hObject;
        if (C_GetAttributeValue(hSession, hObject, rootCertTemplate, 1)==CKR_OK) hRootCert=hObject;
        }
assert (C_FindObjectsFinal(hSession)==CKR_OK);
if (C_DigestInit(hSession, &mecHash)==CKR_OK) {                 // Compute Message Digest
        if (C_DigestUpdate(hSession, data, sizeof(data))==CKR_OK) {
                digestLen = sizeof(digest);
                assert (C_DigestFinal(hSession, digest, &digestLen)==CKR_OK);
        }
if (C_SignInit(hSession, &mecSign, hObject)==CKR_OK) {          // compute Signature
        if (C_SignUpdate(hSession, digest, sizeof(digest))==CKR_OK) {
                signatureLen = sizeof(signature);
                assert (C_SignFinal(hSession, signature, &signatureLen)==CKR_OK);
                }
        }
```

Conditions postérieures :

- assertion entièrement réussie
- les informations suivantes doivent figurer dans le formulaire d'enregistrement : certificats, résumé et signature

3.3.3 Modifier PIN

Conditions préalables : test précédent réussi (session initialisée)

```
CK_SESSION_HANDLE        hSession;
…
CK_UTF8CHAR oldPin[] = {'1','2','3','4'};
CK_UTF8CHAR newPin[] = {'6','5','4','3','2','1'};
…
assert (C_SetPIN(hSession, oldPin, sizeof(oldPin), newPin, sizeof(newPin))==CKR_OK);

assert (C_CloseSession(hSession)==CKR_OK);                    // close authentication session
```

Conditions postérieures :

assertion entièrement réussie

3.3.4 Déclenchement signature avec clé de signature

Conditions préalables : test précédent réussi (initialisation bibliothèque, slot et token)

```
CK_SESSION_HANDLE        hSession;
CK_OBJECT_HANDLE         hObject, hSignKey, hSignCert, hIssuerCert, hRootCert;
CK_ATTRIBUTE             signKeyTemplate[], signCertTemplate[], issuerCertTemplate[], rootCertTemplate[];
CK_BYTE                  application, digest[20], signature[128], data[] = {'S','A','M','P','L','E'};
CK_ULONG                 objectCount, digestLen, signatureLen;
CK_MECHANISM             mecHash = {CKM_SHA_1,NULL_PTR,0},
                mecSign = {CKM_SHA1_RSA_PKCS,NULL_PTR,0};
…
assert (C_OpenSession(slotID, CKF_SERIAL_SESSION,             // open signature session
                (CK_VOID_PTR) &application, NULL_PTR,&hSession)==CKR_OK);
assert (C_FindObjectsInit(hSession, NULL_PTR, 0)==CKR_OK);  // find certificates and keys
while (C_FindObjects(hSession, &hObject, 1, &ObjectCount)==CKR_OK) {
        if (ObjectCount == 0) break;
        if (C_GetAttributeValue(hSession, hObject, signKeyTemplate, 1)==CKR_OK) hSignKey=hObject;
        if (C_GetAttributeValue(hSession, hObject, signCertTemplate, 1)==CKR_OK) hSignCert=hObject;
        if (C_GetAttributeValue(hSession, hObject, issuerCertTemplate, 1)==CKR_OK) hIssuerCert=hObject;
        if (C_GetAttributeValue(hSession, hObject, rootCertTemplate, 1)==CKR_OK) hRootCert=hObject;
        }
assert (C_FindObjectsFinal(hSession)==CKR_OK);
if (C_DigestInit(hSession, &mecHash)==CKR_OK) {               // Compute Message Digest
        if (C_DigestUpdate(hSession, data, sizeof(data))==CKR_OK) {
                digestLen = sizeof(digest);
                assert (C_DigestFinal(hSession, digest, &digestLen)==CKR_OK);
        }
if (C_SignInit(hSession, &mecSign, hObject)==CKR_OK) {        // compute Signature
        if (C_SignUpdate(hSession, digest, sizeof(digest))==CKR_OK) {
                signatureLen = sizeof(signature);
                assert (C_SignFinal(hSession, signature, &signatureLen)==CKR_OK);
                }
        }
```

Conditions postérieures :

- assertion entièrement réussie
- les informations suivantes doivent figurer dans le formulaire d'enregistrement : certificats, résumé et signature

3.3.5 Modifier PIN

Conditions préalables : test précédent réussi (session initialisée)

```
CK_SESSION_HANDLE        hSession;
…
CK_UTF8CHAR oldPin[] = {'6','5','4','3','2','1'};
CK_UTF8CHAR newPin[] = {'1','2','3','4'};
…
assert (C_SetPIN(hSession, oldPin, sizeof(oldPin), newPin, sizeof(newPin))==CKR_OK);

assert (C_CloseSession(hSession)==CKR_OK);                    // close signature session
```

Conditions postérieures :

- assertion entièrement réussie

4. Scénarios de validation de haut niveau

Ces ensembles de scénarios seront exécutés par chaque fournisseur de lecteur eID pour établir la compatibilité de son produit avec la carte eID. Ils sont organisés en quatre parties à exécuter l'une après l'autre (chaque partie réutilise les résultats de la précédente) :

- Installation, désinstallation, mise à jour, reconfiguration
- Authentification forte
- Capture de données et modification PIN
- Signature qualifiée.

Pour chaque scénario de validation, un ensemble de conditions préalables, objectifs de contrôle et conditions postérieures est décrit dans le but d'évaluer la compatibilité du lecteur et des logiciels correspondants.

Note : certains de ces scénarios peuvent ne pas s'appliquer, en fonction des circonstances suivantes :

- conditionnement du lecteur (p.ex. la désinstallation/réinstallation ne s'applique évidemment pas aux lecteurs intégrés à un PC)
- Plug&Play (p.ex. la déconnexion/reconnexion ne s'applique évidemment pas aux lecteurs incorporés dans un autre élément matériel comme un clavier).
- CryptoAPI (p.ex. le support d'un middleware propriétaire/propre à une plate-forme ne concerne que la plate-forme en question).

4.1 Scénario I : installation/désinstallation et Plug&Play

4.1.1. Conditions préalables

* Carte de test activée disponible (avec codes PIN connus)
* PC local préinstallé (avec plate-forme/système d'exploitation cible), navigateur Internet, pilotes du lecteur et environnement runtime eID appropriés
* Connexion Internet configurée avec les ports 80 (http) et 443 (https) disponibles.
* Guide d'utilisation/installation du lecteur disponible.

4.1.2. Objectifs de contrôle

* Installer le lecteur de carte suivant les instructions du guide d'utilisation (p.ex. redémarrage, droits d'administration, etc.)
* [le cas échéant] Désinstaller et réinstaller le lecteur de carte suivant les instructions du guide d'utilisation (p.ex. redémarrage, droits d'administration, etc.)
* [le cas échéant] Débrancher et rebrancher le lecteur de carte
* Validation capture de données eID

==> OC1 : Lancer l'application de capture de données
==> OC3 : Insérer la carte eID
==> OC2 : Capturer les données
==> OC5 : Retirer la carte eID
==> OC6 : Capturer les données
==> OC7 : Insérer la carte eID (sur demande)

* [le cas échéant] Installer un deuxième lecteur de carte du même modèle
* [le cas échéant] Débrancher et rebrancher le deuxième lecteur de carte
* Validation de la capture de données eID (à exécuter sur les deux lecteurs)

==> OC1 : Lancer l'application de capture de données
==> OC3 : Insérer la carte eID
==> OC2 : Capturer les données
==> OC5 : Retirer la carte eID
==> OC6 : Capturer les données
==> OC7 : Insérer la carte eID (sur demande)

4.1.3. Conditions postérieures

* Ensemble de numéros de version de configuration (système d'exploitation/lecteur/runtime, etc.)
* Images d'écran de l'outil de capture de données.

4.2. Scénario II : authentification forte

4.2.1. Conditions prélables

* Scénario I réussi
* Site de validation/test eID disponible
* Navigateurs Internet préinstallés (compatibles SSLv3, avec interface PKCS#11)
* [le cas échéant] Navigateurs Internet préinstallés (compatibles SSLv3, avec interface CryptoAPI)
* Scénario II séléctionné

4.2.2. Objectifs de contrôle

* Connexion au service de test via https avec navigateur compatible PKCS#11 :

==> OC1 : configurer le navigateur pour reconnaître l'interface eID PKCS#11
==> OC2 : Serveur authentifié
==> OC3 : Certificat client "authentification" sélectionné
==> OC4 : Code PIN demandé
==> OC5 : Authentification réussie

* [le cas échéant] Connexion au service de test via https avec navigateur compatible CryptoAPI :

==> OC1 : configurer le navigateur en chargeant les certificats de l'utilisateur final dans le dépôt de certificats
==> OC2 : Serveur authentifié
==> OC3 : Certificat client "authentification" sélectionné
==> OC4 : Code PIN demandé
==> OC5 : Authentification réussie

4.2.3. Conditiions postérieures

* Données enregistrées avec succès sur le site de validation (système d'exploitation/navigateur/interface/carte/etc.).

* Les messages apparaissant sur l'afficheur doivent être signalés.

* Numéro de carte, données et heure approximative de la transaction avec le site de validation doivent être signalés.

* Image d'écran de la dernière page du site de test eID.

4.3. Scénario III : capture de données et modification PIN

4.3.1. Conditions préalables

* Scénario II réussi

* Connexion sécurisée établie avec le site de validation

* Scénario III sélectionné.

4.3.2. Objectifs de contrôle

* Connexion au service de test via https avec navigateur compatible PKCS#11 :

==> OC1 : Serveur authentifié

==> OC2 : Certificat client "authentification" sélectionné

==> OC3 : Code PIN demandé

==> OC4 : Authentification réussie

==> OC5 : Capture de données appelée automatiquement

==> OC6 : Affichage Capture de données

==> OC7 : Modification code PIN

==> OC8 : Modification réussie

* [le cas échéant] Connexion au service de test via https avec navigateur compatible CryptoAPI :

==> OC1 : Serveur authentifié

==> OC2 : Certificat client "authentification" sélectionné

==> OC3 : Code PIN demandé

==> OC4 : Authentification réussie

==> OC5 : Capture de données appelée automatiquement

==> OC6 : Affichage Capture de données

==> OC7 : Modification code PIN

==> OC8 : Modification réussie

4.3.3. Conditions postérieures

* Données enregistrées avec succès sur le site de validation (système d'exploitation/navigateur/interface/carte/etc.).

* Les messages apparaissant sur l'afficheur doivent être signalés.

* Numéro de carte, données et heure approximative de la transaction avec le site de validation doivent être signalés.

* Image d'écran de la dernière page du site de test eID.

4.4. Scénario IV : signature éléctronique

4.4.1. Conditions préalables

* Scénario III réussi

* Connexion sécurisée établie avec le site de validation

* Scénario IV selectionné

4.4.2. Objectifs de contrôle

* Connexion au service de test via https avec navigateur compatible PKCS#11 :

==> OC1 : Serveur authentifié

==> OC2 : Certificat client "authentification" sélectionné

==> OC3 : Code PIN demandé

==> OC4 : Authentification réussie

==> OC5 : Signature électronique appelée automatiquement

==> OC6 : Certificat client "signature" sélectionné

==> OC7 : Code PIN demandé

==> OC8 : Signature réussie

* [le cas échéant] Connexion au service de test via https avec navigateur compatible CryptoAPI :

==> OC1 : Serveur authentifié

==> OC2 : Certificat client "authentification" sélectionné

==> OC3 : Code PIN demandé

==> OC4 : Authentification réussie

==> OC5 : Signature électronique appelée automatiquement

==> OC6 : Certificat client "signature" sélectionné

==> OC7 : Code PIN demandé

==> OC8 : Signature réussie

4.4.3.Conditions postérieures

* Données enregistrées avec succès sur le site de validation (système d'exploitation/navigateur/interface/carte/etc.).

* Les messages apparaissant sur l'afficheur doivent être signalés.

* Numéro de carte, données et heure approximative de la transaction avec le site de validation doivent être signalés.

* Image d'écran de la dernière page du site de test eID.

Art.2. Spécifications des lecteurs

1. Compatibilité matérielle avec la puce

Le lecteur doit être conforme aux normes ou recommandations suivantes :

□ ISO/IEC 7816-1 : caractéristiques physiques

□ ISO/IEC 7816-2 : dimensions et emplacement des contacts

□ ISO/IEC 7816-3 : signaux électroniques et protocoles de communication

□ Format de carte ID1

□ Protocole asynchrone T=0 (T=1 facultatif)

□ Tension électrique VCC = 5V,5 % - maximum 50 mA

□ Tension de programmation VPP égale à VCC, 5 %

□ Fréquence d'horloge initiale de 3,5712 MHz (9.600 bps). La fréquence d'horloge de travail est égale ou supérieure à la fréquence d'horloge initiale.

□ Contacts :

— pression de contact (1 N

— résistance de contact (0,3 Ohm

— force d'insertion de la carte (12 N

— force d'extraction de la carte (1 N

2. Compatibilité logicielle avec la puce

2.1. Tous les lecteurs

□ Tous les lecteurs doivent être conformes aux normes ISO/IEC 7816-4 et 7816-8 de format d'échange de commandes.

2.2. Lecteurs avec clavier PIN

Les lecteurs avec clavier PIN doivent mettre en œuvre toutes les commandes ISO 7816-4 et 7816-8 (APDU) qui nécessitent la saisie du PIN mais sans le communiquer à l'extérieur du lecteur.

2.2.1 Commandes de carte (APDU) à mettre en œuvre

□ ***PIN Verify***

Champ	**Valeur**
CLA	'00'
INS	'20'
P1	'00'
P2	Référence PIN
Lc	Longueur des données de vérification
Data	Données de vérification
Le	Vide

□ ***PIN Change (Change Reference Data)***

Champ	Valeur
CLA	'00'
INS	'24'
P1	'00' (Utilisateur)
P2	Référence PIN
Lc	Longueur du champ de données suivant
Data	PIN existant || Nouveau PIN (concaténation)
Le	Vide

2.2.2. Commandes de lecteur (APDU) à mettre en œuvre

Cette section décrit les commandes APDU que le lecteur doit mettre en œuvre pour interagir avec les commandes PIN émanant de la carte. Si l'accès au lecteur passe par le middleware standard eID, une interface propriétaire peut être utilisée, pourvu que le fabricant fournisse la DLL qui met en œuvre la bonne interface (voir section 3.2). En revanche, si l'accès au lecteur passe par une autre voie (p.ex. application sur mesure), les commandes proposées doivent être accessibles aux concepteurs d'applications.

Bien que la dernière solution puisse fonctionner sur le plan technique, elle est très limitée et pourrait se voir refusée par certaines institutions ou instances d'agrément officielles.

Les deux commandes décrites dans la section 2.2.1 doivent être appelées automatiquement par la demande d'un PIN (ou 2 PIN) sur le clavier PIN lorsque le paramètre Lc des commandes ci-dessus est égal à 0. Dans ce cas, le firmware du lecteur doit d'abord lire le PIN introduit sur le clavier PIN, puis remplacer les paramètres Lc et Data par les données correctes, générées à partir du PIN introduit.

□ ***Reader PIN Verify***

Champ	Valeur
CLA	’00’
INS	’20’
P1	’00’
P2	Référence PIN
Lc	’00’
Data	Vide
Le	Vide

□ ***Reader PIN Change***

Champ	Valeur
CLA	’00’
INS	’24’
P1	’00’ (Utilisateur)
P2	Référence PIN
Lc	’00’
Data	Vide
Le	Vide

Pour PIN Change, le nouveau PIN doit être introduit deux fois (avec comparaison) pour éviter les erreurs.

Le code renvoyé (octet d’état) doit être celui qui provient de la commande de carte, défini dans les normes 7816-4 et 7816-8, ou un des suivants :

Octet d’état	Signification
’ECD2’	Expiration du délai d’introduction du PIN
’ECD6’	Annulé par l’utilisateur
’ECB6’	Pas de diagnostic précis

2.2.3. Format PIN

Le PIN est une chaîne (string) de 8 octets au format suivant (par quartet), défini dans la section Global PIN du document ”Global Platform - Card Specification - version 2.0.1’ - April 7, 2000” :

C	L	P	P	P	P	P/’F’	P/’F’	P/’F’	P/’F’	P/’F’	P/’F’	P/’F’	P/’F’	’F’	’F’

Quartet	Signification
C	Paramètre de contrôle, contient toujours ’2’
L	Longueur du PIN (en quartets) - de ’4’ à ’C’
P	Chiffres PIN (minimum 4)
P/’F’	Le reste des chiffres du PIN suivant la longueur, sinon ’F’
’F’	Remplissage - toujours ’F’

3. Compatibilité logicielle avec le middleware

Cette section s’applique aux lecteurs reliés à un ordinateur et qui utiliseront la carte via l’interface Microsoft CryptoAPI ou l’interface PKCS#11.

3.1. Pilotes PC/SC

Tous les lecteurs doivent s’accompagner d’un pilote conforme PC/SC version 1.1 pour le système d’exploitation cible.

3.2. Intégration au middleware

En vue de l’intégration au middleware, le fournisseur du lecteur doit développer pour chaque système d’exploitation cible une Dynamic Linked Library (ou équivalent) mettant en œuvre les fonctions suivantes :

* SCR_Init ()

* SCR_VerifyPIN ()

* SCR_ChangePIN ()

Actuellement, seule l’application Belgian Identity (ci-après appelée ID) réside sur la carte. Dans la perspective de futures applications (revêtant la forme de fichiers de données ou de répertoires sur la carte), certaines fonctions possèdent un paramètre ApplicationID qui contient une chaîne identifiant l’application appelée à interagir avec un des PIN. Le but est que le lecteur affiche (voir 3.3) l’application qui demande un PIN.

3.2.1. SCR_Init ( )

Cette fonction est appelée tout de suite après le chargement de la DLL. Elle sert à initialiser la DLL et à vérifier si elle supporte le lecteur connecté.

Paramètres	in/out	Type	Description
reader	in	SCARDHANDLE	Handle vers le lecteur
language	in	ushort	Langue d'affichage : SCR_LG_FRENCH SCR_LG_DUTCH SCR_LG_GERMAN SCR_LG_ENGLISH
supported	out	bool	Booléen : 'Vrai' si le lecteur est supporté par la DLL 'Faux' si le lecteur n'est pas supporté par la DLL Cela permet d'essayer dynamiquement toutes les DLL enregistrées pour trouver celle qui correspond au lecteur (mécanisme " plug and play ").

3.2.2. SCR_VerifyPIN ( )

Cette fonction vérifie le PIN introduit par un utilisateur au clavier PIN du lecteur.

Paramètres	in/out	Type	Description
PINID	in	ushort	Identificateur PIN de la carte - fourni par le middleware. A spécifier comme paramètre P2 dans la fonction Reader PIN Verify
Usage	in	char	Raisons de l'introduction du PIN : 'A' : Authentification 'S' : Signature (non-répudiation) 'E' : Encryption (cryptage) 'P' : Changement de préférence 'M' : Maintenance (administration)
ApplicationID	in	char*	Chaîne (ASCII 7 bits) identifiant l'application propriétaire du PIN (max. 3 caractères). A présenter sur l'afficheur du lecteur.

3.2.3. SCR_ChangePIN ( )

Cette fonction remplace le PIN introduit par un utilisateur au clavier PIN du lecteur par un nouveau, également introduit au clavier PIN.

Paramètres	in/out	Type	Description
PINID	in	ushort	Identificateur PIN de la carte.
			A spécifier comme paramètre P2 dans la fonction Reader PIN Change
ApplicationID	in	char*	Chaîne (ASCII 7 bits) identifiant l'application propriétaire du PIN (max. 3 caractères).
			A présenter sur l'afficheur du lecteur.

3.3. Afficheur

Les paramètres Usage et ApplicationID sont des informations importantes pour le citoyen.

Usage indique pourquoi le citoyen est invité à introduire son PIN (pour s'authentifier, pour signer un document, etc.). Voici la signification de chaque possibilité :

□ 'A' : Authentification (log-on)

□ 'S' : Signature (non-répudiation)

□ 'E' : Encryption (cryptage)

□ 'P' : Modification fichier de préférences

□ 'M' : Maintenance (administration)

ApplicationID est une chaîne identifiant l'application. Actuellement, seule l'application Belgian Identity (ID) réside sur la carte, mais d'autres pourraient s'y ajouter (ex. médecins (doctors) : Doc, avocats (lawyers) : LAW, etc.). C'est pourquoi cette information doit aussi être spécifiée.

Voici ce qui doit apparaître sur l'afficheur lors de l'appel des fonctions ci-dessus :

3.3.1. SCR_VerifyPIN ( )

Si l'espace d'affichage est limité, le minimum est :

```
Application:      Identité
Accès demandé:    Signature (non-
répudiation)
Entrez votre PIN:  **** .
```

Si la place le permet, cette information doit être traduite dans la langue spécifiée dans la fonction SCR_Init, p.ex. :

ApplicationID-PIN: Old PIN ? **** *ApplicationID*-PIN: New PIN ? ****

ex: Doc-PIN: Oude PIN ? **** Doc-PIN: Nieuwe PIN ? ****.

3.3.2. SCR_ChangePIN( )

ApplicationID-PIN: Old PIN ? **** *ApplicationID*-PIN: New PIN ? ****

ex: Doc-PIN: Oude PIN ? **** Doc-PIN: Nieuwe PIN ? ****.

Si l'espace d'affichage est limité, le minimum est :

Si la place le permet, cette information doit être traduite dans la langue spécifiée dans la fonction SCR_Init, p.ex. :

```
                PIN Verandering
Applicatie:     Advocaten
Oude   PIN:     ****
Nieuwe PIN:     ****
Nieuwe PIN:     ****  (Controle)
```

Art. 3. Spécifications du middleware Identité

1. Introduction

1.1. Matrice des environnements de développement

L'eID Toolkit offrant plusieurs interfaces pour les mêmes fonctionnalités, l'on a décidé de faire un choix pour retenir la meilleure.

La matrice ci-dessous indique l'interface recommandée pour les environnements et langages de développement courants.

API / Environnement de développement	C	ActiveX	Java	Java applet
Java			√	
C	√			
VB, Delphi		√		
.NET		√		
VBA, Vbscript, etc.		√		
Perl	√	√		
Application Web		6		√

Remarque importante :

Pour accéder à la carte eID à partir d'une page web, il convient de ne pas utiliser ActiveX, parce que :

1. ActiveX ne fonctionne qu'avec Microsoft Internet Explorer

2. Le navigateur ne fait pas confiance à ActiveX, qui sera refusé par la plupart des installations et utilisateurs

3. L'applet Java contient une interface utilisateur qui sert à demander interactivement à l'utilisateur une confirmation d'accès.

De plus, l'ActiveX actuel n'est pas scriptable : les scripts ne sont pas en mesure d'accéder à la mémoire allouée par le composant; un " wrapper " qui utilise un tampon créé par le script doit l'envelopper pour pouvoir le scripter.

2. Spécifications fonctionnelles

2.1. Gestion des versions et compatibilité

Le Toolkit gère automatiquement toutes les différentes versions des cartes. Dans l'utilisation du Toolkit, il n'est pas nécessaire de se soucier de la façon interne dont la carte traite les données puisque celles-ci seront présentées de façon uniforme via l'API.

Il existe une fonction de bas niveau pour connaître les différentes versions des composants de la carte, mais elle est réservée aux développeurs techniques qui doivent avoir accès à des propriétés très spécifiques de la carte. Une application normale n'a pas à s'inquiéter de la version de la carte.

2.2. Saisie du PIN

Plusieurs fonctions acceptent une référence PIN en guise de paramètre entrant. Si une référence PIN est fournie et que la fonction rencontre un refus d'accès (" access denied ") en tentant d'accéder aux ressources de la carte, la fonction demandera automatiquement le PIN à l'utilisateur, puis réessaiera d'accéder aux ressources (après vérification réussie du PIN).

Le contrôle du PIN est effectué " juste à temps ", le PIN n'étant demandé que s'il est nécessaire. Par exemple, un PIN permanent peut avoir été introduit antérieurement et rester valable. Dans ce cas, il n'est pas redemandé.

Si le lecteur autorise la saisie sécurisée du PIN, le clavier PIN est utilisé. A défaut, l'on a recours au clavier du PC.

2.3. Remarque sur la longueur maximale des paramètres

La longueur maximale des données renvoyées par les fonctions est indiquée dans le type du paramètre :

* Octets

* Caractères ASCII

* Caractères UTF-8

Pour les développeurs C, toutes les chaînes ASCII et UTF-8 sont terminées par un vide (null). Attention : la longueur spécifiée ne comprend pas le '[0]' à la fin des chaînes terminées par un vide (null).

2.4. Applications multifilières

La bibliothèque n'est pas à l'épreuve du multifilière. Il incombe à l'application appelante de ne pas utiliser la bibliothèque simultanément dans plusieurs filières parallèles.

Remarque : le CSP est à l'épreuve du multifilière, mais vous ne pouvez pas appeler le CSP dans une filière et le Toolkit dans une autre.

2.5. Organisation des API

Les fonctions sont divisées en 4 catégories :

* Fonctions d'initialisation et de terminaison, obligatoires pour initialiser et clôturer l'utilisation du Toolkit.

* Fonctions d'identité, servant à obtenir les données d'identité (nom, adresse, etc.) de la carte.

* Fonctions générales de haut niveau, servant à accéder à l'information de façon générique (fichiers, PIN), surtout dans les autres applications que l'identité. Il n'est pas nécessaire d'utiliser ces fonctions pour accéder aux données d'identité.

* Fonctions de bas niveau, destinées aux développeurs qui ont besoin de fonctions très techniques, ou à des fins de débogage. Les développeurs normaux peuvent ignorer ces fonctions.

2.6. Fonctions d'initialisation et de terminaison

2.6.1. BEID_Init

Cette fonction initialise le Toolkit.

Elle doit être appelée avant toute autre.

La politique de validation de certificat (via OCSP ou CRL) est donnée. Elle est valable pour tous les appels de fonction suivants, jusqu'à l'appel de BEID_Exit ( ). Les drapeaux obligatoires OCSP et CRL s'excluent mutuellement. Quand OCSP et CRL sont utilisés tous les deux, OCSP est essayé en premier lieu; en cas de réussite, le processus s'arrête; en cas d'échec, CRL est utilisé.

Paramètre	In	Out	Détail	Valeurs ou format permis	Long. max.
Reader Name	X		Nom lecteur	* Chaîne vide ou NULL pour détecter automatiquement le premier lecteur	
				* Nom PC/SC lecteur	
				* "VIRTUAL" pour un lecteur virtuel (voir 2.7.4)	
OCSP	X		Politique OCSP	0=inutilisé (défaut),1=facultatif, 2=obligatoire (voir 2.11)	
CRL	X		Politique CRL	0=inutilisé (défaut),1=facultatif, 2=obligatoire (voir 2.11)	
PC/SC handle		X	Handle PC/SC	Ce paramètre de sortie sera ignoré par la majorité des développeurs. Il s'adresse seulement à ceux qui veulent interagir avec le lecteur au niveau PC/SC, à des fins très techniques.	

2.6.2. BEID_Exit

Cette fonction nettoie toutes les données utilisées par le Toolkit.

Elle doit être appelée à la fin du programme.

Paramètre	In	Out	Détail	Valeurs ou format permis	Long. max.

2.7. Fonctions d'identité

Toutes les fonctions d'identité sont autosuffisantes. Autrement dit, il n'est pas nécessaire d'appeler une autre fonction en même temps qu'une fonction d'identité (sauf les fonctions d'initialisation et de terminaison).

Toutes les fonctions d'identité peuvent être appelées quel que soit l'état courant de la carte, et même si un DF (Data File) autre que l'identité est sélectionné, etc.

2.7.1. BEID_GetID

Paramètre	In	Out	Détail	Valeurs ou format permis	Long. max.
Version		X	Version des données ID	SHORT	
CardNumber		X	Numéro logique de la carte	ASCII	12
ChipNumber		X	Numéro physique de la puce	ASCII	16
ValidityDateBegin		X	Date de début de validité de la carte	ASCII YYYYMMDD	8
ValidityDateEnd		X	Date de fin de validité de la carte	ASCII YYYYMMDD	8
Municipality		X	Commune de délivrance de la carte	UTF-8	50
NationalNumber		X	Numéro national	ASCII	11
Name		X	Nom de famille	UTF-8	80
FirstName1		X	1er prénom (1)	UTF-8	60
FirstName2		X	2ème prénom	UTF-8	30
FirstName3		X	3ème prénom (initiale)	UTF-8	1
Nationality		X	Nationalité - code ISO	ASCII	3
BirthLocation		X	Lieu de naissance	UTF-8	50
BirthDate		X	Date de naissance	ASCII YYYYMMDD	8
Sex		X	Sexe	ASCII M/F	1
NobleCondition		X	Titre de noblesse	UTF-8	40
DocumentType		X	Type de document	LONG 1 : Citoyen belge 2 : Ressortissant Union européenne 3 : Non-ressortissant UE 7 : Carte bootstrap 8 : Carte "habilitation/machtiging"	
WhiteCane		X	Canne blanche autorisée (aveugles)	Booléen	
YellowCane		X	Canne jaune autorisée (malvoyants)	Booléen	
ExtendedMinority		X	Minorité étendue	Booléen	
HashPhoto		X	Hachage de la photo. Ces données n'intéressent pas la plupart des applications, mais seulement les plus techniques.	Binaire (SHA-1)	20
CertifCheck		X	Résultat du contrôle et de la validation du certificat	Voir 2.13	

2.7.2. BEID_GetAddress

Paramètre	In	Out	Détail	Valeurs ou format permis	Long. max.
Version		X	Version des données d'adresse	SHORT	
Street		X	Rue2	UTF-8	80
Street number		X	Numéro de maison	ASCII	8
Box number		X	Numéro de boîte	ASCII	4
Zip		X	Code postal	ASCII	6
municipality		X	Commune	UTF-8	50
Country		X	Code ISO du pays	ASCII	3
CertifCheck		X	Résultat du contrôle et de la validation du certificat	Voir 2.13	

2.7.3. BEID_GetPicture

Paramètre	In	Out	Détail	Valeurs ou format permis	Long. max.
Picture		X	Photo au format JPEG	BYTE stream	10 000
PictureLen	X	X	Longueur du Byte Stream photo	Long : taille en octets * IN : taille du tampon * OUT : taille réelle des données renvoyées	
CertifCheck		X	Résultat du contrôle et de la validation du certificat	Voir 2.13	

2.7.4. Fonctions d'identité hors ligne

Vous aurez parfois besoin d'archiver les données de la carte pour les valider immédiatement et en même temps les conserver comme preuve avec leur signature. Dans ce cas, vous devez recourir à deux fonctions : une pour lire les données brutes de la carte, l'autre pour transmettre ces données brutes aux fonctions d'identité en guise d'input.

2.7.4.1. BEID_GetRawData

Cette fonction renvoie les données brutes issues de la carte (ID, Address, Picture, RRN certificate, CardData, TokenInfo, Challenge/Response from internal authenticate). Vous devez les enregistrer pour les réutiliser plus tard.

Remarque : les données brutes sont seulement lues, sans contrôle. Pour les valider, il faut appeler les fonctions d'identité normales.

Paramètre	In	Out	Détail	Valeurs ou format permis	Long. max.
RawData		X	Données brutes		

2.7.4.2. BEID_SetRawData

Cette fonction fait des données brutes l'input des fonctions d'identité suivantes. Cela permet de vérifier des données d'identité précédemment enregistrées. Notez que le recours à cette fonction n'exige pas la présence d'un lecteur.

Pour contrôler les données brutes, vous devez :

* Appeler la fonction BEID_Init ( ) avec le paramètre lecteur égal à "VIRTUAL"

* Appeler la fonction BEID_SetRawData ( )

Appeler les fonctions d'identité normales.

Paramètre	In	Out	Détail	Valeurs ou format permis	Long. max.
RawData	X		Données brutes		

2.8. Fonctions générales de haut niveau

Ces fonctions donnent accès (un accès intégré au Toolkit) à des fonctions générales destinées aux applications qui doivent effectuer d'autres opérations que la simple consultation des données d'identité.

2.8.1. BEID_BeginTransaction

Cette fonction entame une transaction. Avant de commencer, elle attend la fin de toutes les autres transactions. Aucune autre application n'aura accès à la carte avant que BEID_EndTransaction soit appelée.

Cette fonction doit être appelée avant les autres fonctions d'exploitation de la carte, que l'on doit regrouper. Elle n'est pas nécessaire pour appeler une quelconque fonction individuelle ni les fonctions d'identité.

Normalement, une transaction sert à sélectionner une application avant l'accès à un fichier ou à un PIN.

Paramètre	In	Out	Détail	Valeurs ou format permis	Long. max.

2.8.2. BEID_EndTransaction

Cette fonction achève une transaction précédemment déclarée, pour permettre aux autres applications de reprendre les interactions avec la carte.

Elle doit être appelée à la fin de certaines fonctions d'exploitation de la carte.

Paramètre	In	Out	Détail	Valeurs ou format permis	Long. max.

2.8.3. BEID_SelectApplication

Cette fonction sélectionne une application sur la carte.

Paramètre	In	Out	Détail	Valeurs ou format permis	Long. max.
AID	X		AID application	Byte Stream - dépend de la carte (Ex. A000000177504B43532D3135)	
AIDLen	X		Longueur AID application	Long : longueur (en octets) de l'AID	

2.8.4. BEID_ReadFile

Cette fonction lit un fichier sur la carte. En présence d'une référence PIN, le PIN est demandé et corrigé au besoin (contrôle juste à temps).

Paramètre	In	Out	Détail	Valeurs ou format permis	Long. max.
FileID	X		Chemin d'accès au fichier à lire - par rapport à l'application en cours	Byte Stream - dépend de la carte (Ex. DF 00 50 32)	
FileIDLen	X		Longueur FileID	Long : longueur (en octets) du FileID donné	
OutData		X	Données renvoyées		
OutDataLen	X	X	Longueur données renvoyées	Long : taille en octets * IN : taille du tampon OutData * OUT : taille réelle des données renvoyées	64 Ko
PIN	X		PIN protégeant le fichier	Voir 2.10	

2.8.5. BEID_WriteFile

Cette fonction écrit un fichier sur la carte. En présence d'une référence PIN, le PIN est demandé et corrigé au besoin (contrôle juste à temps).

Paramètre	In	Out	Détail	Valeurs ou format permis	Long. max.
FileID	X		Chemin d'accès au fichier à écrire - par rapport à l'application en cours	Byte Stream - dépend de la carte (Ex. DF 00 50 32)	
FileIDLen	X		Longueur FileID	Long : longueur (en octets) du FileID donné	
InData	X		Données en entrée		
InDataLen	X		Longueur des données en entrée		
PIN	X		PIN protégeant le fichier	Voir 2.10	

2.8.6. BEID_VerifyPIN

Cette fonction vérifie un PIN.

Paramètre	In	Out	Détail	Valeurs ou format permis	Long. max.
PIN	X		PIN protégeant le fichier	Voir 2.10	
Pin	X		Le PIN donné	ASCII Si vide ou NULL, demander à l'utilisateur	12
NrTriesLeft		X	Nombre de tentatives restant	Long : nombre de tentatives restant	

2.8.7. BEID_ChangePIN

Cette fonction modifie un PIN.

Paramètre	In	Out	Détail	Valeurs ou format permis	Long. max.
PIN	X		PIN protégeant le fichier	Voir 2.10	
OldPin	X		L'ancien PIN	ASCII Si vide ou NULL, demander à l'utilisateur	12
NewPin	X		Le nouveau PIN	ASCII Si vide ou NULL, demander à l'utilisateur	12
NrTriesLeft		X	Nombre de tentatives restant	Long : nombre de tentatives restant	

2.8.8. BEID_GetPINStatus

Cette fonction obtient le statut du PIN. Le résultat peut être signé par la clé de base de la carte (clé " 0x81 " dans l'application DF actuelle).

Paramètre	In	Out	Détail	Valeurs ou format permis	Long. max.
PIN	X		PIN protégeant le fichier	Voir 2.10	
NrTriesLeft		X	Nombre de tentatives restant	Long : nombre de tentatives restant	
signature	X		Signature requise	Booléen	
signedStatus		X	Statut signature	Byte Stream	256
SignedStatusLen	X	X	Longueur statut signature	Long : taille en octets * IN : taille du Byte Stream donné * OUT : taille réelle des données renvoyées : 256	

2.9. Fonctions de bas niveau

Remarque : les fonctions de bas niveau sont destinées aux applications qui doivent accéder à des fonctions techniques spécifiques, étrangères aux fonctions ordinaires, ou à des fins de débogage. Il n'est pas nécessaire d'utiliser ces fonctions de bas niveau dans les applications normales.

2.9.1. BEID_GetVersionInfo

Cette fonction renvoie la version des différents composants (applet, système d'exploitation...). Le résultat peut être signé par la clé de base de la carte (clé " 0x81 " dans l'application DF actuelle).

Paramètre	In	Out	Détail	Valeurs ou format permis	Long. max.
VersionInfo		X	Voir spécifications applet et contenu carte		
signature	X		Signature requise	Booléen	
signedStatus		X	Statut signature	BYTE stream	128
SignedStatusLen	X	X	Longueur statut signature	Long : taille en octets * IN : taille du tampon : min. 128 OUT : taille réelle des données renvoyées (128)	

2.9.2. BEID_SendAPDU

Cette fonction envoie une commande APDU à la carte.

Attention : quand vous envoyez une commande APDU à la carte, le Toolkit n'est pas en mesure de vérifier la compatibilité entre les versions des applets (sauf pour la partie communication). L'appel risque de ne pas fonctionner avec une version suivante de l'applet.

En présence d'une référence PIN, le PIN est demandé et corrigé au besoin (contrôle juste à temps).

Remarque : la commande est d'abord tentée sans demander le PIN. En cas d'échec ("access denied"), le PIN est demandé et la commande réessayée.

Paramètre	In	Out	Détail	Valeurs ou format permis	Long. max.
CmdAPDU	X		La commande APDU envoyée à la carte	Byte Stream - dépend de la carte	256
CmdAPDULen	X		La longueur de la commande APDU		
PIN	X		PIN protégeant le fichier	Voir 2.10	
RespAPDU		X	La réponse APDU reçue de la carte		256
RespAPDULen	X	X	La longueur de la réponse APDU	Long : taille en octets * IN : taille du tampon OUT : taille réelle des données renvoyées	

2.9.3. BEID_FlushCache

Cette fonction vide les données mises en cache en mémoire et sur le disque.

Paramètre	In	Out	Détail	Valeurs ou format permis	Long. max.

2.10. Identification PIN

Lorsque vous communiquez un PIN au Toolkit, vous pouvez spécifier le PIN id. PKCS#15 ou la référence PIN du système d'exploitation.

De plus, si le PIN ne fait pas partie des PIN enregistrés dans le Toolkit, vous devez fournir une chaîne décrivant la raison pour laquelle le PIN est requis (p.ex. "Personal data modification"). Vous devez aussi fournir une courte chaîne (max. 3 caractères) à afficher sur l'écran du lecteur de carte (ex. : "MOD").

Un PIN se compose donc de 4 champs :

▷ le type de PIN : PKCS#15 ou système d'exploitation (voir 2.10.1)

▷ la référence OS du PIN ou : PKCS#15 id

▷ le code d'utilisation (Voir 2.10.2)

▷ la description longue (dans la langue de l'utilisateur)

▷ la description courte (dans la langue de l'utilisateur)

Pour améliorer la sécurité et standardiser l'affichage de l'information, la description donnée par l'application peut être écrasée par le Toolkit si celui-ci peut déterminer l'utilisation exacte.

En l'absence de PIN à contrôler, une valeur NULL doit être utilisée.

Remarque : quand un PIN est disponible dans la structure PKCS#15, il est plus sûr d'utiliser la référence PKCS#15 au lieu de celle de l'OS, celui-ci pouvant changer ultérieurement.

2.10.1. Types de PIN

Valeur	Constante C	Explication
0	BEID_PIN_TYPE_PKCS15	PKCS15 PIN
1	BEID_PIN_TYPE_OS	OS PIN

2.10.2. Utilisations du PIN

Valeur	Constante C	Explication
0		Application définie
1	BEID_USAGE_AUTH	Authentification
2	BEID_USAGE_SIGN	Signature
...	...	Autres utilisations à venir

2.11. Paramètres de politique en entrée - OCSP et CRL

Valeur	Constante C	Explication
0	BEID_OCSP_CRL_NOT_USED	Pas de contrôle CRL/OCSP
1	BEID_OCSP_CRL_OPTIONAL	Contrôle facultatif CRL/OCSP
2	BEID_OCSP_CRL_MANDATORY	Contrôle obligatoire CRL/OCSP

2.12. Statut des fonctions

Chaque fonction renvoie un code général spécifiant le type d'erreur. Dans la plupart des cas, seul ce code d'erreur sera utilisé. Cependant, pour obtenir des informations détaillées sur les raisons techniques, les codes d'état suivants sont également prévus :

* Code d'erreur système (long)

* Code d'erreur PC/SC (long)

* Mot d'état (double octet) de la carte à puce

2.12.1. Codes d'erreur généraux

Valeur	Constante C	Explication
0	BEID_OK	Fonction réussie
1	BEID_E_SYSTEM	Erreur système inconnue (voir code d'erreur système)
2	BEID_E_PCSC	Erreur PC/SC inconnue (voir code d'erreur PC/SC)
3	BEID_E_CARD	Erreur carte inconnue (voir mot d'état carte)
4	BEID_E_BAD_PARAM	Paramètre non valable (pointeur NULL, hors limites, etc.)
5	BEID_E_INTERNAL	Un contrôle de cohérence interne a échoué
6	BEID_E_INVALID_HANDLE	Le handle fourni n'est pas valable
7	BEID_E_INSUFFICIENT_BUFFER	Le tampon destiné à recevoir les données renvoyées est trop petit
8	BEID_E_COMM_ERROR	Une erreur de communication interne a été détectée
9	BEID_E_TIMEOUT	Le timeout spécifié a expiré
10	BEID_E_UNKNOWN_CARD	La carte à puce n'a pas été reconnue
11	BEID_E_KEYPAD_CANCELLED	Saise au clavier PIN annulée

Valeur	Constante C	Explication
12	BEID_E_KEYPAD_TIMEOUT	Timeout renvoyé par le clavier PIN
13	BEID_E_KEYPAD_PIN_MISMATCH	Les deux PIN ne correspondent pas
14	BEID_E_KEYPAD_MSG_TOO_LONG	Message trop long sur clavier PIN
15	BEID_E_INVALID_PIN_LENGTH	Longueur PIN non valable
16	BEID_E_VERIFICATION	Erreur dans une vérification de signature ou une validation de certificat (voir 2.13)
17	BEID_E_NOT_INITIALIZED	Toolkit non initialisé
18	BEID_E_UNKNOWN	Une erreur interne a été détectée mais sa source est inconnue
19	BEID_E_UNSUPPORTED_FUNCTION	Fonction non supportée
20	BEID_E_INCORRECT_VERSION	La version du Toolkit est incompatible avec l'interface appelante. Le programme doit être recompilé.
21	BEID_E_INVALID_ROOT_CERT	

2.13. Contrôle de signature et validation de certificat

Toute fonction renvoyant des données signées contrôle toujours la signature ainsi que l'intégrité de l'ensemble de la chaîne du certificat.

La fonction renvoie :

* l'état du contrôle de signature (long)

* l'état global de la validation de certificat (long)

* pour chaque certificat :

— le certificat

— l'étiquette du certificat

— l'état du contrôle individuel

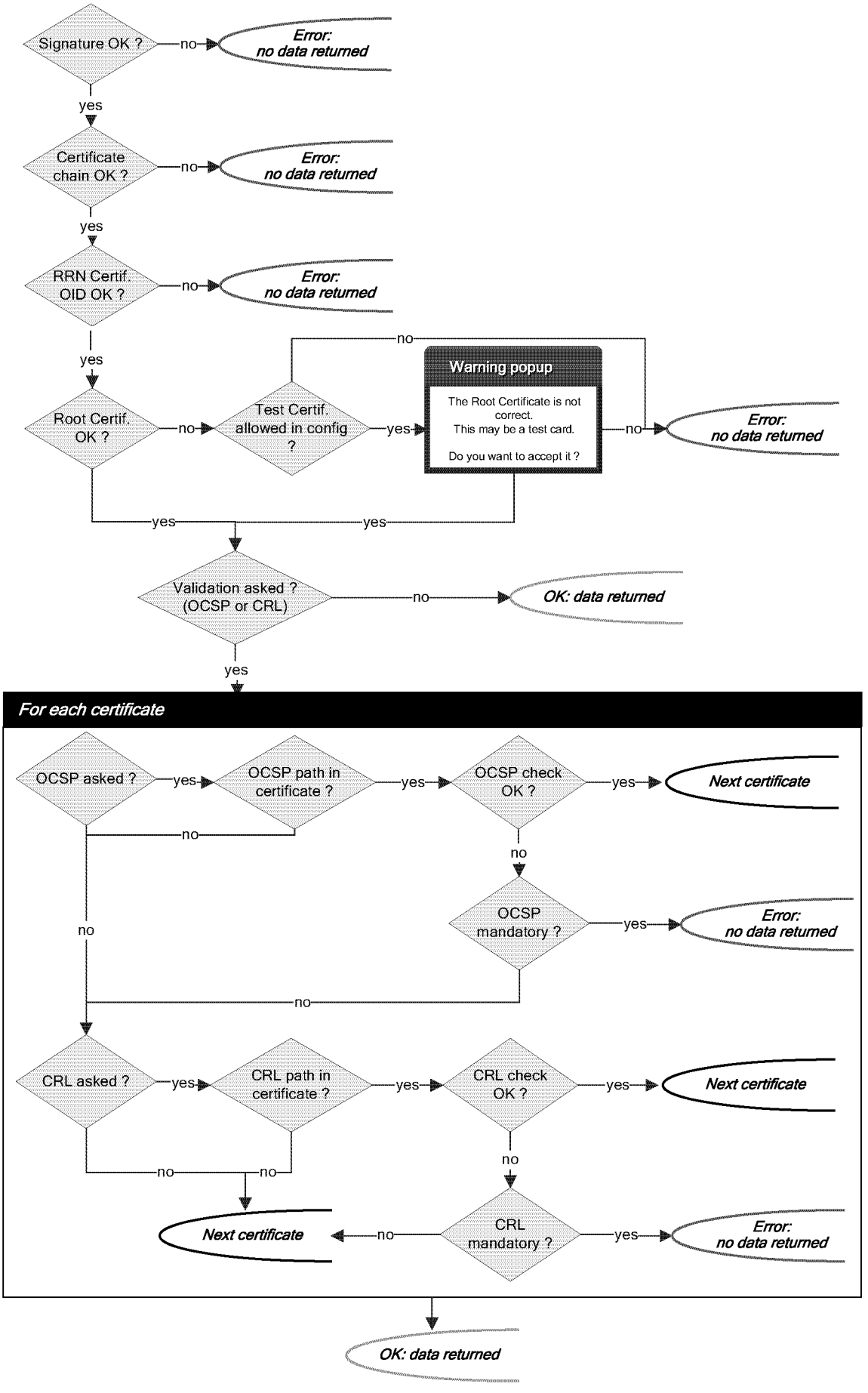
— l'état de la validation individuelle

— la politique individuelle appliquée : OCSP ou CRL.

En cas d'erreur dans la signature ou la chaîne de validation, toutes les fonctions s'arrêtent avec un code d'erreur et **sans renvoyer de données.**

Si le certificat racine n'est pas le certificat officiel, une incrustation *(pop-up box)* avertit que le *Root Certificate* n'est pas celui que l'on attendait. L'utilisateur peut accepter temporairement ce certificat racine. C'est indispensable pour tester les cartes qui possèdent un autre certificat racine que l'officiel.

Voici un schéma décrivant le déroulement du controle de signature :



2.13.1. Contrôle de signature

Valeur	Constante C	Explication
-1	BEID_SIGNATURE_ PROCESSING_ERROR	Erreur dans la vérification de la signature.
0	BEID_SIGNATURE_VALID	La signature est valable.
1	BEID_SIGNATURE_INVALID	La signature n'est pas valable.
2	BEID_SIGNATURE_VALID_WRONG_RRNCERT	La signature est valable mais il s'agit d'un certificat RRN non valable.
3	BEID_SIGNATURE_INVALID_WRONG_RRNCERT	La signature n'est pas valable et le certificat RRN non plus.

2.13.2. Résultat du contrôle et de la validation du certificat

Valeur	Constante C	Explication
0	BEID_CERTSTATUS_CERT_VALIDATED_OK	La validation a réussi.
1	BEID_CERTSTATUS_CERT_NOT_VALIDATED	La validation n'a pas eu lieu.
2	BEID_CERTSTATUS_UNABLE_TO_GET_ISSUER_CERT	Impossible d'obtenir le certificat de l'émetteur.
3	BEID_CERTSTATUS_UNABLE_TO_GET_CRL	Impossible d'obtenir le CRL du certificat.
4	BEID_CERTSTATUS_UNABLE_TO_DECRYPT-_CERT_SIGNATURE	Impossible de décrypter la signature du certificat.
5	BEID_CERTSTATUS_UNABLE_TO_DECRYPT-_CRL_SIGNATURE	Impossible de décrypter la signature du CRL.
6	BEID_CERTSTATUS_UNABLE_TO_DECO-DE_ISSUER_PUBLIC_KEY	Impossible de décoder la clé publique de l'émetteur.
7	BEID_CERTSTATUS_CERT_SIGNATURE_FAILURE	Echec de signature du certificat.
8	BEID_CERTSTATUS_CRL_SIGNATURE_FAILURE	Echec de signature du CRL.
9	BEID_CERTSTATUS_CERT_NOT_YET_VALID	Le certificat n'est pas encore valable.
10	BEID_CERTSTATUS_CERT_HAS_EXPIRED	Le certificat a expiré.
11	BEID_CERTSTATUS_CRL_NOT_YET_VALID	Le CRL n'est pas encore valable.
12	BEID_CERTSTATUS_CRL_HAS_EXPIRED	Le CRL a expiré.
13	BEID_CERTSTATUS_ERR_IN_CERT_NOT_BE-FORE_FIELD	Erreur de format dans le champ notBefore du certificat.
14	BEID_CERTSTATUS_ERR_IN_CERT_NOT_AF-TER_FIELD	Erreur de format dans le champ notAfter du certificat.
15	BEID_CERTSTATUS_ERR_IN_CRL_LAST_U-PDATE_FIELD	Erreur de format dans le champ lastUpdate du CRL.
16	BEID_CERTSTATUS_ERR_IN_CRL_NEXT_U-PDATE_FIELD	Erreur de format dans le champ nextUpdate du CRL.
17	BEID_CERTSTATUS_OUT_OF_MEM	Mémoire insuffisante.
18	BEID_CERTSTATUS_DEPTH_ZERO_SELF_SI-GNED_CERT	Certificat autosigné.
19	BEID_CERTSTATUS_SELF_SIGNED_CERT_IN_CHAIN	Certificat autosigné dans la chaîne de certification.

Valeur	Constante C	Explication
20	BEID_CERTSTATUS_UNABLE_TO_GET_ISSUER_CERT_LOCALLY	Impossible d'obtenir le certificat de l'émetteur local.
21	BEID_CERTSTATUS_UNABLE_TO_VERIFY_LEAF_SIGNATURE	Impossible de vérifier le premier certificat.
22	BEID_CERTSTATUS_CERT_CHAIN_TOO_LONG	Chaîne de certificat trop longue.
23	BEID_CERTSTATUS_CERT_REVOKED	Certificat révoqué.
24	BEID_CERTSTATUS_INVALID_CA	Certificat de CA non valable.
25	BEID_CERTSTATUS_PATH_LENGTH_EXCEEDED	Limite de longueur de chemin d'accès dépassée.
26	BEID_CERTSTATUS_INVALID_PURPOSE	But du certificat non pris en charge.
27	BEID_CERTSTATUS_CERT_UNTRUSTED	Certificat non digne de confiance.
28	BEID_CERTSTATUS_CERT_REJECTED	Certificat rejeté.
29	BEID_CERTSTATUS_SUBJECT_ISSUER_MISMATCH	Sujet et émetteur ne correspondent pas.
30	BEID_CERTSTATUS_AKID_SKID_MISMATCH	Autorité et identificateur de clé de sujet ne correspondent pas.
31	BEID_CERTSTATUS_AKID_ISSUER_SERIAL_MISMATCH	Autorité et numéro de série d'émetteur ne correspondent pas.
32	BEID_CERTSTATUS_KEYUSAGE_NO_CERTSIGN	L'utilisation de la clé ne couvre pas la signature du certificat.
33	BEID_CERTSTATUS_UNABLE_TO_GET_CRL_ISSUER	Impossible d'obtenir le certificat de l'émetteur de CRL.
34	BEID_CERTSTATUS_UNHANDLED_CRITICAL_EXTENSION	Extension critique non gérée.

2.13.3. Politiques OCSP et CRL appliquées

Valeur	Constante C	Explication
0	BEID_POLICY_NONE	Pas de politique appliquée.
1	BEID_POLICY_OCSP	Politique OCSP appliquée.
2	BEID_POLICY_CRL	Politique CRL appliquée.
3	BEID_POLICY_BOTH	Politiques OCSP et CRL appliquées.

3. Interfaces de programmation

3.1. API C

Tous les tampons de sortie doivent être alloués par l'application appelante.

Toutes les fonctions utilisent les conventions d'appel de C et non de Pascal.

Struct packing est fixé à 8.

3.1.1. Structures

```
typedef struct {
     long general;              // General return code
     long system;               // System error (errno)
     long pcsc;                 // PC/SC error
     BYTE cardSW[2];            // Card status word
} BEID_Status;

typedef struct {
 BYTE certif[…];                // Byte stream encoded certificate
 long certifLength;             // Size in bytes of the encoded certificate
 char certifLabel[…];           // Label of the certificate (Authentication,
Signature, CA, Root,…)
 long certifStatus;             // Validation status code (see 2.3.3
                                                    Certificate
                                                    checking and
                                                    validation results)
} BEID_Certif;

typedef struct {
     long usedPolicy;                   // Policy used (see 2.13.3)
     BEID_Certif certificates[…];       // Array of BEID_Certif
structures
     long certificatesLength;           // Number of elements in Array
     long signatureCheck;               // Status of signature (for ID and
                                                    Address) or
                                        // hash (for Picture) on retrieved
                                                    field (see 2.13)

} BEID_Certif_Check;

typedef struct {
     // Get Card Data
     BYTE SerialNumber[16];
     BYTE ComponentCode;
     BYTE OSNumber;
     BYTE OSVersion;
     BYTE SoftmaskNumber;
     BYTE SoftmaskVersion;
     BYTE AppletVersion;
     ushort GlobalOSVersion;
     BYTE AppletInterfaceVersion;
     BYTE PKCS1Support;
     BYTE KeyExchangeVersion;
     BYTE ApplicationLifeCycle;
     // TokenInfo Data
     BYTE GraphPerso;
     BYTE ElecPerso;
     BYTE ElecPersoInterface;
     BYTE Reserved;
```

```
} BEID_VersionInfo;

typedef struct {
     short version;
     char cardNumber[…];
     char chipNumber[…];
     char validityDateBegin[…];
     char validityDateEnd[…];
     TCHAR municipality[…];
     char nationalNumber[…];
     TCHAR name[…];
     TCHAR firstName1[…];
     TCHAR firstName2[…];
     TCHAR firstName3[…];
     char nationality[…];
     TCHAR birthLocation[…];
     char birthDate[…];
     char sex[…];
     TCHAR nobleCondition[…];
     long documentType;
     BOOL whiteCane;
     BOOL yellowCane;
     BOOL extendedMinority;
     BYTE hashPhoto[…]
} BEID_ID_Data;

typedef struct {
     short version;
     TCHAR street[…];
     char streetNumber[…];
     char boxNumber[…];
     char zip[…];
     TCHAR municipality[…];
     char country[…];
} BEID_Address;

typedef struct {
   BYTE *data;
   unsigned long length;
  } BEID_Bytes;

typedef struct {
  long pinType;          // PIN Type (see 2.10.1)
  BYTE id;               // PIN reference or ID
  long usageCode;        // PIN Usage (see 2.10.2)
  char *shortUsage;      // May be NULL for usage known by the
middleware
  char *longUsage;       // May be NULL for usage known by the
middleware
} BEID_Pin;


typedef struct { … } BEID_Raw;
```

3.1.2. Functions

```
BEID_Status                                    // return code
     BEID_Init(
     char *ReaderName,                         // Reader Name
long OCSP,                                     // OCSP policy certificates
checking & validity
     long CRL,                                 // CRL policy certificates
     checking & validity
     long *CardHandle,                         // output PC/SC handle
     );

BEID_Status                                    // return code
     BEID_Exit(
     );

BEID_Status                                    // return code
     BEID_GetID(
     BEID_ID_Data *IDData,                     // output data
BEID_Certif_Check *CertifCheck                 // certificates checking &
validity
     );

BEID_Status                                    // return code
     BEID_GetAddress(
     BEID_ID_Address *Address,                 // output address data
BEID_Certif_Check *CertifCheck                 // certificates checking &
validity
     );

BEID_Status                                    // return code
     BEID_GetPicture(
     BEID_Bytes *Picture,                      // output picture (JPEG
     format)
BEID_Certif_Check *CertifCheck                 // certificates checking &
validity
     );

BEID_Status                                    // return code
     BEID_GetRawData(
     BEID_Raw *Raw                             // output Raw Data
);

BEID_Status                                    // return code
     BEID_SetRawData(
     BEID_Raw *Raw                             // input Raw Data
);

BEID_Status                                // return code
     BEID_GetVersionInfo(
     BEID_VersionInfo *pVersionInfo,
     BOOL Signature,                       // Signature needed
     BEID_Bytes *SignedStatus,             // Signature 256 bytes
     );

BEID_Status                                // return code
BEID_BeginTransaction( );
```

```
BEID_Status                          // return code
BEID_EndTransaction( );

BEID_Status                          // return code
BEID_SelectApplication(
     BEID_Bytes *Application         // Application ID
     );

BEID_Status                          // return code
BEID_FlushCache( );

BEID_Status                          // return code
BEID_SendAPDU(
     BEID_Bytes *CmdAPDU,            // Command APDU
     BEID_Pin *pin,                  // Pin Reference
     BEID_Bytes *RespAPDU            // Response APDU
     );

BEID_Status                          // return code
BEID_VerifyPIN(
     BEID_Pin *pin,                  // Pin Reference
     char *Pin,                      // Pin code
     long *TriesLeft,                // Tries remaining
     );

BEID_Status                          // return code
BEID_ChangePIN(
     BEID_Pin *pin,                  // Pin Reference
     char *OldPin,                   // Old Pin code
     char *NewPin,                   // New Pin code
     long *TriesLeft                 // Tries remaining
     );

BEID_Status                          // return code
BEID_GetPINStatus(
     BEID_Pin *pin,                  // Pin Reference
     long *TriesLeft,                // Tries remaining
     BOOL bSignature,                // Signature needed
     BEID_Bytes *SignedStatus        // Signature
     );

BEID_Status                          // return code
BEID_ReadFile(
     BEID_Bytes *FileID,             // File to read relative path
     BEID_Bytes *OutData,            // Returned data
     BEID_Pin *pin,                  // Pin Reference
);

BEID_Status                          // return code
BEID_WriteFile(
     BEID_Bytes *FileID,             // File to write relative path
     BEID_Bytes *InData,             // Write buffer
     BEID_Pin *pin,                  // Pin Reference
);
```

3.2. API Java

Pour des raisons de compatibilité avec le code natif qui définit l'accès à la carte, l'implémentation Java ne fait pas de gestion d'exceptions : toutes les erreurs sont renvoyées sous forme de codes d'erreur (voir 2.12)

Package: be.belgium.eid;

3.2.1. Data Classes

```
public class BEID_Address
{
  public BEID_Address()
  public short getVersion()
  public String getStreet()
  public String getStreetNumber()
  public String getBoxNumber()
  public String getZip()
  public String getMunicipality()
  public String getCountry()
}

public class BEID_Raw { … }

public class BEID_Bytes
{
  public BEID_Bytes()
  public byte[] getData()
}

public class BEID_Certif_Check
{
  public BEID_Certif_Check()
  public int getUsedPolicy()
  public BEID_Certif getCertificate(int Index)
  public int getCertificatesLength()
  public int getSignatureCheck()
 }

public class BEID_Certif
{
  public BEID_Certif()
  public byte[] getCertif()
  public String getCertifLabel()
  public int getCertifStatus()
}

public class BEID_Long
{
   public BEID_Long()
   public long getLong()
 }

public class BEID_Pin
{
  public BEID_Pin()
  public void setPinType(int pinType)
  public void setId(short id)
  public void setUsageCode(int usageCode)
  public void setShortUsage(String shortUsage)
  public void setLongUsage(String longUsage)
  }
```

```
public class BEID_Status
{
  public BEID_Status()
  public int getGeneral()
  public int getSystem()
  public int getPcsc()
  public byte[] getCardSW()
}

public class BEID_VersionInfo
{
  public BEID_VersionInfo()
  public byte[] getSerialNumber()
  public short getComponentCode()
  public short getOSNumber()
  public short getOSVersion()
  public short getSoftmaskNumber()
  public short getSoftmaskVersion()
  public short getAppletVersion()
  public int getGlobalOSVersion()
  public short getAppletInterfaceVersion()
  public short getPKCS1Support()
  public short getKeyExchangeVersion()
  public short getApplicationLifeCycle()
  public short getGraphPerso()
  public short getElecPerso()
  public short getElecPersoInterface()
  public short getReserved()
}

public class BEID_ID_Data
{
  public BEID_ID_Data()
  public short getVersion()
  public String getCardNumber()
  public String getChipNumber()
  public String getValidityDateBegin()
  public String getValidityDateEnd()
  public String getMunicipality()
  public String getNationalNumber()
  public String getName()
  public String getFirstName1()
  public String getFirstName2()
  public String getFirstName3()
  public String getNationality()
  public String getBirthLocation()
  public String getBirthDate()
  public String getSex()
  public String getNobleCondition()
  public int getDocumentType()
  public boolean getWhiteCane()
  public boolean getYellowCane()
  public boolean getExtendedMinority()
  public byte[] getHashPhoto()
  }
```

3.2.2. Main Class

```
public class eidlib
{
  public static BEID_Status BEID_Init(String ReaderName, int
OCSP, int CRL, BEID_Long CardHandle)

  public static BEID_Status BEID_Exit()

  public static BEID_Status BEID_GetID(BEID_ID_Data IDData,
BEID_Certif_Check CertifCheck)

  public static BEID_Status BEID_GetAddress(BEID_Address Address,
BEID_Certif_Check CertifCheck)

  public static BEID_Status BEID_GetPicture(BEID_Bytes Picture,
BEID_Certif_Check CertifCheck)

  public static BEID_Status BEID_GetRawData(BEID_Raw RawData)

  public static BEID_Status BEID_SetRawData(BEID_Raw RawData)

  public static BEID_Status BEID_GetVersionInfo(BEID_VersionInfo
VersionInfo, int Signature,  BEID_Bytes SignedStatus)

  public static BEID_Status BEID_BeginTransaction()

  public static BEID_Status BEID_EndTransaction()

  public static BEID_Status BEID_SelectApplication(byte[]
Application)

  public static BEID_Status BEID_VerifyPIN(BEID_Pin PinData,
String Pin, BEID_Long TriesLeft)

  public static BEID_Status BEID_ChangePIN(  BEID_Pin Pin, String
                                             oldPin, String
                                             newPin,
                                             BEID_Long TriesLeft)

  public static BEID_Status BEID_GetPINStatus( BEID_Pin PinData,
                                                BEID_Long
                                                TriesLeft, int
                                                Signature,
                                                BEID_Bytes
                                                SignedStatus)

  public static BEID_Status BEID_ReadFile( byte[] FileID,
BEID_Bytes OutData, BEID_Pin PinData)

  public static BEID_Status BEID_WriteFile(byte[] FileID, byte[]
InData, BEID_Pin PinData)

  public static BEID_Status BEID_FlushCache()

  public static BEID_Status BEID_SendAPDU(byte[] CmdAPDU,
BEID_Pin PinData, BEID_Bytes RespAPDU)
 }
```

3.2.3. Applet Classe

```
public class BEID_Applet
{
  public BEID_Applet()
  public int InitLib(String strReader) // If null or empty,
applet parameter is used.
  public int ExitLib()
  public String getCardNumber()
  public String getChipNumber()
  public String getValidityDateBegin()
  public String getValidityDateEnd()
  public String getIssMunicipality()
  public String getNationalNumber()
  public String getName()
  public String getFirstName1()
  public String getFirstName2()
  public String getFirstName3()
  public String getNationality()
  public String getBirthLocation()
  public String getBirthDate()
  public String getSex()
  public String getNobleCondition()
  public int getDocumentType()
  public boolean getWhiteCane()
  public boolean getYellowCane()
  public boolean getExtendedMinority()
  public String getStreet()
  public String getStreetNumber()
  public String getBoxNumber()
  public String getZip()
  public String getMunicipality()
  public String getCountry()
  public byte[] GetPicture()
  public int SetRawData(byte[] IDData, byte[] SigIDData, byte[]
AddrData,
                        byte[] SigAddrData,
                        byte[] PictureData, byte[] RNData, byte[]
cardData,
                        byte[] tokenInfoData,
                        byte[] challengeData, byte[]
responseData)
  public byte[] GetRawIDData()
  public byte[] GetRawSigIDData()
  public byte[] GetRawAddrData()
  public byte[] GetRawSigAddrData()
  public byte[] GetRawPictureData()
  public byte[] GetRawCardData()
  public byte[] GetRawTokenInfoData()
  public byte[] GetRawRNData()
  public byte[] GetRawChallengeData()
  public byte[] GetRawResponseData()
}
```

Paramètres applet :

Nom	Valeurs	Nom
Card Reader name	Reader	String
OCSP checking	OCSP	Voir 2.11
CRL checking	CRL	Voir 2.11

Snippet HTML :

```
<applet
  codebase = "."
  archive  = "eidlib.jar"
  code     = "be.belgium.eid.BEID_Applet.class"
  name     = "BEIDApplet"
  width    = "0"
  height   = "0"
  hspace   = "0"
  vspace   = "0"
>
<param name="Reader" value="">
<param name="OCSP" value="0">
<param name="CRL" value="0">
</applet>
```

3.3. ActiveX API

3.3.1. Interfacedefinities

De interfacedefinities zijn in IDL-formaat.
Control Name: **EIDLibCtrl.EIDlib**

```
Interface IRetStatus
{
     long * GetGeneral( );
     long * GetPCSC( );
     long * GetSystem( );
     VARIANT * GetCardSW( );     ‘ VARIANT type : VT_ARRAY | VT_UI1
}

Interface IMapCollection
{
     VARIANT * GetValue( [IN] BSTR strKey );
     void SetValue( [IN] BSTR strKey, [IN] VARIANT vtValue );
     long * GetCount( );
}

Interface ICertif
{
     VARIANT * GetCertif( );      ‘ VARIANT type : VT_ARRAY | VT_UI1
     BSTR * GetLabel( );
     long * GetStatus( );          ‘ Validation status code see 2.13
}
```

```
Interface ICertifCheck
{
     long * GetPolicy( );
     long * GetSignatureCheck( );        ‘ Validation status code see 2.13
     VARIANT * GetCertificates( );       ‘ Array of ICertif, VARIANT type :
VT_ARRAY | VT_DISPATCH
}
Interface IPin
{
  void   SetPinType([IN] long Type);     ‘ BEID_PIN_TYPE_PKCS15 or
BEID_PIN_TYPE_OS
  void   SetID([IN]VARIANT ID);          ‘ VARIANT type : VT_UI1
  void   SetUsageCode([IN]long usageCode);
  void   SetshortUsage([IN]BSTR shortUsage); ‘ May be NULL for usage
known by the middleware
  void   SetlongUsage([IN]BSTR longUsage);   ‘ May be NULL for usage
known by the middleware
}
Interface IRaw
{
     void SetIDData([in] VARIANT vtID); ‘ VARIANT type : VT_ARRAY |
VT_UI1
     VARIANT *GetIDData(); ‘ VARIANT type : VT_ARRAY | VT_UI1
     void SetIDSigData([in] VARIANT vtIDSigData); ‘ VARIANT type :
VT_ARRAY | VT_UI1
     VARIANT *GetIDSigData(); ‘ VARIANT type : VT_ARRAY | VT_UI1
     void SetAddrData([in] VARIANT vtAddrData); ‘ VARIANT type :
VT_ARRAY | VT_UI1
     VARIANT * GetAddrData(); ‘ VARIANT type : VT_ARRAY | VT_UI1
     void SetAddrSigData([in] VARIANT vtAddrSigData); ‘ VARIANT type
: VT_ARRAY | VT_UI1
     VARIANT *GetAddrSigData(); ‘ VARIANT type : VT_ARRAY | VT_UI1
     void SetPictureData([in] VARIANT vtPictureData); ‘ VARIANT type
: VT_ARRAY | VT_UI1
     VARIANT *GetPictureData(); ‘ VARIANT type : VT_ARRAY | VT_UI1
     void SetRNData([in] VARIANT vtRNData); ‘ VARIANT type : VT_ARRAY
| VT_UI1
     VARIANT *GetRNData(); ‘ VARIANT type : VT_ARRAY | VT_UI1
     void SetCardData([in] VARIANT vtCardData); ‘ VARIANT type :
VT_ARRAY | VT_UI1
     VARIANT *GetCardData(); ‘ VARIANT type : VT_ARRAY | VT_UI1
     void SetTokenInfoData([in] VARIANT vtTokenInfoData); ‘ VARIANT
type : VT_ARRAY | VT_UI1
     VARIANT *GetTokenInfoData(); ‘ VARIANT type : VT_ARRAY | VT_UI1
     void SetChallengeData([in] VARIANT vtChallengeData); ‘ VARIANT
type : VT_ARRAY | VT_UI1
     VARIANT *GetChallengeData(); ‘ VARIANT type : VT_ARRAY | VT_UI1
     void SetResponseData([in] VARIANT vtResponseData); ‘ VARIANT
type : VT_ARRAY | VT_UI1
     VARIANT * GetResponseData(); ‘ VARIANT type : VT_ARRAY | VT_UI1
}
```

3.3.2. Fonctions
Les définitions de fonction sont au format IDL.

```
IRetStatus *
     Init(
     [IN] BSTR strReaderName,
     [IN] long lOCSP,
     [IN] long lCRL,
     [OUT] long * plHandle
     );

IRetStatus *
     Exit(
     );

‘ L’IMapcollection contient les paires clé/valeur des données renvoyées.
‘ Les clés sont prédéfinies pour ne pas redéfinir l’interface pour les nouvelles
versions.
‘ Clés :    CardNumber, ChipNumber, BeginValidityDate, EndValidityDate,
            IssuingMunicipality, NationalNumber, Name, FirstName1, FirstName2,
            FirstName3, Nationality, BirthPlace, BirthDate, Gender, NobilityTitle,
            DocumentType, WhiteCane, YellowCane, ExtendedMinority
IRetStatus *
     GetID(
     [OUT] IMapCollection ** ppMapCollection, ‘ Allocated by
the Toolkit
     [OUT] ICertifCheck ** ppCertifCheck      ‘ Allocated by
     the Toolkit
);

‘ Keys:    Street, HouseNumber, BoxNumber, ZIPCode, Municipality, Country
IRetStatus *
     GetAddress(
     [OUT] IMapCollection ** ppMapCollection,  ‘ Allocated by
the Toolkit
     [OUT] ICertifCheck ** ppCertifCheck       ‘ Allocated by
the Toolkit
     );

‘ Keys:    Picture
IRetStatus *
     GetPicture(
     [OUT] IMapCollection ** ppMapCollection,  ‘ Allocated by
the Toolkit
     [OUT] ICertifCheck ** ppCertifCheck       ‘ Allocated by
the Toolkit
     );

IRetStatus *
     GetRawData (
     [OUT] IRaw ** ppRaw                       ‘ Allocated by
the library
     );

IRetStatus *
     SetRawData (
     [IN] IRaw * pRaw                          ‘
     );
```

```
IRetStatus * GetVersionInfo(
     [IN] BOOL bSignature,
     [OUT] IMapCollection ** ppMapCollection,   ' Allocated by
the Toolkit
     [OUT] VARIANT * pvSignature                ' VARIANT
type : VT_ARRAY | VT_UI1
     );

IRetStatus *
     BeginTransaction( );

IRetStatus *
     EndTransAction( );

IRetStatus *
     FlushCache( );

IRetStatus *
     SelectApplication(
     [IN] VARIANT vtApplication    ' VARIANT type : VT_ARRAY
| VT_UI1
     );

IRetStatus *
     SendAPDU(
     [IN] VARIANT vtCommand,       ' VARIANT type : VT_ARRAY
| VT_UI1
     [IN] IPin * pin,
     [OUT] VARIANT * vtResponse    ' VARIANT type : VT_ARRAY
| VT_UI1
     );

IRetStatus *
     VerifyPin(
          [IN] IPin * pin,
          [IN] BSTR strPin,
          [OUT] long * plTriesLeft
          );

IRetStatus *
     ChangePin(
     [IN] IPin * pin,
     [IN] BSTR strOldPin,
     [IN] BSTR strNewPin ,
     [OUT] long * plTriesLeft);

IRetStatus *
     GetPinStatus(
     [IN] IPin * pin,
     [IN] BOOL bSignature,
     [OUT] VARIANT * pvSignature,  ' VARIANT type : VT_ARRAY
| VT_UI1
     [OUT] long * plTriesLeft);
```

```
IRetStatus *
     ReadFile(
     [IN] IPin * pin,
     [IN] VARIANT strFileID,          ‘ VARIANT type : VT_ARRAY
| VT_UI1
     [OUT] VARIANT * pvtData          ‘ VARIANT type : VT_ARRAY
| VT_UI1
     );

IRetStatus *
     WriteFile(
     [IN] IPin * pin,
     [IN] VARIANT strFileID,          ‘ VARIANT type : VT_ARRAY
| VT_UI1
     [IN] VARIANT vtData              ‘ VARIANT type : VT_ARRAY
| VT_UI1
     );
```

4. Installation

4.1. Package d'installation

Le package d'installation (3) contient les fichiers suivants :

Bibliothèque C	
OS	Le dossier contient le runtime pour les différents systèmes d'exploitation : ▪ ***Windows*** ▪ ***Linux :*** ▪ ▪
eidlib.h, eiddefines.h	Fichier en-tête de la bibliothèque C à inclure par les programmes C appelants
OS/eidlib. ***x***	Bibliothèque C à linker (points d'entrée bibliothèques partagées), où ***x*** est l'extension de la bibliothèque partagée ▪ ***LIB*** pour Visual C++ ▪
test.cpp	Programme de test en C
ActiveX	
Windows\VB	Exemple VB d'appel du contrôle ActiveX
Java	
Java/Test.java	Programme de test Java
Java/BEIDCard.html	Page de test JavaScript.

4.2. Runtime

Le runtime est nécessaire pour tous les environnements.

Il doit être installé suivant les instructions du document "Belgian eID Run-time Users guide".

4.3. Bibliothèque C

L'application doit inclure le fichier include du Toolkit "eidlib.h".

L'application doit être linkée avec la bibliothèque partagée "libeid.so" ou "eidlib.dll" si l'éditeur de liens supporte un lien direct avec une bibliothèque partagée (comme GCC) ou avec les points d'entrée de la bibliothèque "eidlib.lib" ou "libeid.a" dans les autres cas (comme Visual C++).

4.4. Java

Le Toolkit est compatible avec toutes les Java Virtual Machines de Sun à partir de 1.2.

Art. 3*bis*. Spécifications du middleware Crypto

1. But

Le but de ce document est à la fois d'esquisser l'architecture du middleware et de donner au programmeur d'applications les informations nécessaires pour développer des programmes utilisant la carte d'identité électronique belge. Ce document est limité aux informations concernant le middleware "carte belge", destinées aux programmeurs.

2. Architecture

2.1. Interfaces

Comme expliqué à la section précédente, deux interfaces sont mises en œuvre dans le middleware : l'une peut être appelée directement (l'interface PKCS#11), l'autre indirectement (CSP).

2.1.1 L'interface Crypto API

Microsoft(r) Cryptographic API 2.0 (CryptoAPI) permet aux développeurs d'applications d'ajouter l'authentification, le codage et le cryptage à leurs applications Win32(r). Les développeurs d'applications peuvent utiliser les fonctions de CryptoAPI sans rien savoir de la mise en œuvre sous-jacente, de même que l'on peut exploiter une bibliothèque graphique sans connaître les détails de la configuration du matériel graphique.

La partie CSP du middleware établit un lien entre l'interface abstraite CryptoAPI et l'interface PKCS#11 sous-jacente. Le développeur n'appellera jamais les fonctions de CSP directement, mais toujours via CryptoAPI.

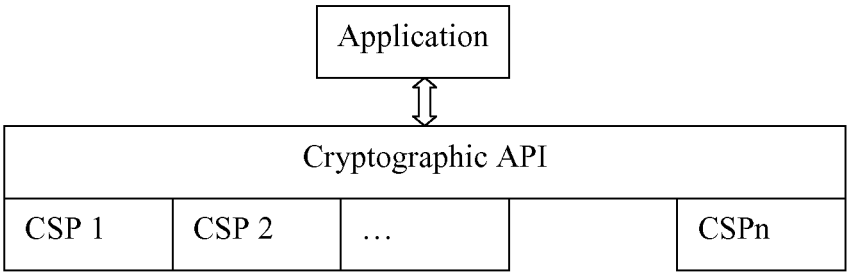
Actuellement, la carte d'identité belge ne supporte que les opérations de signature numérique. Plus tard, quand l'Etat belge décidera que l'utilisateur de la carte d'identité électronique peut stocker des clés sur la carte qui supporte le cryptage, le CSP sera étendu à cette fonctionnalité supplémentaire. De plus, la carte d'identité belge contient deux paires de clés pouvant servir à la signature numérique (authentification et non-répudiation).

Bien que le CSP ne supporte que la signature numérique, le programme est enregistré comme CSP du type PROV_RSA_FULL, afin de pouvoir être utilisé dans les applications standard Microsoft(r).

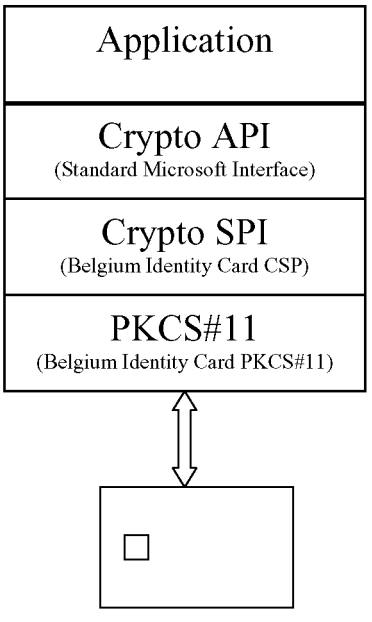
2.1.1.1. CSP - Architecture de haut niveau

L'API (application programming interface) Crypto de Microsoft est une interface abstraite avec les opérations cryptographiques. L'utilisateur de l'API n'est pas tenu de connaître le fonctionnement interne de la cryptographie mise en œuvre au niveau inférieur (celui de la carte). L'interface CryptoAPI est indépendante de la mise en œuvre cryptographique sous-jacente. La carte d'identité électronique belge est une carte à puce (" smartcard "). Dans d'autres applications, cependant, il peut s'agir d'une solution logicielle.

Sous l'interface CryptoAPI, une interface différente est spécifiée pour les concepteurs de systèmes de sécurité. Elle s'appelle CSPI (Cryptgraphic Service Provider Interface). L'interface CSPI isole le dispositif cryptographique sous-jacent de l'interface CryptoAPI. Le nombre de mises en œuvre CSPI sur un même système n'est pas limité. Chaque mise en œuvre CSPI est (normalement) propre à un type de matériel sous-jacent. Le schéma ci-dessous donne une vue d'ensemble de cette architecture :



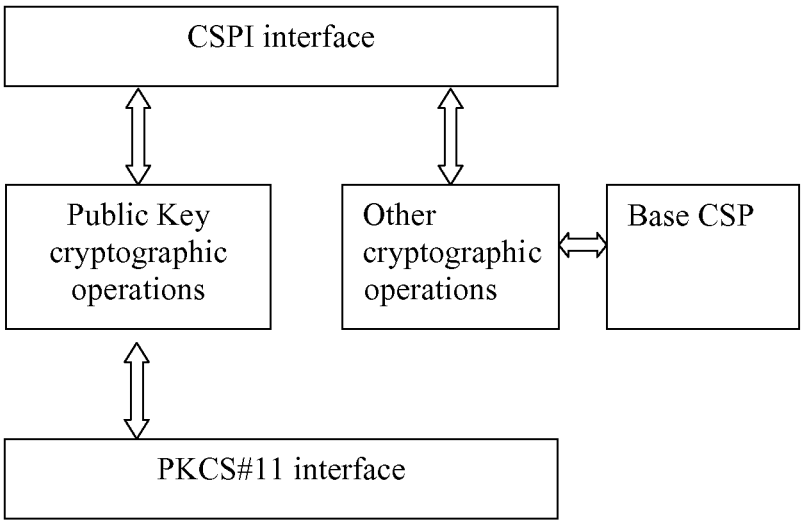
Normalement, chaque CSP s'oriente lui-même vers un type particulier de carte à puce. La plupart du temps, en effet, le CSP est mis à disposition par le fournisseur de la carte à puce lui-même. Le middleware de la carte d'identité belge adopte une approche différente. Pour donner à l'émetteur de la carte d'identité (en l'occurrence l'Etat) un maximum de flexibilité dans le choix du type de carte à puce, le CSP ne met pas en œuvre l'interface propriétaire de la carte à puce directement, mais passe par une interface PKCS#11 (pour en savoir plus sur l'interface PKCS#11, voyez la section 2.1.2). Voici un schéma des blocs concernés :



Si, à un stade ultérieur, le gouvernement décide de changer de type matériel de carte à puce, la mise en œuvre du CSP pourra rester en place (pourvu que la nouvelle carte à puce soit compatible avec la mise en œuvre existante).

2.1.1.2. CSP - Architecture de bas niveau

A un niveau inférieur, la mise en œuvre du CSP opère une traduction entre l'interface CSPI standard et l'interface PKCS#11. La figure ci-dessous esquisse cette architecture :



A l'échelle interne, le CSP met en œuvre une partie de l'interface PKCS#11 pour effectuer les opérations cryptographiques sur la clé publique. Ces opérations se limitent actuellement à la signature numérique (authentification et non-répudiation). Pour les opérations cryptographiques sur la clé privée (p.ex. le calcul du hachage), un CSP de base est utilisé. Normalement, l'on se sert pour cela du PROV_RSA_FULL CSP par défaut, en général celui de Microsoft.

Le CSP tient un fichier de configuration qui contient les paramètres indépendants de la carte d'identité électronique. Le nom (étiquette/label) des différentes clés, par exemple, est stocké avec une référence à la "key ID" correspondante. Ces informations sont communes à toutes les cartes d'identité électroniques belges. Il ne faut donc pas un fichier de configuration par carte utilisée sur un système donné.

Les certificats d'utilisateurs sont stockés par le système d'exploitation, dans un store "MY". Un nom convivial est défini pour chaque certificat (clé d'authentification et clé de signature). Dans l'environnement Microsoft, les conteneurs de certificat servent à établir un lien entre un certificat et la clé privée correspondante. Le nom de ces conteneurs est déduit de l'étiquette de la clé (authentification ou signature) et du numéro de série de la carte, afin de permettre l'utilisation de plusieurs cartes d'identité sur la même machine.

2.1.2. L'interface PKCS#11

L'interface PKCS#11 (v2.11) est utilisée par les applications non-Microsoft, par exemple Netscape. Les applications sur mesure peuvent, elles aussi, faire usage de cette interface au lieu de CryptoAPI. L'interface PKCS#11 est parfois appelée Cryptoki.

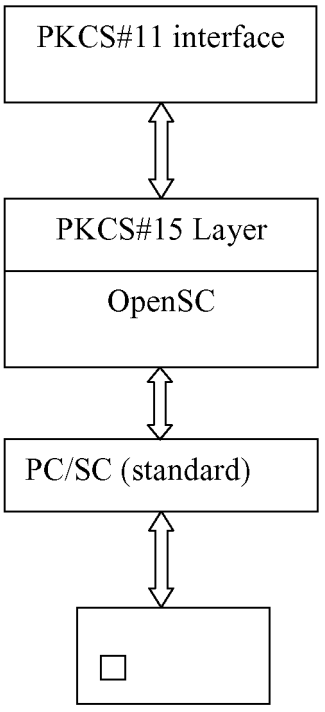
Le site web de RSA Laboratories (http://www.rsasecurity.com/rsalabs/pkcs/pkcs-11/) en donne une description détaillée.

Conformément au souci d'ouverture affiché par le gouvernement, nous avons décidé d'utiliser la mise en œuvre libre (" open source ") PKCS#11 de opensc (http ://www.opensc.org). Cette mise en œuvre ne se limite pas à l'interface PKCS#11 : elle supporte aussi la structure de carte PKCS#15.

2.1.2.1. PKCS#11 - Architecture de haut niveau

L'interface cryptoki est une interface abstraite avec la carte à puce, pour les applications qui ont besoin de la fonctionnalité cryptographique. Elle est normalement utilisée par les applications non-Microsoft telles que Netscape et Mozilla. Les fournisseurs de solutions indépendants peuvent aussi faire appel à cette interface pour mettre en œuvre des fonctionnalités cryptographiques dans des applications tierces.

Voici un schéma qui illustre les modules PKCS#11 :



Comme indiqué dans la figure ci-dessus, la couche supérieure met en œuvre l'interface PKCS#11 standard. C'est cette interface qui est exposée aux applications. Sous l'interface, le lien est établi avec la structure de carte dans PKCS#15, et les commandes nécessaires (APDU) sont envoyées via les fonctions d'interface standard PC/SC.

Dans la couche OpenSC, un pilote de carte spécifique est mis en œuvre pour utiliser la carte d'identité électronique belge. Si une carte différente est adoptée plus tard, ce pilote devra être remplacé.

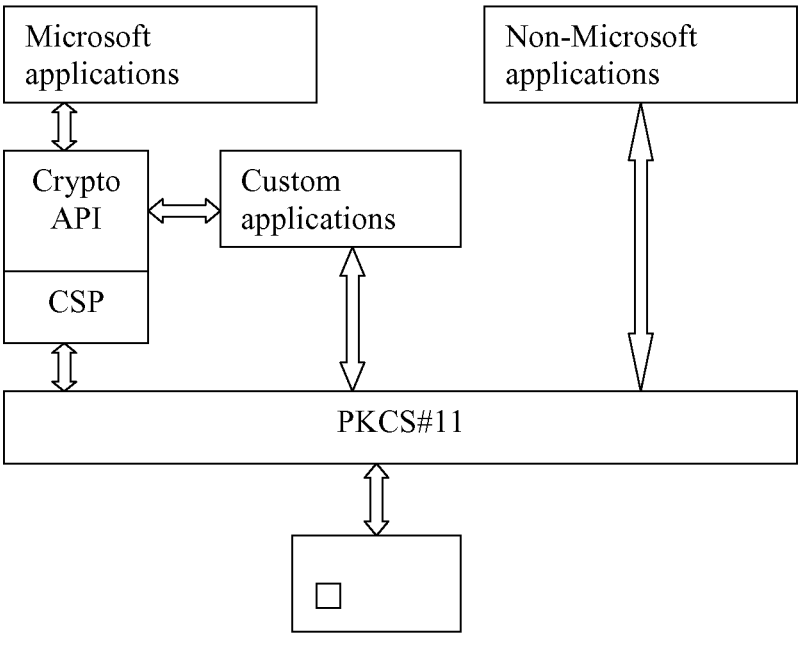
3. Guide de programmation

3.1. Introduction

Le middleware de la carte d'identité belge est un logiciel qui trouve place entre l'application mettant en œuvre les fonctions de sécurité (signature numérique) et le dispositif chargé des opérations cryptographiques proprement dites (la carte à puce).

Le middleware lui-même se compose de deux interfaces indépendantes (voir figure ci-dessous). Mais en dépit de cette indépendance, chacune fait appel à l'autre. Pour les applications standard Microsoft(r) (Office, Outlook...), un Cryptographic Service Provider (CSP) est créé pour effectuer les opérations cryptographiques à partir de la carte à puce. Une application ne fera jamais appel à cette mise en œuvre directement, mais via une interface standard appelée Crypto API. La mise en œuvre CSP fait usage de la deuxième interface, PKCS#11, qui est utilisée par les applications standard non-Microsoft.

Quand une nouvelle application est créée, il appartient au développeur de décider laquelle des deux interfaces servira à mettre les fonctions cryptographiques à la disposition de l'utilisateur.



Ce document explique à la fois les interfaces de programmation et la façon dont le concepteur d'applications peut s'en servir.

3.2. L'interface Crypto API

Microsoft(r) Cryptographic API 2.0 (CryptoAPI) permet aux développeurs d'applications d'ajouter l'authentification, le codage et le cryptage à leurs applications Win32(r). Les développeurs d'applications peuvent utiliser les fonctions de CryptoAPI sans rien savoir de la mise en œuvre sous-jacente, de même que l'on peut exploiter une bibliothèque graphique sans connaître les détails de la configuration du matériel graphique.

La partie CSP du middleware établit un lien entre l'interface abstraite CryptoAPI et l'interface PKCS#11 sous-jacente. Le développeur n'appellera jamais les fonctions de CSP directement, mais via CryptoAPI. Les sections ci-dessous décrivent les appels d'API que CryptoAPI transfère au CSP pour la suite du traitement. Elles ne contiennent pas d'information détaillée sur le fonctionnement de chaque appel d'API. Pour trouver ce type d'information, consultez le Microsoft Developer Network.

Actuellement, la carte d'identité belge ne supporte que les opérations de signature numérique. Les fonctions non liées aux opérations cryptographiques ne sont pas mises en œuvre. Plus tard, quand l'Etat belge décidera que l'utilisateur de la carte d'identité électronique peut stocker des clés sur la carte qui supporte le cryptage, le CSP sera étendu à cette fonctionnalité supplémentaire. De plus, la carte d'identité belge contient deux paires de clés pouvant servir à la signature numérique (authentification et non-répudiation). Pour ces raisons, certains paramètres passés aux fonctions Crypto API sont sans signification. Par exemple, dans l'appel CryptGetUserKey, un paramètre dwKeySpec est passé. Ce paramètre définit le type de clé à obtenir : une clé AT_KEYEXCHANGE ou une clé AT_SIGNATURE. Cependant, dans le cas du CSP Belgian Identity Card, ce paramètre ne suffit pas pour déterminer le type de signature à charger. Dans cette situation, le conteneur qui abrite le certificat correct doit être passé à CryptAcquireContext. Un appel à CryptGetUserKey est ensuite effectué avec succès.

Bien que le CSP ne supporte que la signature numérique, le programme est enregistré comme CSP du type PROV_RSA_FULL, afin de pouvoir être utilisé dans les applications standard Microsoft(r). Un appel aux fonctions Crypto API qui ne sont pas mises en œuvre dans le contexte de la signature numérique renvoie un code d'erreur indiquant que la fonction API n'est pas mise en œuvre.

Ce document est valable pour la version 1.20 du CSP.

3.2.1.CryptAcquireContext

```
BOOL WINAPI CryptAcquireContext(HCRYPTPROV *phProv,
                                LPCTSTR pszContainer,
                                LPCTSTR pszProvider,
                                DWORD dwProvType,
                                DWORD dwFlags);
```

Le paramètre pszContainer contient le nom du conteneur abritant une clé spécifique sur la carte d'identité. Le nom des conteneurs de la carte d'identité peut être obtenu via un appel à CryptGetProvParam.

Le paramètre dwFlags peut recevoir les valeurs suivantes (d'après MSDN) :

0 (équivalent à CRYPT_SCKEYSET)

CRYPT_VERIFYCONTEXT

CRYPT_NEWKEYSET

CRYPT_MACHINE_KEYSET

CRYPT_DELETEKEYSET

Le matériel des clés de la carte d'identité belge étant stocké sur une carte à puce et l'utilisateur n'étant pas autorisé à créer de nouveaux jeux de clés sur la carte, les valeurs CRYPT_NEWKEYSET, CRYPT_MACHINE_KEYSET et CRYPT_DELETEKEYSET ne sont pas supportées. L'utilisation de ces valeurs génère une erreur NTE_BAD_FLAGS.

Une valeur supplémentaire est définie pour ce paramètre : CRYPT_SCKEYSET. Par cette valeur, le développeur définit l'acquisition d'un contexte pour la clé suivant le contenu du paramètre pszContainer.

Pour les opérations de hachage seulement, un CSP de base est utilisé. Si, pour une raison quelconque, le CSP de base échoue, le code d'erreur suivant est généré par SetLastError() :

ERR_CANNOT_LOAD_BASE_CSP (0x1000)

3.2.2. CryptReleaseContext

```
BOOL WINAPI CryptReleaseContext(HCRYPTPROV hProv,
                                DWORD dwFlags);
```

Cette API est mise en œuvre suivant les prescriptions de MSDN.

3.2.3. CryptGenerateKey

```
BOOL WINAPI CryptGenKey( HCRYPTPROV hProv,
                         ALG_ID Algid,
                         DWORD dwFlags,
                         HCRYPTKEY *phKey);
```

Le matériel des clés étant préinstallé sur la carte d'identité belge et l'utilisateur n'étant pas autorisé à créer des paires de clés supplémentaires, cet appel d'API n'est pas mis en œuvre. L'appel de la fonction génère l'erreur E_NOTIMPL dans SetLastError ().

3.2.4. CryptDeriveKey

```
BOOL WINAPI CryptDeriveKey(HCRYPTPROV hProv,
                           ALG_ID Algid,
                           HCRYPTHASH hBaseData,
                           DWORD dwFlags,
                           HCRYPTKEY *phKey);
```

Le matériel des clés étant préinstallé sur la carte d'identité belge et l'utilisateur n'étant pas autorisé à créer des paires de clés supplémentaires, cet appel d'API n'est pas mis en œuvre. L'appel de la fonction génère l'erreur E_NOTIMPL dans SetLastError ().

3.2.5. CryptDestroyKey

```
BOOL WINAPI CryptDestroyKey(HCRYPTKEY hKey);
```

Le matériel des clés étant préinstallé sur la carte d'identité belge et l'utilisateur n'étant pas autorisé à créer des paires de clés supplémentaires, cet appel d'API n'est pas mis en œuvre. L'appel de la fonction génère l'erreur E_NOTIMPL dans SetLastError ().

3.2.6.CryptSetKeyParam

```
BOOL WINAPI CryptSetKeyParam(HCRYPTKEY hKey,
                             DWORD dwParam,
                             BYTE *pbData,
                             DWORD dwFlags);
```

Le matériel des clés étant préinstallé sur la carte d'identité belge et l'utilisateur n'étant pas autorisé à créer des paires de clés supplémentaires, cet appel d'API n'est pas mis en œuvre. L'appel de la fonction génère l'erreur E_NOTIMPL dans SetLastError ().

3.2.7.CryptGetKeyParam

```
BOOL WINAPI CryptGetKeyParam(HCRYPTKEY hKey,
                             DWORD dwParam,
                             BYTE *pbData,
                             DWORD *pcbData,
                             DWORD dwFlags);
```

Le matériel des clés étant préinstallé sur la carte d'identité belge et l'utilisateur n'étant pas autorisé à créer des paires de clés supplémentaires, cet appel d'API n'est pas mis en oeuvre. L'appel de la fonction génère l'erreur E_NOTIMPL dans SetLastError ().

3.2.8.CryptSetProvParam

```
BOOL WINAPI CryptSetProvParam(HCRYPTPROV hProv,
                              DWORD dwParam,
                              BYTE *pbData,
                              DWORD dwFlags);
```

D'après la documentation MSDN, le paramètre dwParam peut recevoir les valeurs suivantes :

PP_CLIENT_HWND

PP_KEYSET_SEC_DESCR

Ce dernier est sans objet, vu que le matériel des clés de la carte d'identité belge est stocké dans la carte à puce plutôt que dans le registre. Il est donc ignoré.

3.2.9.CryptGetProvParam

```
BOOL WINAPI CryptGetProvParam(HCRYPTPROV hProv,
                              DWORD dwParam,
                              BYTE *pbData,
                              DWORD *pcbData,
                              DWORD dwFlags);
```

Cet appel d'API est mis en œuvre conformément à la documentation MSDN, à l'exception du paramètre PP_KEYSET_SEC_DESCR, qui est ignoré.

Pour le paramètre PP_IMPTYPE, la valeur renvoyée est CRYPT_IMPL_MIXED parce que l'opération de signature est assumée par le matériel (la carte à puce), tandis que le hachage est confié au service cryptographique de base.

3.2.10. CryptSetHashParam

```
BOOL WINAPI CryptSetHashParam(HCRYPTHASH hHash,
                              DWORD dwParam,
                              BYTE *pbData,
                              DWORD dwFlags);
```

Cet appel d'API est mis en œuvre conformément à la documentation MSDN.

Le paramètre dwParam = HP_HASHVAL est mis en œuvre mais doit être utilisé avec prudence. Il a pour but de permettre aux applications de signer les valeurs de hachage sans devoir accéder aux données de base. L'application (et a fortiori l'utilisateur) n'ayant aucune idée de ce qui est signé, l'opération présente des risques intrinsèques.

3.2.11. CryptGetHashParam

```
BOOL WINAPI CryptGetHashParam(HCRYPTHASH hHash,
                              DWORD dwParam,
                              BYTE *pbData,
                              DWORD *pcbData,
                              DWORD dwFlags);
```

Cet appel d'API est mis en œuvre conformément à la documentation MSDN.

3.2.12.CryptExportKey

```
BOOL WINAPI CryptExportKey(HCRYPTKEY hKey,
                HCRYPTKEY hExpKey,
                DWORD dwBlobType,
                DWORD dwFlags,
                BYTE *pbData,
                DWORD *pcbDataLen);
```

Cette fonction peut servir à exporter la clé publique associée au paramètre hKey. Un handle vers une clé publique peut être obtenu par un appel à CryptGetUserKey. Les clés privées étant stockées sur une carte à puce et l'exportation des clés privées n'étant pas permise, le seul dwBlobType permis est PUBLICKEYBLOB. Seules les clés publiques peuvent être exportées. Le paramètre hExpKey n'est donc pas utilisé et doit contenir NULL. La clé publique est renvoyée dans le paramètre pbData. Pour obtenir la longueur des données renvoyées, le paramètre pbData peut être mis à NULL. La longueur des données à renvoyer est alors placée dans pcbDataLen. Si le tampon passé à la fonction est trop petit, l'erreur ERROR_MORE_DATA est renvoyée et la longueur correcte du tampon est placée dans le paramètre pcbDataLen.

3.2.13. CryptImportKey

```
BOOL WINAPI CryptImportKey(HCRYPTPROV hProv,
                BYTE *pbData,
                DWORD dwDataLen,
                HCRYPTKEY hPubKey,
                DWORD dwFlags,
                HCRYPTKEY *phKey);
```

Le matériel des clés étant préinstallé sur la carte d'identité belge et l'utilisateur n'étant pas autorisé à créer des paires de clés supplémentaires, cet appel d'API n'est pas mis en œuvre. L'appel de la fonction génère l'erreur E_NOTIMPL dans SetLastError ().

3.2.14.CryptEncrypt

```
BOOL WINAPI CryptEncrypt(HCRYPTKEY hKey,
                HCRYPTHASH hHash,
                BOOL Final,
                DWORD dwFlags,
                BYTE *pbData,
                DWORD *pcbData,
                DWORD cbBuffer);
```

Actuellement, l'utilisation des clés définie par le gouvernement belge ne supporte pas le cryptage. Cet appel d'API n'est donc pas mis en œuvre. L'appel de la fonction génère l'erreur E_NOTIMPL dans SetLastError ().

S'il devient possible ultérieurement d'ajouter à la carte d'identité électronique des clés supportant le cryptage, cette fonction sera mise en œuvre.

3.2.15.CryptDecrypt

```
BOOL WINAPI CryptDecrypt(HCRYPTKEY hKey,
                HCRYPTHASH hHash,
                BOOL Final,
                DWORD dwFlags,
                BYTE *pbData,
                DWORD *pcbData);
```

Actuellement, l'utilisation des clés définie par le gouvernement belge ne supporte pas le cryptage. Cet appel d'API n'est donc pas mis en œuvre. L'appel de la fonction génère l'erreur E_NOTIMPL dans SetLastError ().

S'il devient possible ultérieurement d'ajouter à la carte d'identité électronique des clés supportant le cryptage, cette fonction sera mise en œuvre.

3.2.16. CryptCreateHash

```
BOOL WINAPI CryptCreateHash(HCRYPTPROV hProv,
                ALG_ID Algid,
                HCRYPTKEY hKey,
                DWORD dwFlags,
                HCRYPTHASH *phHash);
```

Cet appel d'API est mis en œuvre conformément à la documentation MSDN. Une erreur supplémentaire peut être renvoyée via SetLastError () :

ERR_INVALID_PROVIDER_HANDLE (0x1001)

Cette erreur signifie que le handle spécifié par hProv n'a pas été trouvé (c.-à-d. qu'il n'a pas été créé avec CryptAcquireContext ()).

Le traitement de cet appel est délégué à un CSP de base.

3.2.17. CryptHashData

```
BOOL WINAPI CryptHashData(HCRYPTHASH hHash,
                          BYTE *pbData,
                          DWORD cbData,
                          DWORD dwFlags);
```

Cet appel d'API est mis en œuvre conformément à la documentation MSDN. Une valeur (autre que 0) peut être spécifiée dans le paramètre dwFlags : CRYPT_USERDATA. Selon le CSP de base retenu, l'appel peut être mis en œuvre ou non. Dans Microsoft Base CSP, par exemple, il ne l'est pas.

Le traitement de cet appel est délégué à un CSP de base.

3.2.18.CryptHashSessionKey

```
BOOL WINAPI CryptHashSessionKey(HCRYPTHASH hHash,
                                HCRYPTKEY hKey,
                                DWORD dwFlags);
```

Certains appels sous-jacents nécessaires à l'utilisation de cette fonction n'étant pas mis en œuvre actuellement par ce CSP, l'appel n'est pas disponible. L'appel de la fonction génère l'erreur E_NOTIMPL dans SetLastError ().

3.2.19.CryptSignHash

```
BOOL WINAPI CryptSignHash(HCRYPTHASH hHash,
                          DWORD dwKeySpec,
                          LPCTSTR sDescription,
                          DWORD dwFlags,
                          BYTE *pbSignature,
                          DWORD *pdwSigLen);
```

Cet appel d'API est mis en œuvre conformément à la documentation MSDN. L'appel de la fonction déclenche une tentative de connexion et de log-on sur la carte d'identité belge (carte à puce). Si une de ces opérations échoue, l'erreur suivante peut être renvoyée via SetLastError () :

ERR_CANNOT_LOGON_TO_TOKEN (0x1004)

Pour signer les données de hachage, il faut lire certaines informations sur la carte (p.ex. longueur de la clé). Si une erreur se produit durant l'opération, le code d'erreur suivant est généré via SetLastError () :

ERR_CANNOT_GET_TOKEN_SLOT_INFO (0x1003)

Le mécanisme utilisé pour générer les signatures numériques s'appelle CKM_RSA_PKCS. Pour plus de détails à ce sujet, voyez la documentation PKCS#11.

Actuellement, voici les algorithmes de hachage qui peuvent servir à signer les données : MD2, MD4, MD5, SHA-1 et SSL3 SHAMD5. Bien que les algorithmes de hachage MDx soient encore disponibles pour la rétrocompatibilité, il est recommandé d'utiliser SHA-1 dans les nouvelles applications.

3.2.20.CryptDestroyHash

```
BOOL WINAPI CryptVerifySignature(HCRYPTHASH hHash,
                                 BYTE *pbSignature,
                                 DWORD dwSigLen,
                                 HCRYPTKEY hPubKey,
                                 LPCTSTR sDescription,
                                 DWORD dwFlags);
```

Cet appel d'API est mis en œuvre conformément à la documentation MSDN.

3.2.21.CryptVerifySignature

```
BOOL WINAPI CryptVerifySignature(HCRYPTHASH hHash,
                                 BYTE *pbSignature,
                                 DWORD dwSigLen,
                                 HCRYPTKEY hPubKey,
                                 LPCTSTR sDescription,
                                 DWORD dwFlags);
```

Cette fonction est mise en oeuvre pour des raisons de facilité. L'appel est délégué au CSP de base.

3.2.22.CryptGenRandom

```
BOOL WINAPI CryptGenRandom(HCRYPTPROV hProv,
                           DWORD dwLen,
                           BYTE *pbBuffer);
```

Cet appel d'API est mis en œuvre conformément à la documentation MSDN. Les données introduites via pbBuffer serviront à ensemencer la génération aléatoire.

3.2.23.CryptGetUserKey

```
BOOL CryptGetUserKey(HCRYPTPROV hProv,
        DWORD dwKeySpec,
        HCRYPTKEY *phUserKey);
```

Cet appel renvoie un handle vers une clé publique du conteneur de clés défini dans CryptAcquireContext. Il ne suffit pas de spécifier AT_SIGNATURE pour le paramètre dwKeySpec, parce qu'avec cette information, le CSP ne peut déterminer quelle clé de signature il faut renvoyer. La clé à charger doit donc d'abord être spécifiée dans CryptAcquireContext.

3.2.24.CryptDuplicateHash

```
BOOL WINAPI CryptDuplicateHash(HCRYPTHASH hHash,
                        DWORD *pdwReserved,
                        DWORD dwFlags,
                        HCRYPTHASH phHash);
```

Cet appel d'API est mis en œuvre conformément à la documentation MSDN.

3.2.25.CryptDuplicateKey

```
BOOL WINAPI CryptDuplicateKey(HCRYPTKEY hKey,
                        DWORD *pdwReserved,
                        DWORD dwFlags,
                        HCRYPTKEY* phKey);
```

Vu que le matériel des clés est stocké sur une carte à puce et ne peut en être extrait, cette fonction est sans objet. Cet appel d'API n'est donc pas mis en œuvre. L'appel de la fonction génère l'erreur E_NOTIMPL dans SetLastError ().

3.3. L'interface PKCS#11

L'interface PKCS#11 (v2.11) est utilisée par les applications non-Microsoft, par exemple Netscape. Les applications sur mesure peuvent, elles aussi, faire usage de cette interface au lieu de CryptoAPI. L'interface PKCS#11 est parfois appelée Cryptoki.

Sa description détaillée se trouve sur le site de RSA Laboratories (http ://www.rsasecurity.com/rsalabs/pkcs/pkcs-11/).

3.3.1. Appels d'API mis en oeuvre

3.3.1.1. Fonctions générales

C_Initialize,

C_Finalize

C_GetInfo

C_GetFunctionList

3.3.1.2. Fonctions de gestion de slot et de token

C_GetSlotList

C_GetSlotInfo

C_GetTokenInfo

C_GetMechanismList

C_GetMechanismInfo

C_WaitForSlotEvent

C_SetPin

3.3.1.3. Fonctions de gestion de session

C_OpenSession

C_CloseSession

C_CloseAllSessions

C_GetSessionInfo

C_Login

C_Logout

3.3.1.4. Fonctions de gestion d'objets

C_FindObjectsInit

C_FindObjects

C_FindObjectsFinal

C_GetAttributeValue

3.3.1.5. Fonctions de signature

C_SignInit

C_Sign

C_SignUpdate

C_SignFinal

3.3.1.6. Fonctions de digest

C_DigestInit

C_Digest

C_DigestUpdate

C_DigestFinal

3.3.1.7. Fonctions de génération aléatoire (à confirmer bientôt)

C_SeedRandom

C_GenerateRandom

3.3.2. Mécanismes de signature supportés

Pour les signatures :

- CKM_RSA_X_509

- CKM_RSA_PKCS : hachages ASN.1-wrapped et purs (MD5, SHA1, SHA1+MD5, RIPEMD160 - si 20 octets sont donnés, le hachage est censé être du type SHA-1)

- CKM_RIPEMD160_RSA_PKCS, CKM_SHA1_RSA_PKCS, CKM_MD5_RSA_PKCS

- S'ils sont supportés par la carte d'identité électronique, les mécanismes de signature suivants seront aussi supportés par le middleware : CKM_RSA_PKCS_PSS, CKM_SHA1_RSA_PKCS_PSS

Pour les digests :

CKM_SHA_1, CKM_RIPEMD160, CKM_MD5

3.3.3. Informations de slot et de token

Il y aura un slot/token virtuel pour chaque PIN (dans le cas de la carte d'identité électronique belge, cela veut donc dire 1 slot/token).

Les clés publiques, les clés privées et les certificats correspondants posséderont le même attribut d'objet CKA_ID.

3.3.4. Comportement en cas de lecteur à clavier PIN

Dans ce cas, CK_TOKEN_INFO aura le drapeau CKF_PROTECTED_AUTHENTICATION_PATH levé, et l'application renverra un NULL PIN avec C_Login. A l'appel de C_Login, la bibliothèque PKCS#11 présentera à l'utilisateur un dialogue l'invitant à introduire son PIN au clavier PIN, en vue de la signature ou de l'authentification.

3.3.5. Comportement en cas de clé de non-répudiation

Si une signature est demandée avec cette clé, la bibliothèque PKCS#11 elle-même affichera une interface graphique invitant l'utilisateur à introduire son PIN ou à le taper au clavier du lecteur.

4. Références

[ISO/IEC 7816-1] Identification Cards - Integrated Circuit Cards with Contacts Part1 : Physical Characteristics

[ISO/IEC 7816-2] Identification Cards - Integrated Circuit Cards with Contacts Part2 : Dimensions and Location of Contacts

[ISO/IEC 7816-3] Identification Cards - Integrated Circuit Cards with Contacts Part3 : Electronic Signals and Transmission Protocols

[ISO/IEC 7816-4] Identification Cards - Integrated Circuit Cards with Contacts Part4 : Inter-industry Commands for Interchange

[ISO/IEC 7816-5] Identification Cards - Integrated Circuit Cards with Contacts Part5 : Numbering System for Application Identifiers

[CEN/CWA 14170] Electronic Signature - Security Requirements Secure Signature Creation Applications

[CEN/CWA 14171] Electronic Signature - Security Requirements Secure Signature Verification Applications

[RSAS/PKCS#11] Public Key Cryptographic Standards Cryptographic Token Interface Standard

[RSAS/PKCS#15] Public Key Cryptographic Standards Cryptographic Token Information Standard

[EID/READERS 2.7.3] Belgian Electronic Identity Card Readers Technical Compatibility v 2.7.3

[EID/CRYPTO 1.4.0] Belgian Electronic Identity Card Security & Crypto Middleware v 1.4.0

[EID/IDENTITY 1.0.0] Belgian Electronic Identity Card Data & Identity Middleware v 1.0.0

[EID/CARD 2.0.0] Belgian Electronic Identity Card Card and Applet Specifications v 2.0.0

[EN45014] Conformity Assessment Generic Criteria (CE)

[EN60950] Information Processing Equipment Security

[EN50081] EMC (Electro-Magnetic Compatibility) Emission Generic Criteria

[EN50081] EMC (Electro-Magnetic Compatibility) Immunity Generic Criteria

[EN55022] REC (Radio-Electric Compatibility) Perturbations Generic Criteria

Vu pour être annexé à Notre arrête du 7 décembre 2006 fixant les specifications et la procedure d'enregistrement des appareils de lecture pour la carte d'identité éléctronique et modifiant l'arrêté royal du 13 février 1998 relatif aux specifications des appareils de lecture de la carte d'identité sociale.

ALBERT

Par le Roi :

Le Ministre de l'Intérieur,
P. DEWAEL

Le Ministre de l'Economie
M. VERWILGHEN

Le Ministre des Affaires sociales,
R. DEMOTTE

La Ministre des Classes moyennes et de l'Agriculture,
Mme S. LARUELLE

Le Ministre de l'Emploi,
P. VANVELTHOVEN

Annexe II

FORMULAIRE TYPE POUR L'ENREGISTREMENT D'UN APPAREIL DE LECTURE DE LA CARTE D'IDENTITE ELECTRONIQUE BELGE (eID)

Mode d'emploi

Ce document comprend l'inventaire des différents documents ainsi que le relevé des données à compléter et à transmettre à l'autorité en vue de l'obtention d'un numéro d'enregistrement pour un appareil de lecture de la carte d'identité électronique belge. Tous les points décrits et toutes les informations demandées doivent faire l'objet d'une réponse aussi complète que possible.

Un modèle du formulaire d'enregistrement peut être téléchargé à partir de l'URL suivant : http://eid.belgium.be

Le formulaire d'enregistrement dûment complété, y compris toutes les annexes requises, doit être transmis :

a) par la poste à l'adresse suivante :

Service public fédéral Technologie de l'Information et de la Communication

rue Marie Thérèse 1-3

1000 Bruxelles

b) par e-mail :

enregistrement@fedict.be

I. IDENTIFICATION DU DEMANDEUR

L'identification du demandeur de l'enregistrement de la solution complète, c'est-à-dire, la personne responsable de l'intégration de la solution.

Dénomination complète de l'entreprise : .....

.....

Adresse du siège social : .....

.....

Forme juridique : .....

.....

Personne de contact : .....

.....

N° de téléphone de la personne de contact : .....

.....

e-mail : .....

.....

II. DESCRIPTION DE LA SOLUTION

Dans cette partie, la solution introduite pour enregistrement doit être décrite en précisant les caractéristiques essentielles et l'environnement y afférent.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

III. Spécifications normatives

Dans cette partie, il y a lieu d'indiquer si les normes imposées sont respectées et de joindre en annexe, pour chaque rubrique, la preuve y afférente, soit sur la base d'un certificat de conformité remis par un organisme de certification, soit sur la base d'une déclaration écrite de conformité

*** TABLEAU ***

IV. RESULTATS DES SCENARIOS TEST

V. MANUELS

Fait à ..... le, ...

Nom : .....

Signature :.....

Vu pour être annexé à Notre arrête du 7 décembre 2006 fixant les specifications et la procedure d'enregistrement des appareils de lecture pour la carte d'identité éléctronique et modifiant l'arrêté royal du 13 février 1998 relatif aux specifications des appareils de lecture de la carte d'identité sociale.

ALBERT

Par le Roi :

Le Ministre de l'Intérieur,

P. DEWAEL

Le Ministre de l'Economie,

M. VERWILGHEN

Le Ministre des Affaires sociales,

R. DEMOTTE

La Ministre des Classes moyennes et de l'Agriculture,

Mme S. LARUELLE

Le Ministre de l'Emploi,

P.VANVELTHOVEN

Notes

(1) Certaines cartes ne contiennent que les deux premiers prénoms ensemble. Dans ce cas, les deux prénoms sont renvoyés dans ce champ et le champ " 2ème prénom " reste vide.

(2) Certaines cartes contiennent aussi le numéro et le numéro de boîte dans le même champ. Dans ce cas, tout est renvoyé dans ce champ et les autres champs restent vides.

(3) Le package peut se trouver dans un fichier archive (ZIP, TAR ou GZ).

Bijlage I

Inhoudsopgave

Art. 3.

Specificaties Identity Middleware

1. Inleiding
   - 1.1. Matrix ontwikkelingsomgeving
2. Functionele specificaties
   - 2.1. Versies en compatibiliteit
   - 2.2. Ingeven PIN
   - 2.3. Opmerking over de maximumlengte van de parameters
   - 2.4. Multi-threaded applicatie
   - 2.5. API-organisatie
   - 2.6. Functies voor initialisatie en beëindiging
     - 2.6.1. BEID_Init
     - 2.6.2. BEID_Exit
   - 2.7. Identiteitsfuncties
     - 2.7.1. BEID_GetID
     - 2.7.2. BEID_GetAddress
     - 2.7.3. BEID_GetPicture
     - 2.7.4. Offline identiteitsfuncties
       - 2.7.4.1. BEID_GetRawData
       - 2.7.4.2. BEID_SetRawData
   - 2.8. Algemene high-level functies
     - 2.8.1. BEID_BeginTransaction
     - 2.8.2. BEID_EndTransaction
     - 2.8.3. BEID_SelectApplication
     - 2.8.4. BEID_ReadFile
     - 2.8.5. BEID_WriteFile
     - 2.8.6. BEID_VerifyPIN
     - 2.8.7. BEID_ChangePIN
     - 2.8.8. BEID_GetPINStatus
   - 2.9. Low-level functies
     - 2.9.1. BEID_GetVersionInfo
     - 2.9.2. BEID_SendAPDU
     - 2.9.3. BEID_FlushCache
   - 2.10. Identificatie PIN
     - 2.10.1. Types PIN
     - 2.10.2. Gebruik PIN
   - 2.11. OCSP en CRL input policy parameters
   - 2.12. Status van de functies
     - 2.12.1. Algemene return-codes
   - 2.13. Controle handtekening en validering certificaten
     - 2.13.1. Controle handtekening
     - 2.13.2. Controle certificaat en validatieresultaten
     - 2.13.3. Genruikte OCSP et CRL policies
3. Programmeerinterfaces
   - 3.1. C API
     - 3.1.1. Structures
     - 3.1.2. Functions
   - 3.2. Java API
     - 3.2.1. Data Classes
     - 3.2.2. Main Classe
     - 3.2.3. Applet Classe
   - 3.3. ActiveX API
     - 3.3.1. Interfacedefinities
     - 3.3.2. Functies
4. Installatie
   - 4.1. Installatiepakket
   - 4.2. Runtime
   - 4.3. C Library
   - 4.4. Java

Art. 3*bis*

Specificaties Crypto Middleware

1. Doel
2. Architectuur
   - 2.1. Interfaces
     - 2.1.1. De CRYPTO API-Interface
       - 2.1.1.1. CSP high-level architectuur
       - 2.1.1.2. CSP low-level architectuur
     - 2.1.2. De PKCS(11-interface
       - 2.1.2.1. PKCS(11 high-level architectuur
3. Programmeurshandleiding
   - 3.1. Inleiding
   - 3.2. De Crypto API-Interface
     - 3.2.1. CryptAcquireContext
     - 3.2.2. CryptReleaseContext
     - 3.2.3. CryptGenerateKey
     - 3.2.4. CryptDeriveKey
     - 3.2.5. CryptDestroyKey
     - 3.2.6. CryptSetKeyParam
     - 3.2.7. CryptGetKeyParam
     - 3.2.8. CryptSetProvParam
     - 3.2.9. CryptGetProvParam
     - 3.2.10. CryptSetHashParam
     - 3.2.11. CryptGetHashParam
     - 3.2.12. CryptExportKey
     - 3.2.13. CryptImportKey
     - 3.2.14. CryptEncrypt
     - 3.2.15. CryptDecrypt
     - 3.2.16. CryptCreateHash
     - 3.2.17. CryptHashData
     - 3.2.18. CryptHashSessionKey
     - 3.2.19. CryptSignHash
     - 3.2.20. CryptDestroyHash
     - 3.2.21. CryptVerifySignature
     - 3.2.22. CryptGenRandom
     - 3.2.23. CryptGetUserKey
     - 3.2.24. CryptDuplicateHash
     - 3.2.25. CryptDuplicateKey
   - 3.3. De PKCS(11-Interface
     - 3.3.1. Geïmplementeerde API calls
       - 3.3.1.1. Algemene functies
       - 3.3.1.2. Functies voor het beheer van slot en token
       - 3.3.1.3. Functies voor het beheer van sessies
       - 3.3.1.4. Functies voor het beheer van objecten
       - 3.3.1.5. Functies voor ondertekening
       - 3.3.1.6. Digest-functies
       - 3.3.1.7. Functies voor random-generatie (worden binnenkort bevestigd)
     - 3.3.2. Ondersteunde handtekeningmechanismen
     - 3.3.3. Slot- en token-informatie
     - 3.3.4. Gedrag in het geval van een pinpad-lezer
     - 3.3.5. Gedrag met een niet-verwerpingssleutel
4. Referenties

Art. 1. Registratieprocedure lezers

1. Inleiding

1.1. Doel van de bijlagen

Dit document bevat de specificaties en beschrijft de vereisten waaraan moet worden voldaan om het label ″Belgische eID compatibel″ voor kaartlezers te verwerven.

Vooraleer het registratieproces te starten, dient de aanvrager (vb. producent, wederverkoper, distributeur of integrator van kaartlezers) intern een aantal taken/scenario's uit te voeren waarop de conformiteitscertificaten, de benchmarkinformatie en de uiteindelijke validering zullen worden gebaseerd :

1) De noodzakelijke ISO-accreditatiecertificaten verwerven waarmee de ISO-organen technische conformiteit garanderen - zie hoofdstuk ″Technische specificaties″.

2) De low-level testscenario's uitvoeren (intern met eigen apparatuur) en de gevraagde benchmarkinformatie ophalen - zie hoofdstuk ″Low-level scenario's″.

3) De high-level valideringsscenario's uitvoeren (online via de eID-testwebsite) en de gevraagde bevestigingsinformatie ophalen - zie hoofdstuk ″High-level scenario's″.

De aanvrager dient dan het registratieformulier in te vullen (met alle gevraagde gedetailleerde productinformatie) en in te dienen bij de registratieautoriteit voor kaartlezers.

Opmerking1 : het registratieformulier bevat een conformiteitsverklaring waarin de producent bevestigt dat de aanvrager alle vorige stappen heeft uitgevoerd en dat de informatie in de verklaring correct is.

Opmerking 2 : de registratieautoriteit voor kaartlezers behoudt zich het recht voor de conformiteit van de kaartlezer verder te controleren (bijvoorbeeld in het geval van klachten). Samen met het registratieformulier worden er drie exemplaren van het product waarnaar in het registratieformulier wordt verwezen, ingediend voor archivering en controle.

1.2. Reikwijdte

Dit document betreft de registratie van kaartlezers. Met kaartlezers bedoelen we elk apparaat dat een SmartCard-interface voor de eID-kaart biedt (apart of niet, geïntegreerd of niet, toepassingsspecifiek of niet).

Dit document heeft geen betrekking op de specifieke vereisten in verband met kaartlezers die worden gebruikt voor het beheer van kaarten (activering van de kaart, (her-)initialisatie PIN/PUK, wijziging van gegevens, enz.) die specifiek zijn voor de autoriteit die de kaart aflevert (bijvoorbeeld initiële activering kaart) en/of het deblokkeren van de kaart (bijvoorbeeld ingeval er te veel foute PIN-codes werden ingevoerd) en/of het wijzigen van de kaart (bijvoorbeeld ingeval het adresbestand moet worden bijgewerkt). Dit type lezers vereist speciale PIN/PUK-samenvoegfuncties die buiten het kader van dit document vallen.

1.3. Lezerscategorieën

De lezers worden verder ondergebracht in categorieën op basis van de volgende criteria :

- connectable (connecteerbaar) : als de lezer over een hostinterface beschikt (vb. USB, RSR232, PCMCIA, enz.);

- secure display (beveiligd scherm) : als de lezer over een specifiek display beschikt voor de visualisatie van berichten;

- secure pinpad (beveiligd pinpad) : als de lezer over een specifiek pinpad beschikt voor het ingeven van de PIN-code;

secure application (beveiligde applicatie) : als de lezer een beveiligd communicatiekanaal heeft geïnstalleerd met de piloting-applicatie;

	Value Checker	Transparante lezer	Secure display lezer	Secure pinpad lezer	Trusted lezer
Connectable		X	X	X	X
Secure display			X		X
Secure pinpad				X	X
Secure application					X

- Value checkers : niet-geconnecteerde lezer waarmee de inhoud van de chip gemakkelijk kan worden gelezen, maar waarmee geen cryptografische transacties kunnen worden uitgevoerd (niet onderworpen aan registratie).

- Transparante lezers : connecteerbare lezer zonder specifiek display of pinpad (de PIN- en statuscodes worden ingevoerd en getoond via niet-beveiligde media zoals het toetsenbord en het scherm).

- Secured lezers : connecteerbare lezer met specifiek display en/of pinpad (de PIN- en statuscodes worden direct ingevoerd via het pinpad en/of op het display getoond).

- Trusted lezers : beveiligde lezers met beveiligde applicaties die over een beveiligd kanaal beschikken om met de lezer te communiceren.

1.4. Platformen

Aangezien de ondersteuning van de eID-kaart platformspecifiek is, zal de registratie per platform worden gevraagd (de lezer kan uiteraard voor verschillende platformen worden geregistreerd via meervoudige registratie).

De volgende platformen worden momenteel ondersteund, maar de registratieautoriteit behoudt zich het recht voor platformen toe te voegen/te schrappen naargelang van de vraag op de markt, de ondersteuning van de leverancier, enz.

Microsoft	**Macintosh**	**Linux (2.x)**	
Windows 98	MacOSX10.1	I386	
Windows ME	MacOSX10.2		
Windows 2000	MacOSX10.3		
Windows XP			

Aangezien er platformen in talrijke varianten verkrijgbaar zijn, gaan we ervan uit dat de tests/scenario's worden uitgevoerd op een basisversie van de eenvoudigste versie van een bepaald platform (vb. de home edition en niet de professional edition, de personal edition en niet de server edition, enz.) waarop alle momenteel beschikbare (beveiligings-) patches werden geïnstalleerd. Als de lezer specifieke varianten en/of patches vereist, moeten deze duidelijk in het registratieformulier worden opgegeven en zullen ze op de website met de lijst van de geregistreerde lezers worden vermeld.

Om een gemeenschappelijke interface voor de applicaties te bieden, moet de lezer over een set API- implementaties beschikken (vb. deel van het installatiepakket of direct vooraf geïnstalleerd). Sommige van deze APIs zijn platformspecifiek (vb. Microsoft CryptoAPI), sommige zijn specifiek Belgisch (vb. eID runtime) - zie hoofdstuk "Specificaties".

De registratieautoriteit behoudt zich het recht voor het testscenario aan te passen om de technische evoluties van platformspecifieke interfaces en standaarden te volgen.

1.5 Omgeving

Aangezien de beveiligingsvereisten van de eID-kaart omgevingspecifiek zijn, zal de registratie tevens een onderscheid maken tussen de lezers voor een thuis-/werk-/mobiele omgeving en lezers voor een openbare omgeving - zie hoofdstuk "Beveiligingsvereisten".

Lezers die in openbare plaatsen worden geïnstalleerd, en niet onder de controle van de Belgische burgers vallen, moeten namelijk aan speciale beveiligingsvereisten voldoen, die als volgt kunnen worden samengevat :

- beschikbaarheid van een secure pinpad om de vertrouwelijkheid van de PIN-code te waarborgen;

- beschikbaarheid van een secure display om de juistheid van de berichten te waarborgen.

Er zullen twee "Belgische eID compatibel"-logo's worden gebruikt om deze twee lezerscategorieën van elkaar te onderscheiden.

2. Vereisten en specificaties

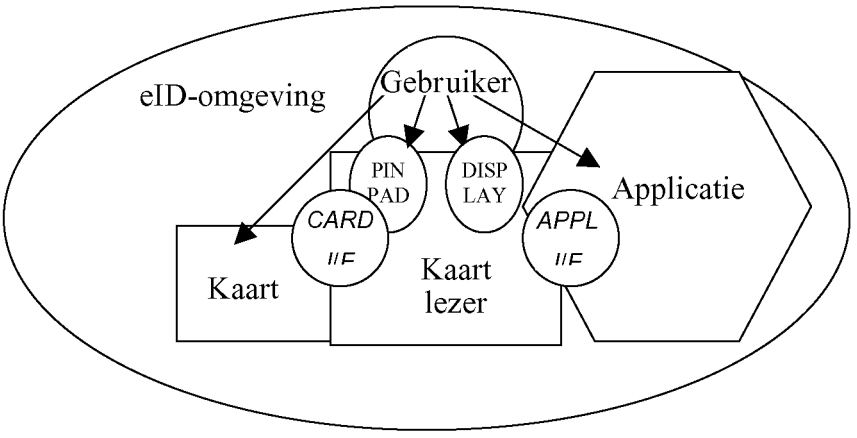
2.1. Model- en beveiligingsvereisten

Een eID-kaart-compatibele omgeving kan worden opgesplitst in een eindgebruiker die met een eID-compatibel systeem werkt. Het eID-systeem kan verder worden opgesplitst in 3 delen :

- de eID-kaart zelf

- de eID-kaart-compatibele lezer

- de eID-kaart-compatibele applicatie



De eID-kaart-compatibele applicatie bevat alle applicatiespecifieke componenten, die onder meer vereist zijn voor het presenteren van documenten, het bekijken van gegevens/attributen, het formatteren van handtekeningen, enz. In één fysieke eenheid kunnen verschillende applicatie-instanties aanwezig zijn die dezelfde lezer delen.

De eID-kaart-compatibele lezer moet alle noodzakelijke algemene componenten (hardware en/of software) bevatten om de eID-kaart te interfacen in overeenstemming met de beveiligingsvereisten die worden beschreven in [CEN/CWA 14170] en omgezet in de technische specificaties [EID/READERS 2.7.3] en [EID/CRYPTO 1.4.0] en [EID/IDENTITY 1.0.0]. Meer bepaald :

- PINPAD (User Input & Authentication Component - Component voor input van de gebruiker en authenticatie) : deze component biedt beveiligde authenticatiefuncties waarmee de eindgebruiker PIN-codes ingeeft op een zodanige manier dat ze kunnen worden vergeleken met de PIN die zich op de kaart bevindt. In het algemeen kunnen we een onderscheid maken tussen lezers die over een specifiek pinpad beschikken en lezers die gebruik maken van het algemene toetsenbord (numpad). Deze component moet voldoen aan de veiligheidsvereisten die worden beschreven in [CEN/CWA 14170 - Hoofdstuk 13] en omgezet in de technische specficaties [EID/READERS 2.7.3 - Hoofdstuk 2&3]

- DISPLAY (User Output & Control Component - Component voor output van de gebruiker en controle) : deze component biedt beveiligde interactiefuncties waarmee de eindgebruiker de applicatie gebruikt om het uitvoeringsproces te controleren en waarover foutcodes en statusberichten worden verstuurd. In het algemeen kunnen we een onderscheid maken tussen lezers die over een specifiek display beschikken en lezers die gebruik maken van een algemeen scherm. Deze component moet voldoen aan de veiligheidsvereisten die worden beschreven in [CEN/CWA 14170 - Hoofdstuk 12] en omgezet in de technische specificaties [EID/READERS 2.7.3 - Hoofdstuk 2&3]

- APPLI/F (Application Input/Output Interface Component - Interfacecomponent voor input/output met de applicatie) : deze component biedt beveiligde functies voor datamanipulatie waarmee de applicatie met de eID-lezer samenwerkt om op een genormaliseerde en beveiligde manier gegevens te verzenden en te ontvangen. Deze component beschikt meestal over een algemene interface naar de applicatie. Deze component moet voldoen aan de veiligheidsvereisten die worden beschreven in [CEN/CWA 14170 - Hoofdstuk 15] en omgezet in de technische specificaties [EID/CRYPTO 1.4.0] en [EID/IDENTITY 1.0.0].

- CARDI/F (Card Input/Output Interface Component - Interfacecomponent voor input/output met de kaart) : deze component biedt interactiefuncties waarmee de applicatie [ISO/IEC 7816] commando's uitwisselt met de kaart. Deze component moet voldoen aan de veiligheidsvereisten die worden beschreven in [CEN/CWA 14170 - Hoofdstuk 16] en omgezet in de technische specficaties [EID/READERS 2.7.3 - Hoofdstuk 1 & Hoofdstuk 2.1]

2.2. Vereisten voor de kaartinterface

De kaartinterface dient te voldoen aan [ISO7816] met de parameters die worden beschreven in de specificaties [EID/READER 2.7.3 Hoofdstuk 1 & Hoofdstuk 2.1].

De interface dient tevens te voldoen aan :

- de beveiligingsvereisten die worden beschreven in [EN60950];

- de algemene criteria betreffende elektromagnetische emissie die worden beschreven in [EN50081];

- de algemene criteria betreffende elektromagnetische immuniteit die worden beschreven in [EN50081];

- de algemene criteria betreffende radio-elektrische storing die worden beschreven in [EN55022].

Opmerking : als de lezer over een tweegleuvensysteem beschikt, dient dit te beantwoorden aan de SIS/SAM-kaartlezerspecificaties (zie www.ksz-bcss.fgov.be).

2.3. Vereisten voor de gebruikersinterface

Het interactieproces met de gebruiker kan plaatsvinden in twee verschillende soorten fysieke omgevingen waarbij het eID-systeem door verschillende organisaties wordt gecontroleerd :

Openbare plaats : het eID-systeem bevindt zich op een openbare plaats, zoals een station, een bankagentschap, een postkantoor of een andere plaats die wordt bediend door een dienstverlener die niet noodzakelijk een relatie heeft met, of onder controle staat van de eindgebruiker. Zonder aanvullende technische veiligheidsmaatregelen is deze omgeving kwetsbaar voor een aantal gevaren (vb. vervanging door een vals eID-systeem). De technische vereisten voor eID-systemen in een dergelijke openbare omgeving zullen dus strenger moeten zijn.

Thuis- en werkomgeving : het eID-systeem bevindt zich thuis of op kantoor, waar de gebruiker directe controle heeft over het eID-systeem (vb. gsm). In dit geval kan aan de beveiligingsvereisten worden voldaan door organisatorische maatregelen van de gebruiker en kunnen de technische middelen om de beveiligingsvereisten te realiseren minder streng zijn.

Naargelang van het gebruik zal de kaartlezer ook moeten voldoen aan verschillende vereisten en specificaties die hieronder worden samengevat :

- lezers die in een openbare plaats worden geïnstalleerd, dienen over een specifiek secure PINpad en display te beschikken die in de lezer zelf zijn geïntegreerd (het standaard-toetsenbord en -scherm mogen niet worden gebruikt voor gebruikersinteracties);

- de lezers die thuis en/of in de werkomgeving worden geïnstalleerd, mogen gebruik maken van het algemene toetsenbord en scherm voor gebruikersinteracties.

2.4. Vereisten voor de applicatie-interface

De federale overheid heeft een eID runtime-omgeving ontwikkeld die gratis ter beschikking wordt gesteld van elke eindgebruiker en die een set libraries, tools en executables bevat om een generieke interface te creëren voor applicaties die met een eID-kaart willen communiceren/interfacen.

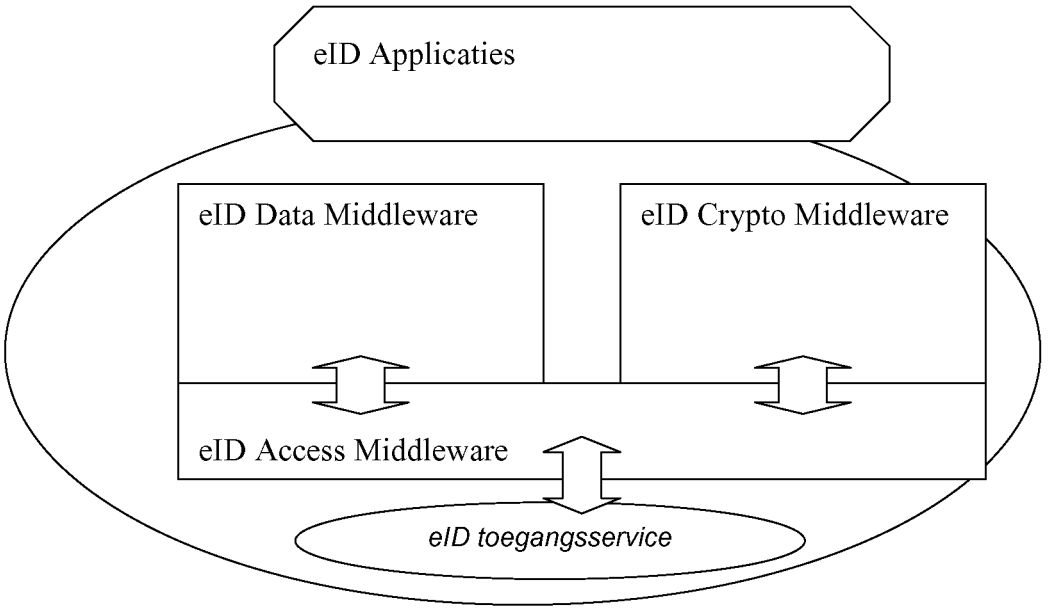
Deze omgeving bestaat uit vier componenten :

- de eID data-middleware die een gestandaardiseerde interface voor data-extractie biedt, die de applicaties in staat stelt om de functies voor datacaptatie van de kaart te gebruiken;

- de eID crypto-middleware die een gestandaardiseerde interface voor cryptografische bewerkingen biedt, die de applicaties in staat stelt om de functies voor authenticatie en ondertekening van de kaart te gebruiken;

- de eID access-middleware die door de twee andere componenten wordt gebruikt om veilig verbinding te maken en te communiceren met een eID-toegangsservice;

- de eID-toegangsservice die door een eID access-middleware wordt gebruikt om verbinding te maken met de fysieke lezer (via een eenvoudige kabel of via een netwerk).



De eID runtime-omgeving is uiteraard platformspecifiek en er is een versie beschikbaar voor alle platformen die worden ondersteund. Er zullen verschillende releases van de eID-omgeving worden ontwikkeld om de releases van de eID-kaart in de loop der jaren te volgen.

Opmerking : afhankelijk van het platform kan de Crypto Middleware over verschillende interfaces beschikken om aan fabrikantspecifieke vereisten te voldoen :

- Linux : alleen PKCS#11 is vereist;

- Microsoft : PKCS#11 en Microsoft CryptoAPI zijn vereist;

- MacOS : PKCS#11 en Apple CryptoAPI zijn vereist.

De Belgische overheid gaat ervan uit dat de eID runtime-omgeving voldoet aan alle eisen die aan de component applicatie-interface worden gesteld. De eID runtime-omgeving werd door de Belgische overheid bovendien beveiligd met een code zodat er alleen authentieke kopieën kunnen worden verspreid. Er is ook een gratis begeleidende kit voor softwareontwikkeling beschikbaar met voorbeelden van en richtlijnen voor het creëren van een interface met de eID runtime-omgeving.

De eID runtime-omgeving wordt beschouwd als een deel van de lezer, wat tot een van de volgende alternatieven leidt :

- ofwel wordt de eID-lezer door de eindgebruiker als een aparte component aangeschaft (vb. connecteerbare lezer) en dan moet er met de lezer een set eID runtime-installatietools en gebruikers-/installatiehandleidingen worden geleverd. Het installatiepakket moet zowel de lezerspecifieke drivers als de eID runtime-omgeving bevatten (het gedeelte in verband met de eID runtime kan optioneel zodanig worden ingesteld dat de eindgebruiker zelf kan beslissen of hij het wil installeren);

- ofwel wordt de eID-lezer aangeschaft als onderdeel van een ander product (vb. een pc met een SmartCard-lezer) en dan moet de eID runtime vooraf worden geïnstalleerd. Er moeten ook eID runtime-installatiepakketten en gebruikers-/installatiehandleidingen worden geleverd (voor het geval de eID runtime opnieuw moet worden geïnstalleerd).

Opmerking : als er geen runtime-omgeving beschikbaar is voor een specifieke lezer (platform niet ondersteund en/of identity middleware en crypto middleware direct geïnstalleerd in de firmware van het toestel), dan mag de aanvrager zelf een runtime-omgeving ontwikkelen (of de officiële runtime-omgeving aanpassen aan zijn omgeving). In een dergelijk geval behoudt de registratieautoriteit zich het recht voor de overeenstemming en de gelijkwaardigheid met de officiële eID runtime-omgeving te controleren.

2.5. Specifieke vereisten

Het moet voor de eindgebruiker mogelijk zijn om de eID runtime-omgeving te installeren/opnieuw te installeren/te verwijderen/te upgraden als er een nieuwe versie beschikbaar is op de eID-website. Om die reden zal deze registratieprocedure geen runtime-omgevingen van derden en/of gewijzigde runtime-omgevingen ondersteunen tenzij dit contractueel met de registratieautoriteit werd overeengekomen.

Eens geïnstalleerd, moet het voor de eindgebruiker mogelijk zijn om de lezer fysiek los te koppelen/aan te koppelen in plug & play mode zonder het systeem te moeten herstarten en/of rebooten. Deze vereiste geldt enkel voor connecteerbare lezers die thuis of in de werkomgeving worden gebruikt.

Men moet gemakkelijk kunnen nagaan of de lezer werd geregistreerd volgens de in dit document beschreven procedure (vb. via een sticker). Men moet ook gemakkelijk het merk en het type van de lezer kunnen controleren, vooral als de lezer in een ander toestel werd geïntegreerd. Het merk en het type moeten duidelijk worden vermeld in de documentatie en op de zichtbare delen van de behuizing.

3. Low-level testscenario's

Deze scenario's dienen door de leverancier van de eID-lezer te worden uitgevoerd om de compatibiliteit met de eID-kaart te testen. Deze scenario's worden in twee delen georganiseerd naargelang van het platform :

- Het eerste deel is een set ISO7816 commando's die moeten worden uitgevoerd en getest via de SendAPDU functie-call van de eID runtime-omgeving.

- Het tweede deel is een set datacaptatiefuncties die moeten worden uitgevoerd en getest via de GetXXX functie-calls van de eID runtime-omgeving.

- Het derde deel is een set Crypto-functies die moeten worden uitgevoerd en getest via de Crypto Middleware (PKCS#11 implementatie) van de eID runtime-omgeving.

- Het vierde deel (alleen vereist voor Microsoft- en Apple-platformen) is een set Crypto-functies die moeten worden uitgevoerd en getest via de Crypto Middleware (CryptoAPI implementatie) van de eID runtime-omgeving.

De specificaties van de kaartinhoud en interfaces zijn beschikbaar in :

- [EID/CRYPTO 1.4.0] voor de Crypto-elementen (sleutels, certificaten, PIN's, enz.) en Crypto Middleware-specificaties;

- [EID/IDENTITY 1.0.0] voor de gegevenselementen (identiteit, adres, foto) en data middleware-specificaties;

- [EID/CARD 2.0.0] voor de ondersteunde ISO7816 commando's en eID-kaart- interfacespecificaties.

3.1 ISO7816/APDU-scenario

Precondities :

- SmartCard-lezer correct geïnstalleerd en geconfigureerd
- Identiteits-library correct geïnstalleerd en geconfigureerd
- Testkaart in de lezer

```
long handle = 0;
BEID_Pin pin = {0};
   pin.id = 0x01;
   pin.pinType = BEID_PIN_TYPE_PKCS15;
   pin.usageCode = BEID_USAGE_AUTH;
BEID_Bytes sendBytes = {(unsigned char *)"\x00\xA4\x08\x0C\x06\x3F\x00\xDF\x01\x40\x32", 11};
BEID_Bytes respBytes = {0};
…
assert (BEID_Init( NULL, 0, 0, &handle ).general==BEID_OK);                          // Init
assert (BEID_BeginTransaction().general==BEID_OK);                                   // Begin transaction
assert (BEID_SendAPDU( &sendBytes, &pin, &respBytes ).general==BEID_OK); // Select SGNID
assert (BEID_EndTransaction().general==BEID_OK);                                     // End transaction
assert (BEID_FlushCache().general==BEID_OK);                                         // Flush Cache
assert (BEID_Exit().general==BEID_OK);                                               // Exit
```

Postcondities :

- alle assertions geslaagd
- de volgende informatie moet in het registratieformulier worden gerapporteerd :
- kaartrespons op de geselecteerde SGNID

3.2. Datacaptatiescenario

3.2.1. Kaartversie en RRN-certificaat

Precondities :

- SmartCard-lezer correct geïnstalleerd en geconfigureerd
- Identiteits-library correct geïnstalleerd en geconfigureerd
- Testkaart in de lezer

```
long handle = 0;
BEID_VersionInfo     version = {0};
BEID_Bytes                   certRRN = {0};
BEID_Bytes                   bytesBuffer = {0};
BEID_Bytes           tAID = { (unsigned char *)"\xA0\x00\x00\x01\x77\x50\x4B\x43\x53\x2D\x31\x35", 12 };
BEID_Bytes                   tFileID = { (unsigned char *)"\x50\x3C", 2 };                   // RRN
…
assert (BEID_Init( NULL, 0, 0, &handle ).general==BEID_OK);                          // Init
assert (BEID_GetVersionInfo( &version, 0, &bytesBuffer ).general==BEID_OK); // Read Version
assert (BEID_SelectApplication( &tAID ).general==BEID_OK);                           // Select application
assert (BEID_ReadFile( &tFileID, &certRRN, 0x00 ).general==BEID_OK);         // Read RRN Cert
assert (BEID_Exit().general==BEID_OK);                                               // Exit
```

Postcondities :

- alle assertions geslaagd
- de volgende informatie moet in het registratieformulier worden gerapporteerd :
- informatie versie (serienummer, componentcode, nummer en versie OS, nummer en versie Softmask, versie Applet, versie Global OS, versie applet interface, PKCS#1 support, versie Key Exchange, levenscyclus applicatie, grafische personalisering, elektronische personalisering, elektronische personalisering interface)
- RRN-certificaat

3.2.2. Identiteit, adres en foto

Precondities :

- SmartCard-lezer correct geïnstalleerd en geconfigureerd
- Identiteits-library correct geïnstalleerd en geconfigureerd
- Testkaart in de lezer

```
long handle = 0;
BEID_ID_Data                 identityData = {0};
BEID_Address                 addressData = {0};
BEID_Certif_Check    certifCheck = {0};
BEID_Bytes                   pictureData = {0};
…
assert (BEID_Init( NULL, 0, 0, &handle ).general==BEID_OK);                          // Init
assert (BEID_GetID(&identityData, &certifCheck).general==BEID_OK);           // Read Identity Data
assert (BEID_GetAddress(&addressData, &certifCheck).general==BEID_OK);       // Read Address Data
assert (BEID_GetPicture( &pictureData, &certifCheck ).general==BEID_OK);     // Read Picture Data
assert (BEID_Exit().general==BEID_OK);                                               // Exit
```

Postcondities :

- alle assertions geslaagd

- de volgende informatie moet in het registratieformulier worden gerapporteerd :

- identiteitsgegevens (nummer kaart en chip, geldigheid, gemeente, RRN, voor- en familienaam, geboorteplaats en -datum, nationaliteit, geslacht, adellijke titel)

- adresgegevens (straat, nummer, busnummer, postcode, stad/gemeente, land)

3.2.3. PIN controleren

Precondities :

- SmartCard-lezer correct geïnstalleerd en geconfigureerd

- Identiteits-library correct geïnstalleerd en geconfigureerd

- Testkaart in de lezer

```
long handle = 0;
long triesLeft = 0;
BEID_Pin pin = {0};
   pin.id = 0x01;
   pin.pinType = BEID_PIN_TYPE_PKCS15;
   pin.usageCode = BEID_USAGE_AUTH;
…
assert (BEID_Init( NULL, 0, 0, &handle ).general==BEID_OK);                    // Init
assert (BEID_VerifyPIN(&pin, NULL, &triesLeft ).general==BEID_OK);             // Verify PIN
assert (BEID_Exit().general==BEID_OK);                                         // Exit
```

Postcondities :

- alle assertions geslaagd

- de volgende informatie moet in het registratieformulier worden gerapporteerd :

- als de lezer een secure pinpad-lezer is, werd de PIN in het pinpad ingegeven

- als de lezer geen secure pinpad-lezer is, werd de PIN op het scherm ingegeven

3.3. Cryptoki/PKCS#11-scenario

3.3.1. Initialiseren library, slot en token

Precondities :

- SmartCard-lezer correct geïnstalleerd en geconfigureerd

- Cryptoki/PKCS#11 library correct geïnstalleerd en geconfigureerd

- Testkaart in de lezer

```
CK_INFO          libInfo;                          // PKCS#11 / Cryptoki Library Information
CK_ULONG         slotCount;                        // Number of Slots available
CK_SLOT_ID_PTR   pSlotList;                        // List of Slots available
CK_SLOT_ID       slotID;                           // Identifier of Slot to be tested
CK_SLOT_INFO     slotInfo;                         // Slot Information
CK_TOKEN_INFO    tokenInfo;                        // Token Information
…
assert (C_Initialize(NULL)==CKR_OK);               // initialize the PKCS#11 / Cryptoki Library
assert (C_GetInfo(&libInfo)==CKR_OK);              // Get PKCS#11 / Cryptoki Library Information
assert (C_GetSlotList(CK_FALSE, NULL_PTR, &slotCount)==CKR_OK); // Get List of Slots / Readers
pSlotList = (CK_SLOT_ID_PTR) malloc(slotCount*sizeof(CK_SLOT_ID));
assert (C_GetSlotList(CK_TRUE, pSlotList, &slotCount)==CKR_OK);   // Get List of Slots with token
slotID = pSlotList[0];                             // Slot to be tested (with card inserted)
assert (C_GetSlotInfo(slotID, &slotInfo)==CKR_OK);             // Get selected slot information
assert (C_GetTokenInfo(slotID, &tokenInfo)==CKR_OK);  // Get selected token information
…
```

...

Postcondities :

- alle assertions geslaagd

- de volgende informatie moet in het registratieformulier worden gerapporteerd :

- informatie over de library (versie cryptoki, id. producent, versie en beschrijving library)

- informatie over slot/lezer (beschrijving slot, id. producent, hardwareversie, firmwareversie)

- informatie over token/kaart (token label en model, id. producent, serienummer, hardwareversie, firmwareversie)

3.3.2. Handtekening aansturen met authenticatiesleutel

Precondities : vorige test geslaagd (library, slot en token geïnitialiseerd)

```
CK_SESSION_HANDLE       hSession;
CK_OBJECT_HANDLE        hObject, hAuthKey, hAuthCert, hIssuerCert, hRootCert;
CK_ATTRIBUTE            authKeyTemplate[], authCertTemplate[], issuerCertTemplate[], rootCertTemplate[];
CK_BYTE                 application, digest[20], signature[128], data[] = {'S','A','M','P','L','E'};
CK_ULONG                objectCount, digestLen, signatureLen;
CK_MECHANISM            mecHash = {CKM_SHA_1,NULL_PTR,0},
                        mecSign = {CKM_SHA1_RSA_PKCS,NULL_PTR,0};
…
assert (C_OpenSession(slotID, CKF_SERIAL_SESSION,                    // open authentication session
                      (CK_VOID_PTR) &application, NULL_PTR,&hSession)==CKR_OK);
assert (C_FindObjectsInit(hSession, NULL_PTR, 0)==CKR_OK);  // find certificates and keys
while (C_FindObjects(hSession, &hObject, 1, &ObjectCount)==CKR_OK) {
        if (ObjectCount == 0) break;
        if (C_GetAttributeValue(hSession, hObject, authKeyTemplate, 1)==CKR_OK) hAuthKey=hObject;
        if (C_GetAttributeValue(hSession, hObject, authCertTemplate, 1)==CKR_OK) hAuthCert=hObject;
        if (C_GetAttributeValue(hSession, hObject, issuerCertTemplate, 1)==CKR_OK) hIssuerCert=hObject;
        if (C_GetAttributeValue(hSession, hObject, rootCertTemplate, 1)==CKR_OK) hRootCert=hObject;
        }
assert (C_FindObjectsFinal(hSession)==CKR_OK);
if (C_DigestInit(hSession, &mecHash)==CKR_OK) {                      // Compute Message Digest
        if (C_DigestUpdate(hSession, data, sizeof(data))==CKR_OK) {
                digestLen = sizeof(digest);
                assert (C_DigestFinal(hSession, digest, &digestLen)==CKR_OK);
        }
if (C_SignInit(hSession, &mecSign, hObject)==CKR_OK) {               // compute Signature
        if (C_SignUpdate(hSession, digest, sizeof(digest))==CKR_OK) {
                signatureLen = sizeof(signature);
                assert (C_SignFinal(hSession, signature, &signatureLen)==CKR_OK);
                }
        }
```

Postcondities :

- alle assertions geslaagd

- de volgende informatie moet in het registratieformulier worden gerapporteerd : certificaten, digest en handtekening

3.3.3. PIN wijzigen

Precondities : vorige test geslaagd (sessie geïnitialiseerd)

```
CK_SESSION_HANDLE       hSession;
…
CK_UTF8CHAR oldPin[] = {'1','2','3','4'};
CK_UTF8CHAR newPin[] = {'6','5','4','3','2','1'};
…
assert (C_SetPIN(hSession, oldPin, sizeof(oldPin), newPin, sizeof(newPin))==CKR_OK);

assert (C_CloseSession(hSession)==CKR_OK);                           // close authentication session
```

Postcondities :

- alle assertions geslaagd

3.3.4. Handtekening aansturen met handtekeningsleutel

Precondities : vorige test geslaagd (library, slot en token geïnitialiseerd)

```
CK_SESSION_HANDLE       hSession;
CK_OBJECT_HANDLE        hObject, hSignKey, hSignCert, hIssuerCert, hRootCert;
CK_ATTRIBUTE            signKeyTemplate[], signCertTemplate[], issuerCertTemplate[], rootCertTemplate[];
CK_BYTE                 application, digest[20], signature[128], data[] = {'S','A','M','P','L','E'};
CK_ULONG                objectCount, digestLen, signatureLen;
CK_MECHANISM            mecHash = {CKM_SHA_1,NULL_PTR,0},
                        mecSign = {CKM_SHA1_RSA_PKCS,NULL_PTR,0};
…
assert (C_OpenSession(slotID, CKF_SERIAL_SESSION,                    // open signature session
                      (CK_VOID_PTR) &application, NULL_PTR,&hSession)==CKR_OK);
assert (C_FindObjectsInit(hSession, NULL_PTR, 0)==CKR_OK);  // find certificates and keys
while (C_FindObjects(hSession, &hObject, 1, &ObjectCount)==CKR_OK) {
        if (ObjectCount == 0) break;
        if (C_GetAttributeValue(hSession, hObject, signKeyTemplate, 1)==CKR_OK) hSignKey=hObject;
        if (C_GetAttributeValue(hSession, hObject, signCertTemplate, 1)==CKR_OK) hSignCert=hObject;
        if (C_GetAttributeValue(hSession, hObject, issuerCertTemplate, 1)==CKR_OK) hIssuerCert=hObject;
        if (C_GetAttributeValue(hSession, hObject, rootCertTemplate, 1)==CKR_OK) hRootCert=hObject;
        }
assert (C_FindObjectsFinal(hSession)==CKR_OK);
if (C_DigestInit(hSession, &mecHash)==CKR_OK) {                      // Compute Message Digest
        if (C_DigestUpdate(hSession, data, sizeof(data))==CKR_OK) {
                digestLen = sizeof(digest);
                assert (C_DigestFinal(hSession, digest, &digestLen)==CKR_OK);
        }
if (C_SignInit(hSession, &mecSign, hObject)==CKR_OK) {               // compute Signature
        if (C_SignUpdate(hSession, digest, sizeof(digest))==CKR_OK) {
                signatureLen = sizeof(signature);
                assert (C_SignFinal(hSession, signature, &signatureLen)==CKR_OK);
                }
        }
```

Postcondities :

* alle assertions geslaagd

* de volgende informatie moet in het registratieformulier worden gerapporteerd : certificaten, digest en handtekening

3.3.5. PIN wijzigen

Precondities : vorige test geslaagd (sessie geïnitialiseerd)

```
CK_SESSION_HANDLE       hSession;
…
CK_UTF8CHAR oldPin[] = {'6','5','4','3','2','1'};
CK_UTF8CHAR newPin[] = {'1','2','3','4'};
…
assert (C_SetPIN(hSession, oldPin, sizeof(oldPin), newPin, sizeof(newPin))==CKR_OK);

assert (C_CloseSession(hSession)==CKR_OK);                           // close signature session
```

Postcondities :

- alle assertions geslaagd

4. High-level valideringsscenario's

Deze scenario's dienen door de leverancier van de eID-lezer te worden uitgevoerd om de compatibiliteit met de eID-kaart te testen. Deze scenario's worden georganiseerd in vier delen die achtereenvolgens worden uitgevoerd (het ene deel maakt gebruik van het resultaat van het vorige deel) :

- Installatie, desinstallatie, update, herconfiguratie.

- Sterke authenticatie.

- Datacaptatie en wijzigen PIN.

- Gekwalificeerde handtekening.

Voor elk valideringsscenario wordt een set precondities, controledoelstellingen en postcondities beschreven om de compatibiliteit van de lezer en de bijbehorende software te beoordelen.

Opmerking : het is mogelijk dat een gedeelte van deze scenario's niet moet worden toegepast, afhankelijk van :

- pakket lezer (vb. desinstallatie/herinstallatie is uiteraard niet van toepassing op lezers die in een pc zijn geïntegreerd)

- plug&play (vb. afkoppelen/opnieuw aankoppelen is uiteraard niet van toepassing op lezers die in andere hardwarecomponenten zoals een toetsenbord zijn geïntegreerd).

- CryptoAPI (vb. ondersteuning van eigen/platformspecifieke middleware is alleen van toepassing op het overeenkomstige platform).

4.1. Scenario I : installatie/desinstallatie en Plug&Play

4.1.1. Precondities

* Geactiveerde testkaart beschikbaar (met gekende PIN-codes)

* Lokale pc vooraf geïnstalleerd (met het doelplatform/os), internet browser en aangepaste drivers voor de lezer en eID runtime-omgeving

* Internetaansluiting geconfigureerd met port 80(http) & 443(https) beschikbaar.

* Gebruikers-/installatiehandleiding beschikbaar

4.1.2. Controledoelstellingen

* De kaartlezer installeren volgens de instructies in de gebruikershandleiding (vb. reboot, administratierechten, enz.)

* [indien van toepassing] De kaartlezer desinstalleren en opnieuw installeren volgens de instructies in de gebruikershandleiding (vb. reboot, administratierechten, enz.)

* [indien van toepassing] De kaartlezer aankoppelen en afkoppelen

* eID-validering datacaptatie

==> CD1 : Applicatie voor datacaptatie opstarten

==> CD3 : eID-kaart in de lezer steken

==> CD2 : Data capteren

==> CD5 : eID-kaart verwijderen

==> CD6 : Data capteren

==> CD7 : eID-kaart in de lezer steken (op verzoek)

* [indien van toepassing] Een tweede kaartlezer van hetzelfde type installeren

* [indien van toepassing] De tweede kaartlezer aan- en afkoppelen

* eID-validering datacaptatie (uit te voeren op beide lezers)

==> CD1 : Applicatie voor datacaptatie opstarten

==> CD3 : eID-kaart in de lezer steken

==> CD2 : Data capteren

==> CD5 : eID-kaart verwijderen

==> CD6 : Data capteren

==> CD7 : eID-kaart in de lezer steken (op verzoek)

4.1.3. Postcondities

* Set versienummers configuratie (OS/lezer/runtime, enz.)

* Screenshots van tool voor datacaptatie

4.2. Scenario II : sterke authenticatie

4.2.1. Precondities

* Scenario I met succes uitgevoerd

* eID-validerings-/testsite beschikbaar

* Internet browsers vooraf geïnstalleerd (SSLv3-compatibele browser met PKCS#11-interface)

* [indien van toepassing] Internet browsers vooraf geïnstalleerd (SSLv3-compatibele browser met CryptoAPI-interface)

* Scenario II geselecteerd

4.2.2. Controledoelstellingen

* Naar de testservice gaan via https met PKCS#11-compatibele browser :

==> CD1 : browser configureren om eID PKCS#11-interface te herkennen

==> CD2 : Server geauthenticeerd

==> CD3 : Client-certificaat ″authenticatie″ geselecteerd

==> CD4 : PIN-code gevraagd

==> CD5 : Authenticatie geslaagd

* [indien van toepassing] Naar de testservice gaan via https met CryptoAPI-compatibele browser :

==> CD1 : browser configureren door certificaten eindgebruiker in de cert. store te laden

==> CD2 : Server geauthenticeerd

==> CD3 : Client-certificaat ″authenticatie″ geselecteerd

==> CD4 : PIN-code gevraagd

==> CD5 : Authenticatie geslaagd

4.2.3. Postcondities

* Data met succes op de valideringssite gelogd (OS/browser/interface/kaart/enz.)

* Berichten die op het display verschijnen moeten worden gerapporteerd.

* Kaartnummer, gegevens en bij benadering de duur van de transactie naar de valideringssite moeten worden gerapporteerd

* Screenshot van de laatste pagina van de eID-testsite

4.3. Scenario III : datacaptatie en wijziging PIN

4.3.1. Precondities

* Scenario II met succes uitgevoerd

* Beveiligde verbinding gemaakt met de valideringssite

* ScenarioIII geselecteerd

4.3.2. Controledoelstellingen

* Naar de testservice gaan via https met PKCS#11-compatibele browser :

==> CD1 : Server geauthenticeerd

==> CD2 : Client-certificaat ″authenticatie″ geselecteerd

==> CD3 : PIN-code gevraagd

==> CD4 : Authenticatie geslaagd

==> CD5 : Datacaptatie automatisch ingeroepen

==> CD6 : Gecapteerde data getoond

==> CD7 : Wijzigen PIN-code

==> CD8 : Wijziging geslaagd

* [indien van toepassing] Naar de testservice gaan via https met CryptoAPI-compatibele browser :

==> CD1 : Server geauthenticeerd

==> CD2 : Client-certificaat ″authenticatie″ geselecteerd

==> CD3 : PIN-code gevraagd

==> CD4 : Authenticatie geslaagd

==> CD5 : Datacaptatie automatisch ingeroepen

==> CD6 : Gecapteerde data getoond

==> CD7 : Wijzigen PIN-code

==> CD8 : Wijziging geslaagd

4.3.3. Postcondities

* Data met succes op de valideringssite gelogd (OS/browser/interface/kaart/enz.)

* Berichten die op het display verschijnen moeten worden gerapporteerd.

* Kaartnummer, gegevens en bij benadering de duur van de transactie naar de valideringssite moeten worden gerapporteerd

* Screenshot van de laatste pagina van de eID-testsite

4.4. Scenario IV : elektronische handtekening

4.4.1. Precondities

* Scenario III met succes uitgevoerd

* Beveiligde verbinding gemaakt met de valideringssite

* Scenario IV geselecteerd

4.4.2. Controledoelstellingen

* Naar de testservice gaan via https met PKCS#11-compatibele browser :

==> CD1 : Server geauthenticeerd

==> CD2 : Client-certificaat ″authenticatie″ geselecteerd

==> CD3 : PIN-code gevraagd

==> CD4 : Authenticatie geslaagd

==> CD5 : Elektronische handtekening automatisch ingeroepen

==> CD6 : Client-certificaat ″authenticatie″ geselecteerd

==> CD7 : PIN-code gevraagd

==> CD8 : Handtekening geslaagd

* [indien van toepassing] Naar de testservice gaan via https met CryptoAPI-compatibele browser :

==> CD1 : Server geauthenticeerd

==> CD2 : Client-certificaat ″authenticatie″ geselecteerd

==> CD3 : PIN-code gevraagd

==> CD4 : Authenticatie geslaagd

==> CD5 : Elektronische handtekening automatisch ingeroepen

==> CD6 : Client-certificaat ″authenticatie″ geselecteerd

==> CD7 : PIN-code gevraagd

==> CD8 : Handtekening geslaagd

4.4.3. Postcondities

* Data met succes op de valideringssite gelogd (OS/browser/interface/kaart/enz.)

* Berichten die op het display verschijnen moeten worden gerapporteerd.

* Kaartnummer, gegevens en bij benadering de duur van de transactie naar de valideringssite moeten worden gerapporteerd

* Screenshot van de laatste pagina van de eID-testsite

Art. 2. Specificaties van de kaartlezers

1. Compatibiliteit van de hardware met de chip

De lezer moet voldoen aan de volgende normen of aanbevelingen :

* ISO/IEC 7816-1 : fysieke kenmerken

* ISO/IEC 7816-2 : afmeting en plaats van de contacten

* ISO/IEC 7816-3 : elektronisch signaal en communicatieprotocols

* ID1 kaartformaat

* Asynchroon protocol T=0 (T=1 is optioneel)

* Elektrische spanning VCC=5V,5% - maximaal 50 mA

* Programmeerspanning VPP gelijk aan VCC, 5%

* Initiële kloksnelheid van 3.5712 MHz (9.600 bps). De kloksnelheid in werking is gelijk aan of groter dan de initiële kloksnelheid.

* Contacten :

? contactdruk (1 N

? contactweerstand (0.3 Ohm

? kracht insteken kaart (12 N

? kracht uitnemen kaart (1 N

2. Compatibiliteit van de software met de chip

2.1. Alle lezers

* Alle lezers moeten voldoen aan de ISO/IEC 7816-4- en 7816-8-standaarden voor het formaat van de uitwisselingscommando's.

2.2. Lezers met pinpad

Lezers met een pinpad moeten alle ISO 7816-4- en 7816-8 -commando's (APDU) implementeren die een PIN-input vereisen zonder de PIN door te geven buiten de lezer.

2.2.1. Te implementeren kaartcommando's (APDU)

□ ***PIN controleren***

Veld	Waarde
CLA	'00'
INS	'20'
P1	'00'
P2	PIN-referentie
Lc	Lengte verificatiegegevens
Data	Verificatiegegevens
Le	Leeg

□ ***PIN wijzigen (referentiegegevens wijzigen)***

Veld	Waarde
CLA	'00'
INS	'24'
P1	'00' (gebruiker)
P2	PIN-referentie
Lc	Lengte van het volgende gegevensveld
Data	Bestaande PIN || Nieuwe PIN (aaneenschakeling)
Le	Leeg

2.2.2. Te implementeren lezercommando's (APDU)

eze sectie beschrijft het APDU-commando dat de lezer moet implementeren om met de commando's die verband houden met de PIN van de kaart, in interactie te gaan. Als men naar de lezer gaat via de eID-standaardmiddleware, kan er een proprietary interface worden gebruikt, op voorwaarde dat de producent de DLL verstrekt die de juiste interface implementeert, zoals wordt beschreven in section 3.2.

Als men echter via andere middelen (vb. op maat ontwikkelde applicaties) naar de lezer gaat, dan moeten de voorgestelde commando's ter beschikking worden gesteld van de applicatieontwikkelaars.

Hoewel deze laatste oplossing technisch zal voldoen, is ze erg beperkt en kan ze door bepaalde instellingen en officiële goedkeuringsinstanties worden geweigerd.

De twee commando's die in sectie 2.1.1 worden beschreven, moeten automatisch worden opgeroepen met een PIN (of 2 PINs) die op het pinpad wordt gevraagd wanneer de parameter Lc van de bovengenoemde commando's gelijk is aan 0. In dit geval moet de firmware van de lezer eerst de PIN van het pinpad lezen en vervolgens de parameters Lc en Data vervangen door de juiste informatie die afkomstig is van de PIN die werd ingegeven.

□ ***Lezer PIN controleren***

Veld	Waarde
CLA	'00'
INS	'20'
P1	'00'
P2	PIN-referentie
Lc	'00'
Data	Leeg
Le	Leeg

□ ***Lezer PIN wijzigen***

Veld	Waarde
CLA	'00'
INS	'24'
P1	'00' (gebruiker)
P2	PIN referentie
Lc	'00'
Data	Leeg
Le	Leeg

Om de PIN te wijzigen, moet de nieuwe PIN twee keer worden ingegeven (met vergelijking) om vergissingen te vermijden.

De return-code (status byte) moet de return-code van het kaartcommando zijn die is gedefinieerd in de ISO 7816-4- en 7816-8-standaarden, of een van de volgende :

Status byte	Betekenis
'ECD2'	De tijd voor het ingeven van de PIN is verstreken
'ECD6'	Geannuleerd door de gebruiker
'ECB6'	Geen specifieke diagnose

2.2.3. PIN-formaat

De PIN bestaat uit strings van 8-bytes met het volgende formaat (per nibble), zoals gedefinieerd in de sectie Global PIN in het document "Global Platform - Card Specification - version 2.0.1' - April 7, 2000" :

C	L	P	P	P	P	P/'F'	P/'F'	P/'F'	P/'F'	P/'F'	P/'F'	P/'F'	P/'F'	'F'	'F'

Nibble	Betekenis
C	Controleparameter, bevat altijd '2'
L	Lengte van de PIN (in nibbles) - van '4' tot 'C'
P	PIN-cijfers (minimaal 4)
P/'F'	De overige PIN-cijfers, afhankelijk van de lengte, 'F' else
'F'	Opvulling bevat altijd 'F'

3. Compatibiliteit van de software met de middleware

Deze sectie is bedoeld voor lezers die verbinding maken met een computer die de kaart zal gebruiken via de Microsoft CryptoAPI- of via de PKCS#11-interface.

3.1. PC/SC drivers

Alle lezers moeten over een driver beschikken die compatibel is met PC/SC-versie 1.1 voor het doel-besturingssysteem.

3.2. Integratie met de middleware

Voor de integratie met de middleware dient de producent van de lezer voor elk doel-besturingssysteem een Dynamic Linked Library (of gelijkwaardig) te ontwikkelen die de volgende functies implementeert :

* SCR_Init()

* SCR_VerifyPIN()

* SCR_ChangePIN()

Momenteel bevindt alleen de applicatie Belgian Identity (hieronder aangegeven met ID) zich op de kaart; met het oog op eventuele andere applicaties (die als aparte Data Files of Directories in de kaart worden geïntegreerd) in de toekomst, krijgen sommige functies een parameter, ApplicationID genoemd, die een string bevat die de applicatie die met een van zijn PIN's in interactie wil gaan, identificeert. Het is de bedoeling dat de lezer de applicatie toont die een PIN vraagt (zie 3.3).

3.2.1. SCR_Init( )

Deze functie wordt opgeroepen onmiddellijk na het laden van de DLL. Ze wordt gebruikt om de DLL te initialiseren en te controleren of hij de aangesloten lezer ondersteunt.

Parameters	in/uit	Type	Beschrijving
reader	in	SCARDHANDLE	Handle voor de lezer
language	in	ushort	Taal voor het display : SCR_LG_FRENCH SCR_LG_DUTCH SCR_LG_GERMAN SCR_LG_ENGLISH
supported	out	bool	Boolean : Waar' als de lezer door de DLL wordt ondersteund Vals' als de lezer niet door de DLL wordt ondersteund Dit maakt het mogelijk om alle geregistreerde DLL's dynamisch te testen om de DLL te vinden die met de lezer overeenstemt – «plug and play»- mechanisme.

3.2.2. SCR_VerifyPIN( )

Deze functie controleert de PIN die de gebruiker heeft ingegeven op het pinpad van de lezer.

Parameters	in/uit	Type	Beschrijving
PINID	in	ushort	PIN-identificator op de kaart – afkomstig van de middleware. Als parameter P2 te specificeren in de functie Reader PIN Verify
Usage	in	char	Reden om de PIN in te geven : A' : Authenticatie S' : Handtekening (niet-verwerping) E' : Encryptie P' : Wijziging voorkeur M' : Onderhoud (administratie)
ApplicationID	in	char*	String (ASCII 7 bits) die de applicatie identificeert die de PIN bezit (max. 3 karakters). Moet worden getoond op het display van de lezer.

3.2.3. SCR_ChangePIN( )

Deze functie vervangt de PIN die de gebruiker heeft ingegeven op het pinpad van de lezer door een nieuwe PIN, die eveneens werd ingegeven op het pinpad van de lezer.

Parameters	in/uit	Type	Beschrijving
PINID	in	ushort	PIN-identificator op de kaart.
			Als parameter P2 te specificeren in de functie Reader PIN Change
ApplicationID	in	char*	String (ASCII 7 bits) die de applicatie identificeert die de PIN bezit (max. 3 karakters).
			Moet worden getoond op het display van de lezer.

3.3. Display

De parameters Usage en ApplicationID vormen belangrijke informatie voor de burger.

Usage toont waarom aan de burger wordt gevraagd zijn PIN in te geven (om zich te identificeren, om een document te ondertekenen, enz.). Dit is de betekenis van de verschillende types :

□ 'A' : Authenticatie (logon)

□ 'S' : Handtekening (niet-verwerping)

□ 'E' : Encryptie

□ 'P' : Voorkeur bestandswijziging

□ 'M' : Onderhoud (administratieve actie)

ApplicationID is een string die de applicatie identificeert. Momenteel is alleen de applicatie Belgian Identity (ID) op de kaart aanwezig, maar later kunnen er nog applicaties volgen (vb : geneesheren (doctors) : Doc, advocaten (lawyers) : LAW, enz.). Daarom moet deze informatie eveneens worden gegeven.

Dit is de informatie die de lezer op het display moet tonen als de bovenvermelde functies worden opgeroepen :

3.3.1. SCR_VerifyPIN( )

Als de ruimte op het display beperkt is, moet minimaal de volgende informatie worden getoond :

ApplicationID-Usage: PIN ? ****

vb: ID-S: PIN ? ****.

Als er voldoende ruimte is op het display, moet de informatie worden vertaald in de taal die is gedefinieerd in de functie SCR_Init, zoals

Application: *Identité*
Accès demandé: *Signature (non-répudiation)*
Entrez votre PIN: **** .

3.3.2. SCR_ChangePIN( )

Als de ruimte op het display beperkt is, moet minimaal de volgende informatie worden getoond :

ApplicationID-PIN: Old PIN ? **** *ApplicationID*-PIN: New PIN ? ****

vb: Doc-PIN: Oude PIN ? **** Doc-PIN: Nieuwe PIN ? ****.

Als er voldoende ruimte is op het display, moet de informatie worden vertaald in de taal die is gedefinieerd in de functie SCR_Init, zoals

PIN Verandering
Applicatie: *Advocaten*
Oude PIN: ****
Nieuwe PIN: ****
Nieuwe PIN: **** (Controle)

Art. 3. Specificaties Identity Middleware

1. Inleiding

1.1. Matrix ontwikkelingsomgeving

De eID Toolkit biedt verschillende interfaces voor dezelfde functionaliteiten en bijgevolg dient men de meest geschikte interface te kiezen.

Deze matrix geeft de aanbevolen interface voor verschillende gangbare ontwikkelingsomgevingen en talen.

API / Ontwikkelingsomgeving	C	ActiveX	Java	Java applet
Java			√	
C	√			
VB, Delphi		√		
.NET		√		
VBA, Vbscript, enz.		√		
Perl	√	√		
Web application		6		√

Belangrijke opmerking :

De ActiveX mag niet worden gebruikt om naar de eID-kaart te gaan vanuit een internetpagina omdat

1. hij uitsluitend met Microsoft Internet Explorer werkt;

2. hij door de browser niet wordt vertrouwd en door de meeste programma's en gebruikers zal worden geweigerd;

3. de Java applet een gebruikersinterface bevat om de gebruiker interactief te vragen de toegang te bevestigen.

De huidige ActiveX is bovendien niet programmeerbaar omdat de scripts niet in staat zijn naar het geheugen te gaan dat door de component werd toegewezen; er moet een wrapper worden gemaakt die gebruik maakt van een door het script verstrekte buffer om hem te kunnen programmeren.

2. Functionele specificaties

2.1. Versies en compatibiliteit

De Toolkit verwerkt automatisch alle verschillende versies van de kaarten. Bij het gebruik van de Toolkit hoeft men zich geen zorgen te maken over de manier waarop de gegevens in de kaart zijn opgeslagen omdat ze op een eenvormige manier ter beschikking worden gesteld via de API.

Er bestaat een low-level functie om de verschillende versies van de kaartcomponenten op te roepen, maar ze is enkel bestemd voor technische ontwikkelaars die naar zeer specifieke eigenschappen van de kaart moeten gaan - voor een normale applicatie heeft de versie van de kaart geen belang.

2.2. Ingeven PIN

Verschillende functies aanvaarden een inputparameter "PIN-referentie". Als er een PIN- referentie wordt gegeven en het bericht "toegang geweigerd" verschijnt als de functie naar een bron van de kaart wil gaan, dan zal de functie automatisch de PIN aan de gebruiker vragen en opnieuw trachten naar de bron te gaan (na succesvolle controle van de PIN).

Dit is een "just-in-time" controle van de PIN, want de PIN wordt alleen gevraagd als dat nodig is. Misschien werd er bijvoorbeeld al een permanente PIN ingegeven die nog steeds geldig is. In dat geval zal de PIN niet opnieuw worden gevraagd.

Als de lezer een beveiligde invoer van de PIN toelaat, wordt het pinpad van de lezer gebruikt; in andere gevallen wordt het toetsenbord van de pc gebruikt.

2.3. Opmerking over de maximumlengte van de parameters

De maximumlengte van de gegevens die de functies tonen, hangt af van het type parameter :

* Bytes

* ASCII-karakters

* UTF-8-karakters

Voor C-ontwikkelaars eindigen alle ASCII- en UTF-8-strings op nul. Opgelet : in de opgegeven lengte is de '[0]' op het einde van op nul eindigende strings niet inbegrepen.

2.4. Multi-threaded applicatie

De library is niet thread-safe. Het is de taak van de calling-applicatie om de library niet gelijktijdig in parallelle threads te gebruiken.

Opmerking : de CSP is thread-safe, maar het is niet mogelijk om de CSP in één thread en de Toolkit in een andere thread op te roepen.

2.5. API-organisatie

De functies zijn ondergebracht in 4 categorieën :

* Functies voor initialisatie en beëindiging, noodzakelijk voor de initialisatie en de beëindiging van de Toolkit.

* Identiteitsfuncties, die worden gebruikt om de identiteitsgegevens (naam, adres, enz.) op de kaart op te vragen.

* Algemene high-level functies, die worden gebruikt om op een algemene manier naar de informatie te gaan (bestanden, PIN), hoofdzakelijk in andere applicaties dan de identiteitsapplicatie. Deze functies hoeven niet naar de identiteitsgegevens te gaan.

* Low-level functies, bestemd voor ontwikkelaars die zeer technische functies nodig hebben, of voor debugging. De gewone ontwikkelaars hebben deze functies niet nodig.

2.6. Functies voor initialisatie en beëindiging

2.6.1. BEID_Init

Deze functie initialiseert de Toolkit.

Deze functie moet vóór elke andere functie worden opgeroepen.

Het principe voor de validering van het certificaat (hetzij door OCSP of CRL te gebruiken) wordt gegeven. Het geldt voor alle functie-calls totdat BEID_Exit( ) wordt opgeroepen. De OCSP en CRL Mandatory flags sluiten elkaar uit. Als zowel OCSP als CRL worden gebruikt, zal eerst OCSP worden geprobeerd; als dat lukt, wordt het proces stopgezet, anders wordt CRL gebruikt.

Parameter	In	Out	Detail	Toegelaten waarden of formaat	Max. lengte
Reader Name	X		Naam lezer	Lege string of NULL voor automatische detectie van de eerste lezer	
				PC/SC naam lezer	
				«VIRTUAL» voor een virtuele lezer (Zie 2.7.4)	
OCSP	X		OCSP Policy	0=Niet gebruikt (standaard),1=Optioneel, 2=Verplicht (zie 2.11)	
CRL	X		CRL Policy	0=Niet gebruikt (standaard),1=Optioneel, 2=Verplicht (zie 2.11)	
PC/SC handle		X	PC/SC handle	De meeste ontwikkelaars hoeven geen rekening te houden met deze output-parameter; hij wordt alleen gegeven voor de ontwikkelaars die met de lezer willen werken op PC/SC-niveau voor zeer specifieke technische doeleinden.	

2.6.2. BEID_Exit

Deze functie verwijdert alle gegevens die door de Toolkit werden gebruikt.

Deze functie moet worden opgeroepen aan het einde van het programma.

Parameter	In	Out	Detail	Toegelaten waarden of formaat	Max. lengte

2.7. Identiteitsfuncties

Alle identiteitsfuncties zijn onafhankelijk. Dat betekent dat er geen andere functie moet worden opgeroepen samen met een identiteitsfunctie (behalve de functies voor initialisatie en beëindiging).

Al deze functies kunnen worden opgeroepen ongeacht de huidige status van de kaart - ongeacht of er een andere DF (Data File) dan de identiteitsfunctie wordt geselecteerd, enz.

2.7.1. BEID_GetID

Parameter	In	Out	Detail	Toegelaten waarden of formaat	Max. lengte
Version		X	IDdataversie	SHORT	
CardNumber		X	Logisch kaartnummer	ASCII	12
ChipNumber		X	Fysiek chipnummer	ASCII	16
ValidityDateBegin		X	Begindatum geldigheid kaart	ASCII : YYYYMMDD	8
ValidityDateEnd		X	Einddatum geldigheid kaart	ASCII : YYYYMMDD	8
Municipality		X	Gemeente die de kaart heeft afgeleverd	UTF-8	50
NationalNumber		X	Nationaal nummer	ASCII	11
Name		X	Familienaam	UTF-8	80
FirstName1		X	1e voornaam (1)	UTF-8	60
FirstName2		X	2e voornaam	UTF-8	30
FirstName3		X	3e voornaam (eerste letter)	UTF-8	1
Nationality		X	Nationaliteit – ISO-code	ASCII	3
BirthLocation		X	Geboorteplaats	UTF-8	50
BirthDate		X	Geboortedatum	ASCII : YYYYMMDD	8
Sex		X	Geslacht	ASCII : M/F	1
NobleCondition		X	Adellijke titel	UTF-8	40
DocumentType		X	Type document	LONG 1. Belgische burger 2. EU-burger 3. niet EU-burger 7. Bootstrap card 8. Habilitation/machtigings» card	
WhiteCane		X	Witte stok toegelaten (blinden)	Boolean	
YellowCane		X	Gele stok toegelaten (slechtzienden)	Boolean	
ExtendedMinority		X	Uitgebreide minderheid	Boolean	
HashPhoto		X	Hash van de foto. Dit gegeven is voor de meeste applicaties niet relevant; het wordt enkel gegeven voor technische applicaties.	Binary (SHA-1)	20
CertifCheck		X	Controle certificaat en valideringsresultaat	zie 2.13	

2.7.2. BEID_GetAddress

Parameter	In	Out	Detail	Toegelaten waarden of formaat	Max. lengte
Version		X	Versie adresgegevens	SHORT	
street		X	Straat (2)	UTF-8	80
Street number		X	Huisnummer	ASCII	8
Box number		X	Busnummer	ASCII	4
zip		X	Postcode	ASCII	6
municipality		X	Naam gemeente	UTF-8	50
country		X	ISO-code land	ASCII	3
CertifCheck		X	Controle certificaat en validatieresultaat	zie 2.13	

2.7.3.BEID_GetPicture

Parameter	In	Out	Detail	Toegelaten waarden of formaat	Max. lengte
Picture		X	Foto, in JPEG-formaat	BYTE stream	10 000
PictureLen	X	X	Lengte Byte Stream foto	Long : omvang in bytes IN : omvang opgegeven buffer OUT : reële omvang opgegeven data	
CertifCheck		X	Controle certificaat en validatieresultaat	zie 2.13	

2.7.4. Offline identiteitsfuncties

Soms kan het nodig zijn de gegevens van de kaart te archiveren om ze niet alleen nu te valideren, maar ze ook met hun handtekening te bewaren als bewijs. In dat geval hebt u twee functies nodig : een functie om de raw data van de kaart te lezen, en een functie om de raw data te geven die u hebt opgeslagen als input voor de identiteitsfuncties.

2.7.4.1. BEID_GetRawData

Deze functie geeft de raw data files van de kaart. (ID, Address, Picture, RRN certificate, CardData, TokenInfo, Challenge/Response from internal authenticate). U moet deze data opslaan voor later gebruik.

Opmerking : de raw data worden niet gecontroleerd, enkel gelezen. U moet de gewone identiteitsfuncties oproepen om ze te valideren.

Parameter	In	Out	Detail	Toegelaten waarden of formaat	Max. lengte
RawData		X	Raw Data		

2.7.4.2. BEID_SetRawData

Deze functie gebruikt de raw data als input voor de volgende identiteitsfuncties. Dit maakt het mogelijk om bepaalde identiteitsgegevens die eerder al werden opgeslagen, te controleren. De kaartlezer moet niet aanwezig zijn om deze functie te gebruiken.

Om de raw data te controleren moet u :

* De functie BEID_Init( ) oproepen met de parameter reader op ”VIRTUAL”

* De functie BEID_SetRawData( ) oproepen

De gewone identiteitsfuncties oproepen

Parameter	In	Out	Detail	Toegelaten waarden of formaat	Max. lengte
RawData	X		Raw Data		

2.8. Algemene high-level functies

Deze functies geven toegang - geïntegreerd in de Toolkit - tot algemene functies voor applicaties die andere acties moeten uitvoeren dan het louter naar de identiteitsgegevens gaan.

2.8.1. BEID_BeginTransaction

Deze functie begint de transactie en wacht tot alle andere transacties zijn beëindigd voor ze begint. Geen enkele andere applicatie zal toegang hebben tot de kaart totdat BEID_EndTransaction wordt opgeroepen.

De functie moet worden opgroepen vóór andere functies voor het gebruik van de kaart die moeten worden gegroepeerd. Deze functie is niet nodig om individuele functies of identiteitsfuncties op te roepen.

Meestal wordt er een transactie gebruikt om een applicatie te selecteren vooraleer naar een bestand of PIN te gaan.

Parameter	In	Out	Detail	Toegelaten waarden of formaat	Max. lengte
RawData	X		Raw Data		

2.8.2. BEID_EndTransaction

Deze functie beëindigt een eerder opgegeven transactie en laat de andere applicaties toe de interacties met de kaart te beëindigen.

Deze functie moet op het einde van bepaalde operationele kaartfuncties worden opgeroepen.

Parameter	In	Out	Detail	Toegelaten waarden of formaat	Max. lengte
RawData	X		Raw Data		

2.8.3. BEID_SelectApplication

Deze functie kiest een applicatie op de kaart.

Parameter	In	Out	Detail	Toegelaten waarden of formaat	Max. lengte
AID	X		AID applicatie	Byte Stream – afhankelijk van de kaart (Vb. A000000177504B43532D3135)	
AIDLen	X		AID applicatie Lengte	Long : lengte (in bytes) van opgegeven AID	

2.8.4. BEID_ReadFile

Deze functie leest een bestand op de kaart. Als er een PIN-referentie wordt gegeven, zal de PIN worden gevraagd en zo nodig gecontroleerd (just-in-time checking).

Parameter	In	Out	Detail	Toegelaten waarden of formaat	Max. lengte
FileID	X		Pad naar te lezen bestand – afhankelijk van de huidige applicatie	Byte Stream – afhankelijk van de kaart (Vb. DF 00 50 32)	
FileIDLen	X		Lengte FileID	Long : lengte (in bytes) van FileID	
OutData		X	Opgegeven data		
OutDataLen	X	X	Opgegeven data lengte	Long : omvang in bytes IN : omvang OutData buffer OUT : omvang opgegeven data	64 Kb
PIN	X		PIN die het bestand beschermt	zie 2.10	

2.8.5. BEID_WriteFile

Deze functie schrijft een bestand op de kaart. Als er een PIN-referentie wordt gegeven, zal de PIN worden gevraagd en zo nodig gecontroleerd (just-in-time checking).

Parameter	In	Out	Detail	Toegelaten waarden of formaat	Max. lengte
FileID	X		Pad naar te lezen bestand – afhankelijk van de huidige applicatie	Byte Stream – afhankelijk van de kaart (Vb. DF 00 50 32)	
FileIDLen	X		Lengte FileID	Long : lengte (in bytes) van FileID	
InData	X		Opgegeven data		
InDataLen	X		Opgegeven data lengte		
PIN	X		PIN die het bestand beschermt	zie 2.10	

2.8.6. BEID_VerifyPIN

Deze functie controleert een PIN.

Parameter	In	Out	Detail	Toegelaten waarden of formaat	Max. lengte
PIN	X		PIN die het bestand beschermt	zie 2.10	
Pin	X		Opgegeven Pin	ASCII Indien leeg of NULL, aan de gebruiker vragen	12
NrTriesLeft		X	Aantal resterende pogingen	Long : aantal resterende pogingen	

2.8.7. BEID_ChangePIN

Deze functie wijzigt een PIN.

Parameter	In	Out	Detail	Toegelaten waarden of formaat	Max. lengte
PIN	X		PIN die het bestand beschermt	zie 2.10	
OldPin	X		Oude Pin	ASCII Indien leeg of NULL, aan de gebruiker vragen	12
NewPin	X		Nieuwe Pin	ASCII Indien leeg of NULL, aan de gebruiker vragen	12
NrTriesLeft		X	Aantal resterende pogingen	Long : aantal resterende pogingen	

2.8.8. BEID_GetPINStatus

Deze functie haalt de pinstatus op. De resulterende informatie kan door de basissleutel van de kaart worden ondertekend (key '0x81' in de huidige DF-applicatie).

Parameter	In	Out	Detail	Toegelaten waarden of formaat	Max. lengte
PIN	X		PIN die het bestand beschermt	zie 2.10	
NrTriesLeft		X	Aantal resterende pogingen	Long : aantal resterende pogingen	
signature	X		Handtekening vereist	Boolean	
signedStatus		X	Ondertekende status	Byte Stream	256
signedStatusLen	X	X	Lengte ondertekende status	Long : omvang in bytes IN : omvang Byte Stream OUT : eigenlijke omvang opgegeven data : 256	

2.9. Low-level functies

Opmerking : de low-level functies zijn bestemd voor applicaties die naar specifieke technische functies willen gaan die niet beschikbaar zijn via de gewone functies, of voor debugging. De gewone applicaties hebben deze low-level functies niet nodig.

2.9.1. BEID_GetVersionInfo

Deze functie geeft de versie van de verschillende componenten (applet, OS,...). De resulterende informatie kan worden ondertekend door de basissleutel van de kaart (key '0x81' in de huidige DF-applicatie).

Parameter	In	Out	Detail	Toegelaten waarden of formaat	Max. lengte
VersionInfo		X	Zie specificaties applet en inhoud kaart		
signature	X		Handtekening vereist	Boolean	
signedStatus		X	Ondertekende status	BYTE stream	128
signedStatusLen	X	X	Lengte ondertekende status	Long : omvang in bytes IN : omvang buffer : min. 128 OUT : eigenlijke omvang opgegeven data (128)	

2.9.2. BEID_SendAPDU

Deze functie stuurt een APDU-commando naar de kaart.

Opgelet : wanneer de APDU naar de kaart wordt gestuurd, kan de Toolkit de compatibiliteit tusen de applet-versies niet garanderen (behalve voor het communicatiegedeelte). Het is mogelijk dat de call niet werkt in een volgende versie van de applet.

Als er een PIN-referentie wordt gegeven, zal de PIN worden gevraagd en zo nodig gecontroleerd (just-in-time checking).

Opmerking : het commando zal eerst worden geprobeerd zonder de PIN te vragen; als het commando niet slaagt met status "toegang geweigerd", zal de PIN worden gevraagd en wordt het commando opnieuw uitgevoerd.

Parameter	In	Out	Detail	Toegelaten waarden of formaat	Max. lengte
CmdAPDU	X		Het commando APDU wordt naar de kaart gestuurd	Byte Stream – afhankelijk van de kaart	256
CmdAPDULen	X		Lengte input APDU-commando		
PIN	X		PIN die het bestand beschermt	zie 2.10	
RespAPDU		X	APDU-respons ontvangen van de kaart		256
RespAPDULen	X	X	Lengte respons APDU-commando	Long : omvang in bytes IN : omvang buffer OUT : eigenlijke omvang opgegeven data	

2.9.3. BEID_FlushCache

Deze functie flusht de gegevens die zich in het geheugen en op de schijf bevinden.

Parameter	In	Out	Detail	Toegelaten waarden of formaat	Max. lengte

Als er een bepaald PIN-id aan de Toolkit wordt gegeven, kunt u ofwel de PKCS#15 PIN id, ofwel de PIN-referentie van het besturingsssysteem opgeven.

Als de PIN niet in de Toolkit is geregistreerd, dan moet u een string ingeven die beschrijft waarom de PIN vereist is (vb. : "Wijziging persoonlijke gegevens"). U moet ook een short string opgeven (max. 3 karakters) die op het display van de kaartlezer moet worden getoond als die aanwezig is (vb : "MOD").

Een PIN bestaat dus uit 4 velden :

▷ Het type PIN : PKCS#15 of Operating System (zie 2.10.1).

▷ De PIN OS referentie of : PKCS#15 id.

▷ De gebruikscode (zie 2.10.2).

▷ De lange beschrijving (in de taal van de gebruiker).

▷ De korte beschrijving (in de taal van de gebruiker).

Om de beveiliging te verbeteren en de informatie op het display te standaardiseren, kan de beschrijving die door de applicatie wordt gegeven worden overschreven door de Toolkit als de Toolkit het juiste gebruik kan bepalen.

Als er geen PIN moet worden gecontroleerd, moet er een NULL-waarde worden gebruikt.

Opmerking : als er een PIN beschikbaar is in de PKCS#15-structuur, dan is het veiliger de PKCS#15-referentie te gebruiken dan de OS-referentie omdat het OS in de toekomst kan veranderen.

2.10.1. Types PIN

Waarde	C constant	Verklaring
0	BEID_PIN_TYPE_PKCS15	PKCS15 PIN
1	BEID_PIN_TYPE_OS	OS PIN

2.10.2. Gebruik PIN

Waarde	C constant	Verklaring
0		Applicatie gedefinieerd
1	BEID_USAGE_AUTH	Gebruik authenticatie
2	BEID_USAGE_SIGN	Gebruik handtekening
…	…	Toekomstig gebruik

2.11. OCSP en CRL input policy parameters

Waarde	C constant	Verklaring
0	BEID_OCSP_CRL_NOT_USED	Geen controle CRL/OCSP
1	BEID_OCSP_CRL_OPTIONAL	Optionele controle CRL/OCSP
2	BEID_OCSP_CRL_MANDATORY	Verplichte controle CRL/OCSP

2.12. Status van de functies

Elke functie geeft een algemene return-code die het type fout opgeeft. In de meeste gevallen zal alleen deze return-code worden gebruikt. Om meer gedetailleerde informatie te verkrijgen over de technische oorzaken, kunnen ook de volgende status-codes worden gegeven :

* De code systeemfout (lang).

* De foutcode PC/SC (lang).

* De smart card Status Word (dubbele byte).

2.12.1. Algemene return-codes

Waarde	C constant	Verklaring
0	BEID_OK	Functie geslaagd
1	BEID_E_SYSTEM	Onbekende systeemfout (zie foutcode systeem)
2	BEID_E_PCSC	Onbekende PC/SC-fout (zie PC/SC-foutcode)
3	BEID_E_CARD	Onbekende kaartfout (zie kaartstatus word)
4	BEID_E_BAD_PARAM	Ongeldige parameter (NULL pointer, out of bound, enz.)
5	BEID_E_INTERNAL	Interne consistentiecontrole mislukt
6	BEID_E_INVALID_HANDLE	Ongeldige handle
7	BEID_E_INSUFFICIENT_BUFFER	De databuffer is te klein voor de opgegeven data
8	BEID_E_COMM_ERROR	Er werd een interne communicatiefout vastgesteld
9	BEID_E_TIMEOUT	De specifieke timeout-waarde is verlopen
10	BEID_E_UNKNOWN_CARD	De smart card wordt niet herkend
11	BEID_E_KEYPAD_CANCELLED	Input op pinpad geannuleerd
12	BEID_E_KEYPAD_TIMEOUT	Time-out van pinpad
13	BEID_E_KEYPAD_PIN_MISMATCH	De twee PIN's stemmen niet overeen
14	BEID_E_KEYPAD_MSG_TOO_LONG	Bericht op pinpad te lang
15	BEID_E_INVALID_PIN_LENGTH	Ongeldige lengte PIN
16	BEID_E_VERIFICATION	Fout bij controle handtekening of validering certificaat (zie 2.13)
17	BEID_E_NOT_INITIALIZED	Toolkit niet geïnitialiseerd
18	BEID_E_UNKNOWN	Er werd een interne fout vastgesteld maar de bron is niet bekend
19	BEID_E_UNSUPPORTED_FUNCTION	Functie niet ondersteund
20	BEID_E_INCORRECT_VERSION	De Toolkit-versie is niet compatibel met de calling interface.
		Het programma moet opnieuw worden gecompileerd.
21	BEID_E_INVALID_ROOT_CERT	

2.13. Controle handtekening en validering certificaten

Elke functie die ondertekende gegevens geeft, controleert altijd de handtekening en de integrtiteit van de volledige certificaatketen.

De functie geeft

* de status van de handtekeningcontrole (long);

* de globale status van de certificaatvalidering (long);

* voor elk certificaat

— het certificaat,

— het label van het certificaat,

— de individuele controlestatus,

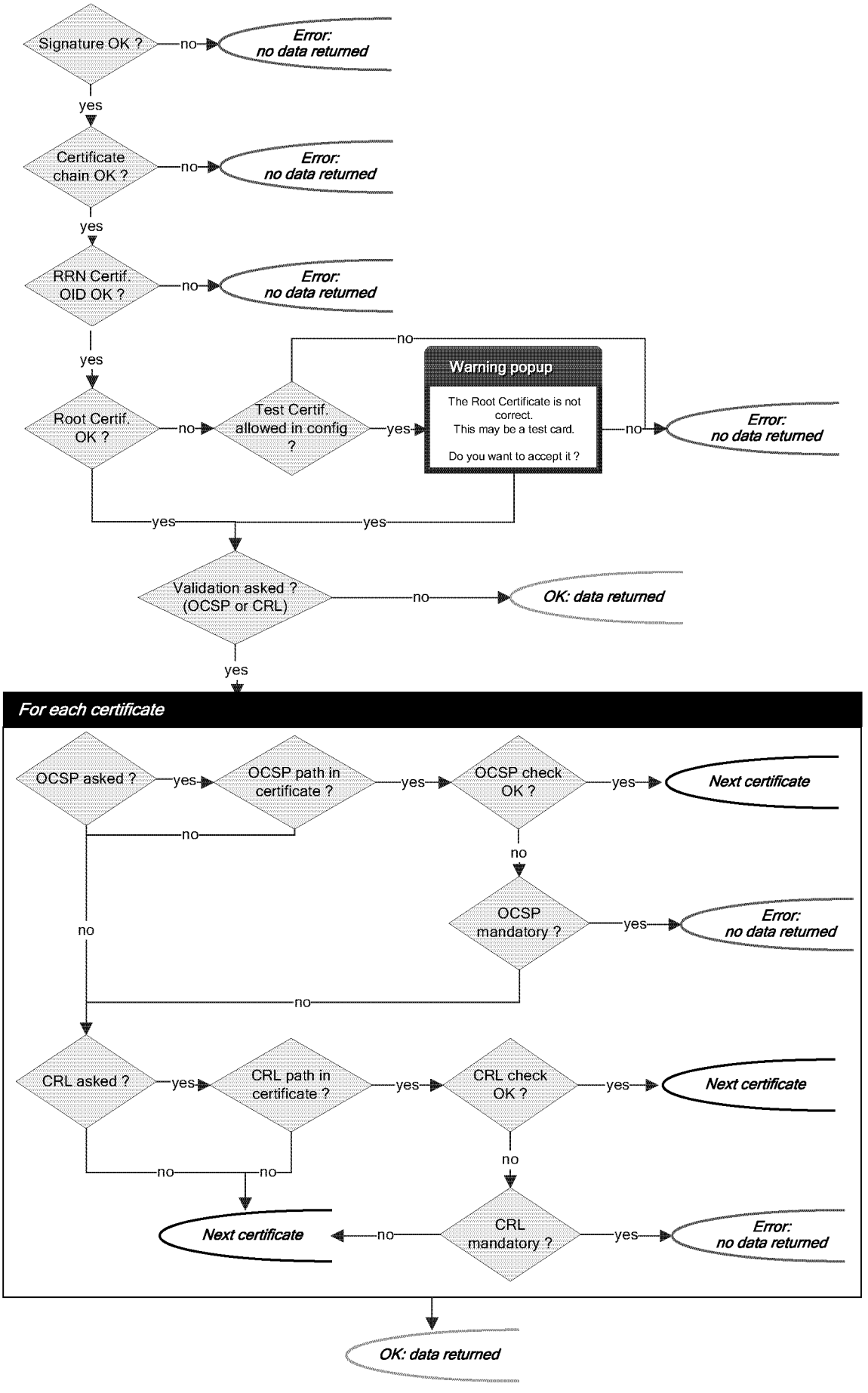
— de individuele valideringsstatus,

— de individuele policy : OCSP of CRL.

Als er een fout optreedt in de handtekeningen of de valideringsketen, eindigen alle functies met een foutcode en **worden er geen data gegeven.**

Als het root-certificaat geen officieel root-certificaat is, dan verschijnt er een venster met het bericht dat het Root Certificate niet het certificaat is dat werd verwacht. De gebruiker kan dit Root Certificate tijdelijk aanvaarden - dat is nodig om de testkaarten met een ander dan het officiële Root Certificate te gebruiken.

Het onderstaande diagram beschrijft de werkstroom van de controle van de handtekening :



2.13.1. Controle handtekening

Waarde	C constant	Verklaring
-1	BEID_SIGNATURE_ PROCESSING_ERROR	Fout bij de controle van de handtekening.
0	BEID_SIGNATURE_VALID	De handtekening is geldig.
1	BEID_SIGNATURE_INVALID	De handtekening is niet geldig.
2	BEID_SIGNATURE_VALID_WRONG_RRNCERT	De handtekening is geldig maar het RRN-certificaat is fout.
3	BEID_SIGNATURE_INVALID_WRONG_RRNCERT	De handtekening is niet geldig en het RRN-certificaat is fout.

2.13.2. Controle certificaat en validatieresultaten

Waarde	C constant	Verklaring
0	BEID_CERTSTATUS_CERT_VALIDATED_OK	Validering met succes uitgevoerd.
1	BEID_CERTSTATUS_CERT_NOT_VALIDATED	Geen validering uitgevoerd.
2	BEID_CERTSTATUS_UNABLE_TO_GET_ISSUER_CERT	Kan geen gebruikerscertificaat ophalen
3	BEID_CERTSTATUS_UNABLE_TO_GET_CRL	Kan geen CRL-certificaat ophalen
4	BEID_CERTSTATUS_UNABLE_TO_DECRYPT-_CERT_SIGNATURE	Kan handtekening certificaat niet decrypteren
5	BEID_CERTSTATUS_UNABLE_TO_DECRYPT-_CRL_SIGNATURE	Kan handtekening CRL niet decrypteren
6	BEID_CERTSTATUS_UNABLE_TO_DE-CODE_ISSUER_PUBLIC_KEY	Kan uitgever publieke sleutel niet decoderen
7	BEID_CERTSTATUS_CERT_SIGNATURE_FAILURE	Fout certificaat handtekening
8	BEID_CERTSTATUS_CRL_SIGNATURE_FAILURE	Fout CRL-handtekening
9	BEID_CERTSTATUS_CERT_NOT_YET_VALID	Het certificaat is nog niet geldig
10	BEID_CERTSTATUS_CERT_HAS_EXPIRED	Het certificaat is vervallen
11	BEID_CERTSTATUS_CRL_NOT_YET_VALID	CRL is nog niet geldig
12	BEID_CERTSTATUS_CRL_HAS_EXPIRED	CRL is vervallen
13	BEID_CERTSTATUS_ERR_IN_CERT_NOT_BE-FORE_FIELD	Formatteringsfout in veld notBefore van certificaat
14	BEID_CERTSTATUS_ERR_IN_CERT_NO-T_AFTER_FIELD	Formatteringsfout in veld notAfter van certificaat
15	BEID_CERTSTATUS_ERR_IN_CRL_LAST_UP-DATE_FIELD	Formatteringsfout in veld lastUpdate van CRL
16	BEID_CERTSTATUS_ERR_IN_CRL_NEXT_UP-DATE_FIELD	Formatteringsfout in veld next Update van CRL
17	BEID_CERTSTATUS_OUT_OF_MEM	Onvoldoende geheugen
18	BEID_CERTSTATUS_DEPTH_ZERO_SELF_S-IGNED_CERT	Self signed certificaat
19	BEID_CERTSTATUS_SELF_SIGNED_CERT_IN_CHAIN	Self signed certificaat in de certificaatketen

Waarde	C constant	Verklaring
20	BEID_CERTSTATUS_UNABLE_TO_GET_ISSUER_CERT_LOCALLY	Kan certificaat lokale uitgever niet vinden
21	BEID_CERTSTATUS_UNABLE_TO_VERIFY_LEAF_SIGNATURE	Kan het eerste certificaat niet controleren
22	BEID_CERTSTATUS_CERT_CHAIN_TOO_LONG	Certificaatketen te lang
23	BEID_CERTSTATUS_CERT_REVOKED	Certificaat herroepen
24	BEID_CERTSTATUS_INVALID_CA	Ongeldig CA-certificaat
25	BEID_CERTSTATUS_PATH_LENGTH_EXCEEDED	Beperking padlengte overschreden
26	BEID_CERTSTATUS_INVALID_PURPOSE	Ongeldig doelcertificaat
27	BEID_CERTSTATUS_CERT_UNTRUSTED	Certificaat niet vertrouwd
28	BEID_CERTSTATUS_CERT_REJECTED	Certificaat verworpen
29	BEID_CERTSTATUS_SUBJECT_ISSUER_MISMATCH	Geen overeenstemming subject issuer
30	BEID_CERTSTATUS_AKID_SKID_MISMATCH	Geen overeenstemming autoriteit en subject key identifier
31	BEID_CERTSTATUS_AKID_ISSUER_SERIAL_MISMATCH	Geen overeenstemming serienummer autoriteit en uitgever
32	BEID_CERTSTATUS_KEYUSAGE_NO_CERTSIGN	Sleutel bevat geen ondertekening certificaat
33	BEID_CERTSTATUS_UNABLE_TO_GET_CRL_ISSUER	Kan certificaat CRL-uitgever niet vinden
34	BEID_CERTSTATUS_UNHANDLED_CRITICAL_EXTENSION	Kritische extensie niet behandeld

2.13.3. Gebruikte OCSP en CRL policies

Waarde	C constant	Verklaring
0	BEID_POLICY_NONE	Geen policy gebruikt
1	BEID_POLICY_OCSP	OCSP policy gebruikt
2	BEID_POLICY_CRL	CRL policy gebruikt
3	BEID_POLICY_BOTH	OCSP en CRL policy gebruikt

3. Programmeerinterfaces

3.1. C API

Alle outputbuffers moeten worden toegewezen door de calling application.

Alle functies gebruiken de C calling convention, niet de Pascal convention.

Struct packing is ingesteld op 8.

3.1.1. Structures

```
typedef struct {
      long general;              // General return code
      long system;               // System error (errno)
      long pcsc;                 // PC/SC error
      BYTE cardSW[2];            // Card status word
} BEID_Status;

typedef struct {
 BYTE certif[…];                 // Byte stream encoded certificate
 long certifLength;              // Size in bytes of the encoded certificate
 char certifLabel[…];            // Label of the certificate (Authentication,
Signature, CA, Root,…)
 long certifStatus;              // Validation status code (see 2.3.3
                                                    Certificate
                                                    checking and
                                                    validation results)
} BEID_Certif;

typedef struct {
      long usedPolicy;                    // Policy used (see 2.13.3)
      BEID_Certif certificates[…];        // Array of BEID_Certif
structures
      long certificatesLength;            // Number of elements in Array
      long signatureCheck;                // Status of signature (for ID and
                                                    Address) or
                                          // hash (for Picture) on retrieved
                                                    field (see 2.13)

} BEID_Certif_Check;

typedef struct {
      // Get Card Data
      BYTE SerialNumber[16];
      BYTE ComponentCode;
      BYTE OSNumber;
      BYTE OSVersion;
      BYTE SoftmaskNumber;
      BYTE SoftmaskVersion;
      BYTE AppletVersion;
      ushort GlobalOSVersion;
      BYTE AppletInterfaceVersion;
      BYTE PKCS1Support;
      BYTE KeyExchangeVersion;
      BYTE ApplicationLifeCycle;
      // TokenInfo Data
      BYTE GraphPerso;
      BYTE ElecPerso;
      BYTE ElecPersoInterface;
      BYTE Reserved;
```

```
} BEID_VersionInfo;

typedef struct {
     short version;
     char cardNumber[…];
     char chipNumber[…];
     char validityDateBegin[…];
     char validityDateEnd[…];
     TCHAR municipality[…];
     char nationalNumber[…];
     TCHAR name[…];
     TCHAR firstName1[…];
     TCHAR firstName2[…];
     TCHAR firstName3[…];
     char nationality[…];
     TCHAR birthLocation[…];
     char birthDate[…];
     char sex[…];
     TCHAR nobleCondition[…];
     long documentType;
     BOOL whiteCane;
     BOOL yellowCane;
     BOOL extendedMinority;
     BYTE hashPhoto[…]
} BEID_ID_Data;

typedef struct {
     short version;
     TCHAR street[…];
     char streetNumber[…];
     char boxNumber[…];
     char zip[…];
     TCHAR municipality[…];
     char country[…];
} BEID_Address;

typedef struct {
   BYTE *data;
   unsigned long length;
  } BEID_Bytes;

typedef struct {
  long pinType;          // PIN Type (see 2.10.1)
  BYTE id;               // PIN reference or ID
  long usageCode;        // PIN Usage (see 2.10.2)
  char *shortUsage;      // May be NULL for usage known by the
middleware
  char *longUsage;       // May be NULL for usage known by the
middleware
} BEID_Pin;


typedef struct { … } BEID_Raw;
```

3.1.2. Functions

```
BEID_Status                                    // return code
     BEID_Init(
     char *ReaderName,                         // Reader Name
long OCSP,                                     // OCSP policy certificates
checking & validity
     long CRL,                                 // CRL policy certificates
     checking & validity
     long *CardHandle,                         // output PC/SC handle
     );

BEID_Status                                    // return code
     BEID_Exit(
     );

BEID_Status                                    // return code
     BEID_GetID(
     BEID_ID_Data *IDData,                     // output data
BEID_Certif_Check *CertifCheck                 // certificates checking &
validity
     );

BEID_Status                                    // return code
     BEID_GetAddress(
     BEID_ID_Address *Address,                 // output address data
BEID_Certif_Check *CertifCheck                 // certificates checking &
validity
     );

BEID_Status                                    // return code
     BEID_GetPicture(
     BEID_Bytes *Picture,                      // output picture (JPEG
     format)
BEID_Certif_Check *CertifCheck                 // certificates checking &
validity
     );

BEID_Status                                    // return code
     BEID_GetRawData(
     BEID_Raw *Raw                             // output Raw Data
);

BEID_Status                                    // return code
     BEID_SetRawData(
     BEID_Raw *Raw                             // input Raw Data
);

BEID_Status                                    // return code
     BEID_GetVersionInfo(
     BEID_VersionInfo *pVersionInfo,
     BOOL Signature,                           // Signature needed
     BEID_Bytes *SignedStatus,                 // Signature 256 bytes
     );

BEID_Status                                    // return code
BEID_BeginTransaction( );
```

```
BEID_Status                          // return code
BEID_EndTransaction( );

BEID_Status                          // return code
BEID_SelectApplication(
     BEID_Bytes *Application         // Application ID
     );

BEID_Status                          // return code
BEID_FlushCache( );

BEID_Status                          // return code
BEID_SendAPDU(
     BEID_Bytes *CmdAPDU,            // Command APDU
     BEID_Pin *pin,                  // Pin Reference
     BEID_Bytes *RespAPDU            // Response APDU
     );

BEID_Status                          // return code
BEID_VerifyPIN(
     BEID_Pin *pin,                  // Pin Reference
     char *Pin,                      // Pin code
     long *TriesLeft,                // Tries remaining
     );

BEID_Status                          // return code
BEID_ChangePIN(
     BEID_Pin *pin,                  // Pin Reference
     char *OldPin,                   // Old Pin code
     char *NewPin,                   // New Pin code
     long *TriesLeft                 // Tries remaining
     );

BEID_Status                          // return code
BEID_GetPINStatus(
     BEID_Pin *pin,                  // Pin Reference
     long *TriesLeft,                // Tries remaining
     BOOL bSignature,                // Signature needed
     BEID_Bytes *SignedStatus        // Signature
     );

BEID_Status                          // return code
BEID_ReadFile(
     BEID_Bytes *FileID,             // File to read relative path
     BEID_Bytes *OutData,            // Returned data
     BEID_Pin *pin,                  // Pin Reference
);

BEID_Status                          // return code
BEID_WriteFile(
     BEID_Bytes *FileID,             // File to write relative path
     BEID_Bytes *InData,             // Write buffer
     BEID_Pin *pin,                  // Pin Reference
);
```

3.2. Java API

Om compatibel te blijven met de native code die de toegang tot de kaart implementeert, gebruikt de Java-implementatie geen exception handling; maar worden alle fouten weergegeven in de foutcodes, zoals beschreven in 2.12.

Package: be.belgium.eid;

3.2.1. Data Classes

```
public class BEID_Address
{
  public BEID_Address()
  public short getVersion()
  public String getStreet()
  public String getStreetNumber()
  public String getBoxNumber()
  public String getZip()
  public String getMunicipality()
  public String getCountry()
}

public class BEID_Raw { … }

public class BEID_Bytes
{
  public BEID_Bytes()
  public byte[] getData()
}

public class BEID_Certif_Check
{
  public BEID_Certif_Check()
  public int getUsedPolicy()
  public BEID_Certif getCertificate(int Index)
  public int getCertificatesLength()
  public int getSignatureCheck()
 }

public class BEID_Certif
{
  public BEID_Certif()
  public byte[] getCertif()
  public String getCertifLabel()
  public int getCertifStatus()
}

public class BEID_Long
{
   public BEID_Long()
   public long getLong()
 }

public class BEID_Pin
{
  public BEID_Pin()
  public void setPinType(int pinType)
  public void setId(short id)
  public void setUsageCode(int usageCode)
  public void setShortUsage(String shortUsage)
  public void setLongUsage(String longUsage)
   }
```

```
public class BEID_Status
{
  public BEID_Status()
  public int getGeneral()
  public int getSystem()
  public int getPcsc()
  public byte[] getCardSW()
}

public class BEID_VersionInfo
{
  public BEID_VersionInfo()
  public byte[] getSerialNumber()
  public short getComponentCode()
  public short getOSNumber()
  public short getOSVersion()
  public short getSoftmaskNumber()
  public short getSoftmaskVersion()
  public short getAppletVersion()
  public int getGlobalOSVersion()
  public short getAppletInterfaceVersion()
  public short getPKCS1Support()
  public short getKeyExchangeVersion()
  public short getApplicationLifeCycle()
  public short getGraphPerso()
  public short getElecPerso()
  public short getElecPersoInterface()
  public short getReserved()
}

public class BEID_ID_Data
{
  public BEID_ID_Data()
  public short getVersion()
  public String getCardNumber()
  public String getChipNumber()
  public String getValidityDateBegin()
  public String getValidityDateEnd()
  public String getMunicipality()
  public String getNationalNumber()
  public String getName()
  public String getFirstName1()
  public String getFirstName2()
  public String getFirstName3()
  public String getNationality()
  public String getBirthLocation()
  public String getBirthDate()
  public String getSex()
  public String getNobleCondition()
  public int getDocumentType()
  public boolean getWhiteCane()
  public boolean getYellowCane()
  public boolean getExtendedMinority()
  public byte[] getHashPhoto()
  }
```

3.2.2. Main Class

```
public class eidlib
{
  public static BEID_Status BEID_Init(String ReaderName, int
OCSP, int CRL, BEID_Long CardHandle)

  public static BEID_Status BEID_Exit()

  public static BEID_Status BEID_GetID(BEID_ID_Data IDData,
BEID_Certif_Check CertifCheck)

  public static BEID_Status BEID_GetAddress(BEID_Address Address,
BEID_Certif_Check CertifCheck)

  public static BEID_Status BEID_GetPicture(BEID_Bytes Picture,
BEID_Certif_Check CertifCheck)

  public static BEID_Status BEID_GetRawData(BEID_Raw RawData)

  public static BEID_Status BEID_SetRawData(BEID_Raw RawData)

  public static BEID_Status BEID_GetVersionInfo(BEID_VersionInfo
VersionInfo, int Signature,  BEID_Bytes SignedStatus)

  public static BEID_Status BEID_BeginTransaction()

  public static BEID_Status BEID_EndTransaction()

  public static BEID_Status BEID_SelectApplication(byte[]
Application)

  public static BEID_Status BEID_VerifyPIN(BEID_Pin PinData,
String Pin, BEID_Long TriesLeft)

  public static BEID_Status BEID_ChangePIN(  BEID_Pin Pin, String
                                             oldPin, String
                                             newPin,
                                             BEID_Long TriesLeft)

  public static BEID_Status BEID_GetPINStatus( BEID_Pin PinData,
                                                 BEID_Long
                                                 TriesLeft, int
                                                 Signature,
                                                 BEID_Bytes
                                                 SignedStatus)

  public static BEID_Status BEID_ReadFile( byte[] FileID,
BEID_Bytes OutData, BEID_Pin PinData)

  public static BEID_Status BEID_WriteFile(byte[] FileID, byte[]
InData, BEID_Pin PinData)

  public static BEID_Status BEID_FlushCache()

  public static BEID_Status BEID_SendAPDU(byte[] CmdAPDU,
BEID_Pin PinData, BEID_Bytes RespAPDU)
 }
```

Applet Parameter :

Naam	Waarden	Naam
Card Reader name	Lezer	String
OCSP checking	OCSP	Zie 2.11
CRL checking	CRL	Zie 2.11

HTLM snippet :

```
<applet
  codebase = "."
  archive  = "eidlib.jar"
  code     = "be.belgium.eid.BEID_Applet.class"
  name     = "BEIDApplet"
  width    = "0"
  height   = "0"
  hspace   = "0"
  vspace   = "0"
>
<param name="Reader" value="">
<param name="OCSP" value="0">
<param name="CRL" value="0">
</applet>
```

3.3. ActiveX API

3.3.1. Interfacedefinities

De interfacedefinities zijn in IDL-formaat.
Control Name: **EIDLibCtrl.EIDlib**

```
Interface IRetStatus
{
     long * GetGeneral( );
     long * GetPCSC( );
     long * GetSystem( );
     VARIANT * GetCardSW( );     ‘ VARIANT type : VT_ARRAY | VT_UI1
}

Interface IMapCollection
{
     VARIANT * GetValue( [IN] BSTR strKey );
     void SetValue( [IN] BSTR strKey, [IN] VARIANT vtValue );
     long * GetCount( );
}

Interface ICertif
{
     VARIANT * GetCertif( );      ‘ VARIANT type : VT_ARRAY | VT_UI1
     BSTR * GetLabel( );
     long * GetStatus( );         ‘ Validation status code see 2.13
}
```

```
Interface ICertifCheck
{
     long * GetPolicy( );
     long * GetSignatureCheck( );        ‘ Validation status code see 2.13
     VARIANT * GetCertificates( );       ‘ Array of ICertif, VARIANT type :
VT_ARRAY | VT_DISPATCH
}
Interface IPin
{
  void   SetPinType([IN] long Type);     ‘ BEID_PIN_TYPE_PKCS15 or
BEID_PIN_TYPE_OS
  void   SetID([IN]VARIANT ID);          ‘ VARIANT type : VT_UI1
  void   SetUsageCode([IN]long usageCode);
  void   SetshortUsage([IN]BSTR shortUsage); ‘ May be NULL for usage
known by the middleware
  void   SetlongUsage([IN]BSTR longUsage);   ‘ May be NULL for usage
known by the middleware
}
Interface IRaw
{
     void SetIDData([in] VARIANT vtID); ‘ VARIANT type : VT_ARRAY |
VT_UI1
     VARIANT *GetIDData(); ‘ VARIANT type : VT_ARRAY | VT_UI1
     void SetIDSigData([in] VARIANT vtIDSigData); ‘ VARIANT type :
VT_ARRAY | VT_UI1
     VARIANT *GetIDSigData(); ‘ VARIANT type : VT_ARRAY | VT_UI1
     void SetAddrData([in] VARIANT vtAddrData); ‘ VARIANT type :
VT_ARRAY | VT_UI1
     VARIANT * GetAddrData(); ‘ VARIANT type : VT_ARRAY | VT_UI1
     void SetAddrSigData([in] VARIANT vtAddrSigData); ‘ VARIANT type
: VT_ARRAY | VT_UI1
     VARIANT *GetAddrSigData(); ‘ VARIANT type : VT_ARRAY | VT_UI1
     void SetPictureData([in] VARIANT vtPictureData); ‘ VARIANT type
: VT_ARRAY | VT_UI1
     VARIANT *GetPictureData(); ‘ VARIANT type : VT_ARRAY | VT_UI1
     void SetRNData([in] VARIANT vtRNData); ‘ VARIANT type : VT_ARRAY
| VT_UI1
     VARIANT *GetRNData(); ‘ VARIANT type : VT_ARRAY | VT_UI1
     void SetCardData([in] VARIANT vtCardData); ‘ VARIANT type :
VT_ARRAY | VT_UI1
     VARIANT *GetCardData(); ‘ VARIANT type : VT_ARRAY | VT_UI1
     void SetTokenInfoData([in] VARIANT vtTokenInfoData); ‘ VARIANT
type : VT_ARRAY | VT_UI1
     VARIANT *GetTokenInfoData(); ‘ VARIANT type : VT_ARRAY | VT_UI1
     void SetChallengeData([in] VARIANT vtChallengeData); ‘ VARIANT
type : VT_ARRAY | VT_UI1
     VARIANT *GetChallengeData(); ‘ VARIANT type : VT_ARRAY | VT_UI1
     void SetResponseData([in] VARIANT vtResponseData); ‘ VARIANT
type : VT_ARRAY | VT_UI1
     VARIANT * GetResponseData(); ‘ VARIANT type : VT_ARRAY | VT_UI1
}
```

```
IRetStatus *
     Init(
     [IN] BSTR strReaderName,
     [IN] long lOCSP,
     [IN] long lCRL,
     [OUT] long * plHandle
     );

IRetStatus *
     Exit(
     );

‘ De IMapcollection bevat Key/Value paren van de opgegeven data.
‘ De sleutels zijn vooraf gedefinieerd om de interface niet opnieuw te moeten definiëren voor
nieuwe versies
‘ Sleutels: CardNumber, ChipNumber, BeginValidityDate, EndValidityDate, IssuingMunicipality,
          NationalNumber, Name, FirstName1, FirstName2, FirstName3, Nationality, BirthPlace,
          BirthDate, Gender, NobilityTitle, DocumentType, WhiteCane, YellowCane,
          ExtendedMinority
IRetStatus *
     GetID(
     [OUT] IMapCollection ** ppMapCollection, ‘ Allocated by the Toolkit
     [OUT] ICertifCheck ** ppCertifCheck      ‘ Allocated by the Toolkit
);

‘ Keys:    Street, HouseNumber, BoxNumber, ZIPCode, Municipality, Country
IRetStatus *
     GetAddress(
     [OUT] IMapCollection ** ppMapCollection,   ‘ Allocated by the Toolkit
     [OUT] ICertifCheck ** ppCertifCheck        ‘ Allocated by the Toolkit
     );

‘ Keys:    Picture
IRetStatus *
     GetPicture(
     [OUT] IMapCollection ** ppMapCollection,   ‘ Allocated by the Toolkit
     [OUT] ICertifCheck ** ppCertifCheck        ‘ Allocated by the Toolkit
     );

IRetStatus *
     GetRawData (
     [OUT] IRaw ** ppRaw                        ‘ Allocated by the library
     );

IRetStatus *
     SetRawData (
     [IN] IRaw * pRaw                           ‘
     );

IRetStatus * GetVersionInfo(
     [IN] BOOL bSignature,
     [OUT] IMapCollection ** ppMapCollection,   ‘ Allocated by the Toolkit
     [OUT] VARIANT * pvSignature                ‘ VARIANT type :
VT_ARRAY | VT_UI1
     );
```

```
IRetStatus *
     BeginTransaction( );

IRetStatus *
     EndTransAction( );

IRetStatus *
     FlushCache( );

IRetStatus *
     SelectApplication(
     [IN] VARIANT vtApplication    ‘ VARIANT type : VT_ARRAY | VT_UI1
     );

IRetStatus *
     SendAPDU(
     [IN] VARIANT vtCommand,       ‘ VARIANT type : VT_ARRAY | VT_UI1
     [IN] IPin * pin,
     [OUT] VARIANT * vtResponse    ‘ VARIANT type : VT_ARRAY | VT_UI1
     );

IRetStatus *
     VerifyPin(
          [IN] IPin * pin,
          [IN] BSTR strPin,
          [OUT] long * plTriesLeft
          );

IRetStatus *
     ChangePin(
     [IN] IPin * pin,
     [IN] BSTR strOldPin,
     [IN] BSTR strNewPin ,
     [OUT] long * plTriesLeft);

IRetStatus *
     GetPinStatus(
     [IN] IPin * pin,
     [IN] BOOL bSignature,
     [OUT] VARIANT * pvSignature,  ‘ VARIANT type : VT_ARRAY | VT_UI1
     [OUT] long * plTriesLeft);

IRetStatus *
     ReadFile(
     [IN] IPin * pin,
     [IN] VARIANT strFileID,       ‘ VARIANT type : VT_ARRAY | VT_UI1
     [OUT] VARIANT * pvtData       ‘ VARIANT type : VT_ARRAY | VT_UI1
     );

IRetStatus *
     WriteFile(
     [IN] IPin * pin,
     [IN] VARIANT strFileID,       ‘ VARIANT type : VT_ARRAY | VT_UI1
     [IN] VARIANT vtData           ‘ VARIANT type : VT_ARRAY | VT_UI1
     );
```

4.. Installatie

4.1. Installatiepakket

Het installatiepakket2 bevat de volgende bestanden :

C library	
OS	Directory met de run-time voor de verschillende besturingssystemen ▪ ***Windows*** ▪ ***Linux***
eidlib.h, eiddefines.h	C library header bestand voor C calling programs
OS/eidlib. ***x***	C library te koppelen met (toegangspunten gedeelde libraries), waarbij ***x*** de extensie van de gedeelde library is ▪ ***LIB*** voor Visual C++
test.cpp	C testprogramma
ActiveX	
Windows\VB	VB voorbeeld van oproepen ActiveX control
Java	
Java/Test.java	Java-testprogramma
Java/BEIDCard.html	JavaScript-testpagina

4.2. Run-time

De run-time is vereist voor alle omgevingen.

Hij moet worden geïnstalleerd zoals beschreven in het document "Belgian eID Run-time Users guide".

4.3. C Library

De applicatie moet het bestand "eidlib.h" van de Toolkit bevatten.

De applicatie moet worden verbonden met de gedeelde library "libeid.so" of "eidlib.dll" als de linker een directe link met de gedeelde library ondersteunt (zoals GCC), of met de toegangspunten van de library "eidlib.lib" of "libeid.a" als dat niet het geval is (zoals Visual C++).

4.4. Java

De Toolkit is compatibel met alle Java Virtual Machines van Sun vanaf 1.2.

Art.3*bis*. Specificaties Crypto Middleware

1. Doel

Dit document heeft tot doel de architectuur van de middleware-software te beschrijven en de applicatieprogrammeurs de noodzakelijke informatie te verschaffen voor het ontwikkelen van applicaties die gebruik maken van de Belgische elektronische identiteitskaart. Dit document is beperkt tot informatie voor de programmeurs betreffende de middleware van de Belgische kaart.

2. Architectuur

2.1. Interfaces

Zoals beschreven in het vorige hoofdstuk werden er twee interfaces in de middleware-software geïmplementeerd : één interface die direct kan worden opgeroepen (de PKCS#11 interface) en één interface die indirect wordt opgeroepen (de CSP).

2.1.1. De Crypto API-interface

De Microsoft(r) Cryptographic API 2.0 (CryptoAPI) stelt applicatieontwikkelaars in staat om authenticatie, codering en encryptie aan hun op Win32(r) gebaseerde applicaties toe te voegen. De applicatieontwikkelaars kunnen functies in de CryptoAPI gebruiken zonder iets af te weten van de onderliggende implementatie, zoals ze een grafische library kunnen gebruiken zonder op de hoogte te zijn van de specifieke configuratie van de grafische hardware.

Het CSP-gedeelte van de middleware legt de verbinding tussen de abstracte CryptoAP en de onderliggende PKCS#11-interface. De ontwikkelaar zal een functie van de CSP nooit direct oproepen maar altijd via de CryptoAPI.

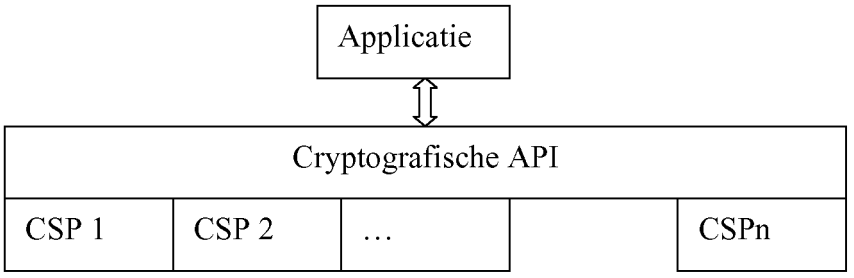
De Belgische identiteitskaart ondersteunt (momenteel) alleen operaties met digitale handtekeningen. Als de Belgische overheid later zou beslissen de gebruiker van de elektronische identiteitskaart toe te laten om sleutelmateriaal aan de kaart toe te voegen dat encryptie ondersteunt, dan zal de CSP worden uitgebreid om deze bijkomende functionaliteit mogelijk te maken. De Belgische identiteitskaart bevat ook twee sleutelparen die kunnen worden gebruikt voor digitale handtekeningen (zowel authenticatie als niet-verwerping).

Hoewel de CSP alleen digitale handtekeningen ondersteunt, is hij geregistreerd als een PROV_RSA_FULL type CSP om het gebruik van de CSP in Microsoft(r) standaardapplicaties toe te laten.

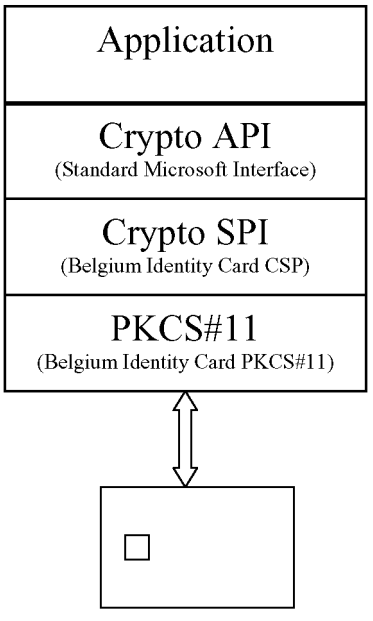
2.1.1.1. CSP high-level architectuur

De Crypto application programming interface (API) van Microsoft is een abstracte interface voor cryptografische bewerkingen. Hij schermt de gebruiker van de API af, zodat hij niet hoeft te weten hoe de cryptografische functionaliteit op een lager (kaart-) niveau is geïmplementeerd. Deze CryptoAPI-interface is onafhankelijk van de onderliggende cryptografische implementatie. In het geval van de Belgische elektronische identiteitskaart is dit een smartcard. In andere applicaties kan dit zelfs een softwareoplossing zijn.

Onder de CryptAPI-interface werd een andere interface gespecificeerd voor de ontwikkelaars van beveiligingsimplementaties. Deze interface wordt CSPI (Cryptgraphic Service Provider Interface) genoemd. Deze interface scheidt het onderliggende cryptografische apparaat van de CryptoAPI-interface. Er kan een onbeperkt aantal CSPI-implementaties beschikbaar zijn op elk systeem. Elke CSPI-implementatie is (meestal) specifiek voor een bepaald type onderliggende hardware. De volgende figuur geeft een overzicht van deze architectuur :



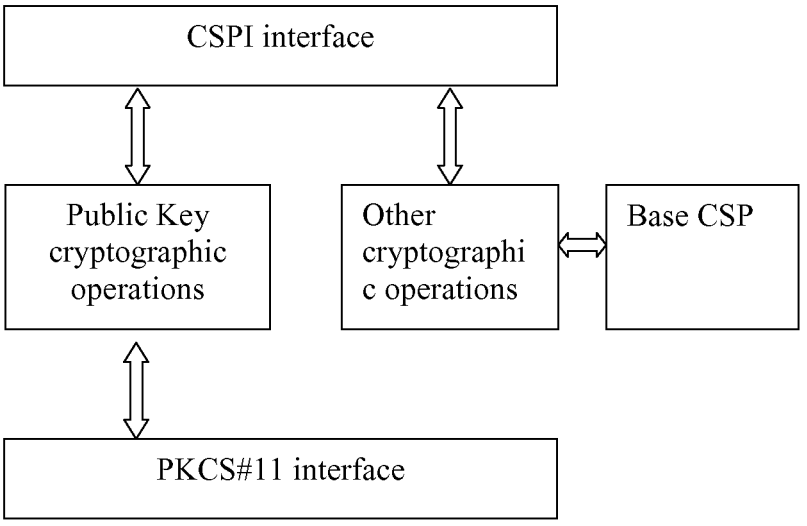
Elke CSP richt zich meestal op één specifiek type smartcard omdat de CSP meestal door de leverancier van de smartcard zelf ter beschikking wordt gesteld. De middleware van de Belgische identiteitskaart gaat anders te werk. Om de uitgever van de identiteitskaart (nl. de overheid) maximale flexibiliteit te verlenen met betrekking tot de keuze van het type smartcard dat wordt gebruikt, implementeert de CSP de proprietary interface van de smartcard niet direct, maar via een PKCS#11-interface (u vindt meer informatie over de PKCS#11-interface in sectie 2.1.2). De onderstaande figuur toont de relevante blokken :



Als de overheid later zou beslissen naar een andere fysieke smartcard over te stappen, dan moet de CSP-implementatie dus niet worden aangepast (op voorwaarde dat de nieuwe smartcard compatibel is met de bestaande).

2.1.1.2.CSP low-level architectuur

Op een lager niveau maakt de CSP-implementatie een vertaling tussen de CSPI-standaardinterface en de PKCS#11-interface. Deze architectuur wordt getoond in de onderstaande figuur :



Intern implementeert de CSP een gedeelte van de PKCS#11-interface om de cryptografische bewerkingen met de publieke sleutel uit te voeren. Deze bewerkingen zijn momenteel beperkt tot digitale handtekeningen, zowel authenticatie als niet-verwerping. Voor cryptografische bewerkingen die geen betrekking hebben op de publieke sleutel, zoals het berekenen van hash-waarden, wordt een basis-CSP gebruikt. Voor deze bewerking wordt meestal de standaard-PROV_RSA_FULL CSP gebruikt. In de praktijk zal dat meestal de Microsoft CSP zijn.

De CSP houdt een configuratiebestand bij. Dat configuratiebestand bevat kaartonafhankelijke instellingen die gekoppeld zijn aan deze elektronische identiteitskaart. De naam (het label) van de verschillende sleutels wordt bijvoorbeeld opgeslagen met een link naar het sleutel-ID voor deze sleutel. Deze informatie is voor alle Belgische elektronische identiteitskaarten hetzelfde. Er wordt dus geen configuratiebestand per kaart gebruikt op een bepaald systeem.

De gebruikerscertificaten worden opgeslagen in de certificaat-stores van het besturingssysteem. Voor gebruikerscertificaten is dat de "MY" store. Voor elk certificaat wordt een friendly name gedefinieerd (authenticatiesleutel en handtekeningsleutel). In de Microsoft-omgeving worden er certificaatcontainers gebruikt om de verbinding te maken tussen een certificaat en de bijbehorende private sleutel. De naam van deze containers wordt buiten het label van de sleutel (authenticatie of handtekening) en het serienummer van de kaart samengesteld om meer dan één identiteitskaart op hetzelfde apparaat te kunnen gebruiken.

2.1.2. De PKCS#11-interface

De PKCS#11 (v2.11)-interface wordt door niet-Microsoft-applicaties gebruikt, zoals bijvoorbeeld Netscape. Ook op maat ontwikkelde applicaties kunnen deze interface gebruiken in plaats van de CryptoAPI-interface. De PKCS#11-interface wordt soms ook Cryptoki genoemd.

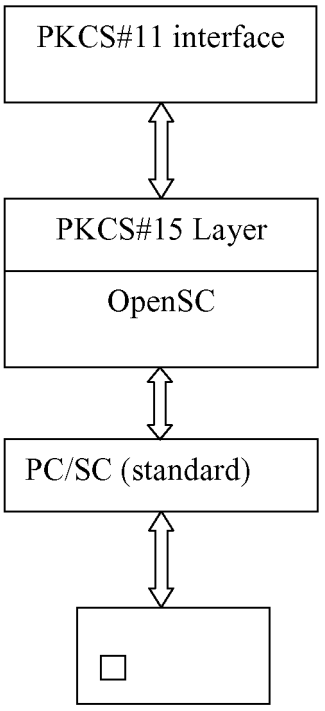
Er is een gedetailleerde beschrijving van deze interface beschikbaar op de website van RSA Laboratories (http ://www.rsasecurity.com/rsalabs/pkcs/pkcs-11/).

In overeenstemming met de wens van de overheid om zo een zo open mogelijk systeem te creëren, hebben we besloten de open source PKCS#11-implementatie van opensc (http ://www.opensc.org) te gebruiken. Deze implementatie biedt niet alleen de implementatie van de PKCS#11-interface maar ondersteunt ook de PKCS#15-kaartstructuur.

2.1.2.1. PKCS#11 high-level architectuur

De cryptoki-interface biedt een abstracte interface naar de smartcard voor applicaties die cryptografische functionaliteit nodig hebben. Deze interface wordt meestal door niet-Microsoft-applicaties gebruikt, zoals Netscape en Mozilla. Ook onafhankelijke solution providers kunnen deze interface gebruiken om cryptografische functionaliteit te implementeren in applicaties van andere leveranciers.

De figuur hieronder illustreert de PKCS#11-modules :



Zoals aangeduid in de bovenstaande figuur, implementeert de bovenlaag de PKCS#11-standaardinterface. Het is deze interface die aan de applicaties wordt getoond. Onder deze interfacelaag wordt de verbinding gelegd met de kaartstructuur in PKCS#15 en worden de vereiste commando's (APDU) via de PC/SC-standaardinterfacefuncties verstuurd.

In de OpenSC-laag werd een specifieke kaart-driver geïmplementeerd voor het gebruik van de Belgische elektronische identiteitskaart. Als er later een andere kaart zou worden gebruikt, dan moet deze driver worden vervangen.

3. Programmeurshandleiding

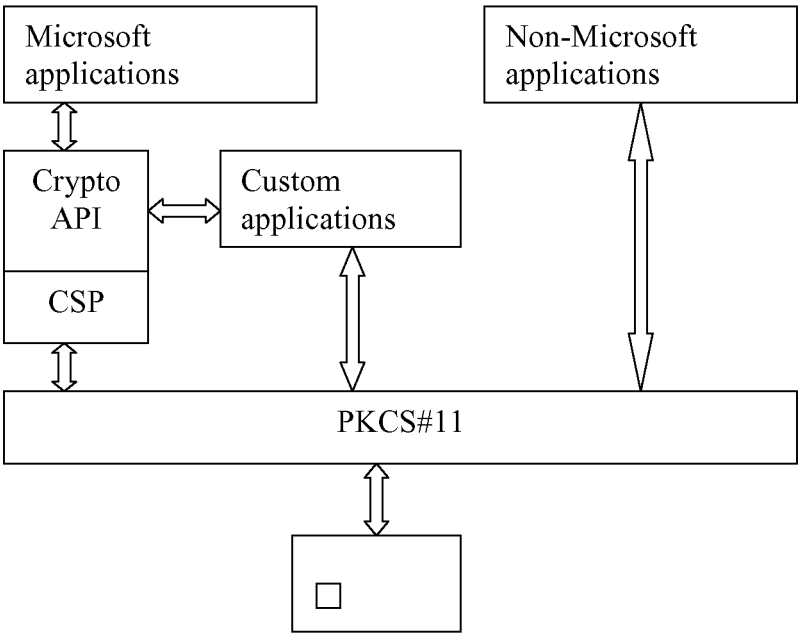
3.1. Inleiding

De middleware van de Belgische identiteitskaart is software die wordt aangebracht tussen de applicatie die beveiligingseigenschappen implementeert (digitale handtekeningen) en het apparaat dat de cryptografische operaties uitvoert (de smartcard).

De middleware zelf bestaat uit twee onafhankelijke interface-implementaties (zie figuur onderaan). Hoewel de implementaties onafhankelijk zijn, maken ze gebruik van elkaar. Voor de standaardapplicaties van Microsoft(r) (Office, Outlook...) wordt er een Cryptographic Service Provider (CSP) gecreëerd die de cryptografische bewerkingen van de smartcard implementeert. Een applicatie zal deze implementatie nooit direct oproepen maar via een standaardinterface, Crypto API genoemd. De CSP-implementatie maakt gebruik van de tweede geïmplementeerde interface, PKCS#11. Deze interface wordt gebruikt door standaardapplicaties die niet van Microsoft afkomstig zijn.

Als er een nieuwe applicatie wordt gecreëerd, is het aan de ontwikkelaar om te beslissen welke van de twee interfaces zal worden gebruikt om de gebruiker cryptografische functionaliteit ter beschikking te stellen.

Dit document beschrijft de programmeerinterfaces en de manier waarop een applicatieontwikkelaar ze kan gebruiken.



3.2. De Crypto API-interface

De Microsoft(r) Cryptographic API 2.0 (CryptoAPI) stelt applicatieontwikkelaars in staat om authenticatie, codering en encryptie aan hun op Win32(r) gebaseerde applicaties toe te voegen. De applicatieontwikkelaars kunnen functies in de CryptoAPI gebruiken zonder iets af te weten van de onderliggende implementatie, zoals ze een grafische library kunnen gebruiken zonder op de hoogte te zijn van de specifieke configuratie van de grafische hardware.

Het CSP-gedeelte van de middleware legt de verbinding tussen de abstracte CryptoAPI- en de onderliggende PKCS#11-interface. De ontwikkelaar zal de functies van de CSP nooit direct oproepen maar altijd via de CryptoAPI. De onderstaande secties geven een beschrijving van de API calls die CryptoAPI naar de CSP stuurt voor verdere verwerking. Dit document bevat geen gedetailleerde informatie over de werking van elke API call. Gelieve voor dit type informatie het Microsoft Developer Network te raadplegen.

De Belgische identiteitskaart ondersteunt alleen operaties met digitale handtekeningen. Niet alle functies die verband houden met deze cryptografische bewerkingen werden geïmplementeerd. Als de Belgische overheid later zou beslissen de gebruiker van de elektronische identiteitskaart toe te laten om sleutelmateriaal aan de kaart toe te voegen dat encryptie ondersteunt, dan zal de CSP worden uitgebreid om deze bijkomende functionaliteit mogelijk te maken. De Belgische identiteitskaart bevat ook twee sleutelparen die kunnen worden gebruikt voor digitale handtekeningen (zowel authenticatie als niet-verwerping). Dat heeft tot gevolg dat bepaalde parameters die aan de Crypto API-functies worden doorgegeven geen betekenis hebben. In de call CryptGetUserKey wordt bijvoorbeeld een parameter 'dwKeySpec' doorgegeven. Deze parameter wordt gebruikt om te definiëren welk type sleutel er wordt gevraagd, een AT_KEYEXCHANGE-sleutel of een AT_SIGNATURE-sleutel. In het geval van de CSP van de Belgische identiteitskaart volstaat deze parameter echter niet om te bepalen welke handtekeningsleutel er moet worden geladen. In dit geval moet de container die het juiste certificaat bevat naar CryptAcquireContext worden doorgegeven en dan zal een nieuwe call naar CryptGetUserKey met succes worden uitgevoerd.

Hoewel de CSP alleen digitale handtekeningen ondersteunt, is hij geregistreerd als een PROV_RSA_FULL type CSP om het gebruik van de CSP in Microsoft(r)-standaardapplicaties toe te laten. Crypto Het oproepen van API-functies die niet in de context van een digitale handtekening worden gebruikt, zal een foutwaarde genereren die aangeeft dat de API-functie niet werd geïmplementeerd.

De beschrijving in dit document heeft betrekking op versie 1.20 van de CSP.

3.2.1.CryptAcquireContext

```
BOOL WINAPI CryptAcquireContext(HCRYPTPROV *phProv,
                                LPCTSTR pszContainer,
                                LPCTSTR pszProvider,
                                DWORD dwProvType,
                                DWORD dwFlags);
```

De parameter pszContainer bevat de naam van de sleutelcontainer die een specifieke sleutel op de identiteitskaart bevat. De namen van de containers op de identiteitskaart kunnen worden verkregen via een call naar

CryptGetProvParam.

De parameter dwFlags kan worden ingesteld op de volgende waarden (volgens MSDN) :

0 (gelijk aan CRYPT_SCKEYSET)

CRYPT_VERIFYCONTEXT

CRYPT_NEWKEYSET

CRYPT_MACHINE_KEYSET

CRYPT_DELETEKEYSET

Aangezien het sleutelmateriaal van de Belgische identiteitskaart op een smartcard is opgeslagen en de gebruiker niet over de autorisaties beschikt om nieuwe sleutelparen te creëren op de kaart, worden de parameterwaarden CRYPT_NEWKEYSET, CRYPT_MACHINE_KEYSET en CRYPT_DELETEKEYSET niet ondersteund. Het gebruik van deze waarden zal de fout NTE_BAD_FLAGS genereren.

Er werd een extra waarde gedefinieerd voor deze parameter, CRYPT_SCKEYSET. Met deze waarde bepaalt de ontwikkelaar dat er een context voor de sleutel is vereist zoals gedefinieerd in de pszContainer parameter.

Voor hashing-bewerkingen alleen wordt een basis-CSP gebruikt. Als het laden van de basis-CSP zou mislukken, dan wordt de volgende foutcode gegenereerd door SetLastError() :

ERR_CANNOT_LOAD_BASE_CSP (0x1000)

3.2.2. CryptReleaseContext

```
BOOL WINAPI CryptReleaseContext(HCRYPTPROV hProv,
                                DWORD dwFlags);
```

Deze API call is geïmplementeerd zoals gedefinieerd door MSDN.

3.2.3. CryptGenerateKey

```
BOOL WINAPI CryptGenKey( HCRYPTPROV hProv,
                         ALG_ID Algid,
                         DWORD dwFlags,
                         HCRYPTKEY *phKey);
```

Aangezien het sleutelmateriaal op de Belgische identiteitskaart vooraf door de overheid werd geïnstalleerd en de gebruiker geen bijkomende sleutelparen kan creëren, is deze API call niet geïmplementeerd. Als deze functie toch wordt opgeroepen wordt de fout E_NOTIMPL gegenereerd door SetLastError ().

3.2.4.CryptDeriveKey

```
BOOL WINAPI CryptDeriveKey(HCRYPTPROV hProv,
                           ALG_ID Algid,
                           HCRYPTHASH hBaseData,
                           DWORD dwFlags,
                           HCRYPTKEY *phKey);
```

Aangezien het sleutelmateriaal op de Belgische identiteitskaart vooraf door de overheid werd geïnstalleerd en de gebruiker geen bijkomende sleutelparen kan creëren, is deze API niet geïmplementeerd. Als deze functie toch wordt opgeroepen wordt de fout E_NOTIMPL gegenereerd door SetLastError ().

3.2.5.CryptDestroyKey

```
BOOL WINAPI CryptDestroyKey(HCRYPTKEY hKey);
```

Aangezien het sleutelmateriaal op de Belgische identiteitskaart vooraf door de overheid werd geïnstalleerd en de gebruiker geen bijkomende sleutelparen kan creëren, is deze API call niet geïmplementeerd. Als deze functie toch wordt opgeroepen wordt de fout E_NOTIMPL gegenereerd door SetLastError ().

3.2.6CryptSetKeyParam

```
BOOL WINAPI CryptSetKeyParam(HCRYPTKEY hKey,
                             DWORD dwParam,
                             BYTE *pbData,
                             DWORD dwFlags);
```

Aangezien het sleutelmateriaal op de Belgische identiteitskaart vooraf door de overheid werd geïnstalleerd en de gebruiker geen bijkomende sleutelparen kan creëren, is deze API call niet geïmplementeerd. Als deze functie toch wordt opgeroepen wordt de fout E_NOTIMPL gegenereerd door SetLastError ().

3.2.7. CryptGetKeyParam

```
BOOL WINAPI CryptGetKeyParam(HCRYPTKEY hKey,
                             DWORD dwParam,
                             BYTE *pbData,
                             DWORD *pcbData,
                             DWORD dwFlags);
```

Aangezien het sleutelmateriaal op de Belgische identiteitskaart vooraf door de overheid werd geïnstalleerd en de gebruiker geen bijkomende sleutelparen kan creëren, is deze API call niet geïmplementeerd. Als deze functie toch wordt opgeroepen wordt de fout E_NOTIMPL gegenereerd door SetLastError ().

3.2.8.CryptSetProvParam

```
BOOL WINAPI CryptSetProvParam(HCRYPTPROV hProv,
                              DWORD dwParam,
                              BYTE *pbData,
                              DWORD dwFlags);
```

Volgens de MSDN-documentatie kan de parameter dwParam op de volgende waarden worden ingesteld :

PP_CLIENT_HWND

PP_KEYSET_SEC_DESCR

Deze laatste parameter heeft geen betekenis omdat het sleutelmateriaal in het geval van de Belgische identiteitskaart in de smartcard zelf wordt opgeslagen in plaats van in de registry. Deze parameter wordt bijgevolg genegeerd.

3.2.9.CryptGetProvParam

```
BOOL WINAPI CryptGetProvParam(HCRYPTPROV hProv,
                              DWORD dwParam,
                              BYTE *pbData,
                              DWORD *pcbData,
                              DWORD dwFlags);
```

Deze API call is geïmplementeerd zoals beschreven in de MSDN-documentatie met uitzondering van de parameter PP_KEYSET_SEC_DESCR die wordt genegeerd.

Voor de parameter PP_IMPTYPE wordt de waarde CRYPT_IMPL_MIXED gegenereerd omdat de handtekening-bewerking door de hardware (nl. de smartcard) wordt uitgevoerd en de hashing-bewerking door de base cryptografic provider wordt uitgevoerd.

3.2.10.CryptSetHashParam

```
BOOL WINAPI CryptSetHashParam(HCRYPTHASH hHash,
                              DWORD dwParam,
                              BYTE *pbData,
                              DWORD dwFlags);
```

Deze API call is geïmplementeerd zoals beschreven in de MSDN-documentatie.

De parameter dwParam = HP_HASHVAL werd geïmplementeerd maar moet met de nodige voorzichtigheid worden gebruikt. Deze parameter werd gedefinieerd om applicaties de mogelijkheid te bieden hash-waarden te ondertekenen zonder dat ze toegang hebben tot de basisdata. Omdat de applicatie (en de gebruiker nog minder) onmogelijk kan weten wat er wordt ondertekend, is dit een inherent riskante bewerking.

3.2.11. CryptGetHashParam

```
BOOL WINAPI CryptGetHashParam(HCRYPTHASH hHash,
                              DWORD dwParam,
                              BYTE *pbData,
                              DWORD *pcbData,
                              DWORD dwFlags);
```

Deze API call is geïmplementeerd zoals beschreven in de MSDN-documentatie.

3.2.12. CryptExportKey

```
BOOL WINAPI CryptExportKey(HCRYPTKEY hKey,
                           HCRYPTKEY hExpKey,
                           DWORD dwBlobType,
                           DWORD dwFlags,
                           BYTE *pbData,
                           DWORD *pcbDataLen);
```

Deze functie kan worden gebruikt om de publieke sleutel die met de parameter hKey is geassocieerd te exporteren. Er kan een handle voor een publieke sleutel worden verkregen via een call naar CryptGetUserKey. Aangezien de private sleutels op een smartcard zijn opgeslagen en het exporteren van private sleutels niet is toegelaten, kan alleen PUBLICKEYBLOB als dwBlobType worden gedefinieerd. Aangezien alleen publieke sleutels kunnen worden geëxporteerd, wordt de parameter hExpKey niet gebruikt en moet hij dus op NULL worden ingesteld. De publieke sleutel wordt gegeven door de parameter pbData. Om de lengte van de data die worden gegeven te verkrijgen kan de parameter pbData op NULL worden ingesteld. De lengte van de data die worden gegeven, wordt dan in pcbDataLen geplaatst. Als de buffer die naar deze functie wordt doorgegeven niet voldoende groot is, wordt de fout ERROR_MORE_DATA gegeven en wordt de juiste bufferlengte in de parameter pcbDataLen geplaatst.

3.2.13. CryptImportKey

```
BOOL WINAPI CryptImportKey(HCRYPTPROV hProv,
                           BYTE *pbData,
                           DWORD dwDataLen,
                           HCRYPTKEY hPubKey,
                           DWORD dwFlags,
                           HCRYPTKEY *phKey);
```

Aangezien het sleutelmateriaal op de Belgische identiteitskaart vooraf door de overheid werd geïnstalleerd en de gebruiker geen bijkomende sleutelparen kan creëren, is deze API call niet geïmplementeerd. Als deze functie toch wordt opgeroepen wordt de fout E_NOTIMPL gegenereerd door SetLastError ().

3.2.14.CryptEncrypt

```
BOOL WINAPI CryptEncrypt(HCRYPTKEY hKey,
                         HCRYPTHASH hHash,
                         BOOL Final,
                         DWORD dwFlags,
                         BYTE *pbData,
                         DWORD *pcbData,
                         DWORD cbBuffer);
```

Het gebruik van de sleutels zoals ze door de Belgische overheid werden gedefinieerd, ondersteunt momenteel geen encryptie. Daarom werd deze API call niet geïmplementeerd. Als deze functie toch wordt opgeroepen wordt de fout E_NOTIMPL gegenereerd door SetLastError ().

Als er in de toekomst sleutelmateriaal aan de elektronische identiteitskaart kan worden toegevoegd dat encryptie ondersteunt, dan zal deze functie eveneens worden geïmplementeerd.

3.2.15.CryptDecrypt

```
BOOL WINAPI CryptDecrypt(HCRYPTKEY hKey,
                         HCRYPTHASH hHash,
                         BOOL Final,
                         DWORD dwFlags,
                         BYTE *pbData,
                         DWORD *pcbData);
```

Het gebruik van de sleutels zoals ze door de Belgische overheid werden gedefinieerd, ondersteunt momenteel geen encryptie. Daarom werd deze API call niet geïmplementeerd. Als deze functie toch wordt opgeroepen wordt de fout E_NOTIMPL gegenereerd door SetLastError ().

Als er in de toekomst sleutelmateriaal aan de elektronische identiteitskaart kan worden toegevoegd dat encryptie ondersteunt, dan zal deze functie eveneens worden geïmplementeerd.

3.2.16.CryptCreateHash

```
BOOL WINAPI CryptCreateHash(HCRYPTPROV hProv,
                            ALG_ID Algid,
                            HCRYPTKEY hKey,
                            DWORD dwFlags,
                            HCRYPTHASH *phHash);
```

Deze API call is geïmplementeerd zoals beschreven in de MSDN-documentatie. Er kan één bijkomende fout worden gegenereerd via SetLastError () :

ERR_INVALID_PROVIDER_HANDLE (0x1001)

Deze fout geeft aan dat de handle zoals hij door hProv wordt gespecificeerd, niet kon worden gevonden (vb. hij werd niet gecreëerd met CryptAcquireContext ())

De verwerking van deze call wordt gedelegeerd naar een basis-CSP.

3.2.17.CryptHashData

```
BOOL WINAPI CryptHashData(HCRYPTHASH hHash,
                          BYTE *pbData,
                          DWORD cbData,
                          DWORD dwFlags);
```

Deze API call is geïmplementeerd zoals beschreven in de MSDN-documentatie. In de parameter dwFlags kan één waarde (behalve 0) worden gespecificeerd : CRYPT_USERDATA. Hij is al of niet geïmplementeerd afhankelijk van de basis-CSP die wordt gekozen. De Microsoft Base CSP implementeert deze parameter bijvoorbeeld niet.

De verwerking van deze call wordt gedelegeerd naar een basis-CSP.

3.2.18. CryptHashSessionKey

```
BOOL WINAPI CryptHashSessionKey(HCRYPTHASH hHash,
                    HCRYPTKEY hKey,
                    DWORD dwFlags);
```

Aangezien sommige van de onderliggende calls die vereist zijn om deze functie te gebruiken momenteel niet zijn geïmplementeerd door deze CSP, is deze call evenmin beschikbaar. Als deze functie toch wordt opgeroepen wordt de fout E_NOTIMPL gegenereerd door SetLastError ().

3.2.19.CryptSignHash

```
BOOL WINAPI CryptSignHash(HCRYPTHASH hHash,
                    DWORD dwKeySpec,
                    LPCTSTR sDescription,
                    DWORD dwFlags,
                    BYTE *pbSignature,
                    DWORD *pdwSigLen);
```

Deze API call is geïmplementeerd zoals beschreven in de MSDN-documentatie. Als de functie wordt opgeroepen, wordt er een poging ondernomen om verbinding te maken met en in te loggen op de Belgische identiteitskaart (smartcard). Als een van deze bewerkingen faalt, dan kan de volgende fout worden gegenereerd via SetLastError () :

ERR_CANNOT_LOGON_TO_TOKEN (0x1004)

Om de opgegeven hash-data te ondertekenen, moet er bepaalde informatie (vb. lengte sleutel) worden gelezen van de smartcard. Als er tijdens deze bewerking een fout gebeurt, dan wordt de volgende fout gegenereerd via SetLastError () :

ERR_CANNOT_GET_TOKEN_SLOT_INFO (0x1003)

Het ondertekeningsmechanisme dat wordt gebruikt om digitale handtekeningen te produceren, is CKM_RSA_PKCS. In de PKCS#11-documentatie vindt u meer gedetailleerde informatie over dit mechanisme.

Momenteel kunnen de volgende hashing-algoritmen worden gebruikt om data te tekenen : MD2, MD4, MD5, SHA-1 en SSL3 SHAMD5. Hoewel de MDx hashing-algoritmen nog steeds beschikbaar zijn voor achterwaartse compatibiliteit, wordt het gebruik van SHA-1 aanbevolen voor nieuwe applicaties.

3.2.20.CryptDestroyHash

```
BOOL WINAPI CryptDestroyHash(HCRYPTHASH hHash);
```

Deze API call is geïmplementeerd zoals beschreven in de MSDN-documentatie.

3.2.21. CryptVerifySignature

```
BOOL WINAPI CryptVerifySignature(HCRYPTHASH hHash,
                    BYTE *pbSignature,
                    DWORD dwSigLen,
                    HCRYPTKEY hPubKey,
                    LPCTSTR sDescription,
                    DWORD dwFlags);
```

Deze functie is geïmplementeerd om praktische redenen. Deze call wordt gedelegeerd naar de basis-CSP.

3.2.22. CryptGenRandom

```
BOOL WINAPI CryptGenRandom(HCRYPTPROV hProv,
                    DWORD dwLen,
                    BYTE *pbBuffer);
```

Deze API call is geïmplementeerd zoals beschreven in de MSDN-documentatie. De data die via pbBuffer worden ingegeven, zullen worden gebruikt als basis voor random-generatie.

3.2.23. CryptGetUserKey

```
BOOL CryptGetUserKey(HCRYPTPROV hProv,
                    DWORD dwKeySpec,
                    HCRYPTKEY *phUserKey);
```

Deze call geeft een handle voor de publieke sleutel van de sleutelcontainer die werd gedefinieerd via CryptAcquireContext. Het volstaat niet AT_SIGNATURE te specificeren voor de parameter dwKeySpec omdat de CSP met deze informatie nog steeds niet kan bepalen welke handtekeningsleutel hij moet geven. Daarom moet de te laden sleutel eerst worden gespecificeerd via CryptAcquireContext.

3.2. 24.CryptDuplicateHash

```
BOOL WINAPI CryptDuplicateHash(HCRYPTHASH hHash,
                    DWORD *pdwReserved,
                    DWORD dwFlags,
                    HCRYPTHASH phHash);
```

Deze API call is geïmplementeerd zoals beschreven in de MSDN-documentatie.

3.2.25.CryptDuplicateKey

```
BOOL WINAPI CryptDuplicateKey(HCRYPTKEY hKey,
                              DWORD *pdwReserved,
                              DWORD dwFlags,
                              HCRYPTKEY* phKey);
```

Aangezien het sleutelmateriaal op de smartcard is opgeslagen en niet kan worden opgehaald, heeft deze functie geen zin en daarom werd deze API call niet geïmplementeerd. Als deze functie toch wordt opgeroepen wordt de fout E_NOTIMPL gegenereerd door SetLastError ().

3.3. De PKCS#11-interface

De PKCS#11 (v2.11)-interface wordt gebruikt door niet-Microsoft-applicaties zoals bijvoorbeeld Netscape. Ook op maat ontwikkelde applicaties kunnen deze interface gebruiken in plaats van de CryptoAPI-interface. De PKCS#11-interface wordt soms ook Cryptoki genoemd.

Er is een gedetailleerde beschrijving van deze interface beschikbaar op de website van RSA Laboratories (http ://www.rsasecurity.com/rsalabs/pkcs/pkcs-11/).

3.3.1. Geïmplementeerde API calls

3.3.1.1. Algemene functies

C_Initialize,

C_Finalize

C_GetInfo

C_GetFunctionList

3.3.1.2.Functies voor het beheer van slot en token

C_GetSlotList

C_GetSlotInfo

C_GetTokenInfo

C_GetMechanismList

C_GetMechanismInfo

C_WaitForSlotEvent

C_SetPin

3.3.1.3. Functies voor het beheer van sessies

C_OpenSession

C_CloseSession

C_CloseAllSessions

C_GetSessionInfo

C_Login

C_Logout

3.3.1.4. Functies voor het beheer van objecten

C_FindObjectsInit

C_FindObjects

C_FindObjectsFinal

C_GetAttributeValue

3.3.1.5. Functies voor ondertekening

C_SignInit

C_Sign

C_SignUpdate

C_SignFinal

3.3.1.6. Digest-functies

C_DigestInit

C_Digest

C_DigestUpdate

C_DigestFinal

3.3.1.7. Functies voor random-generatie (worden binnenkort bevestigd)

C_SeedRandom

C_GenerateRandom

3.3.2. Ondersteunde handtekeningmechanismen

Voor handtekeningen :

- CKM_RSA_X_509

- CKM_RSA_PKCS : zowel ASN.1-wrapped als zuivere hashes (MD5, SHA1, SHA1+MD5, RIPEMD160, indien er 20 bytes worden gegeven, veronderstelt men een SHA-1 hash)

- CKM_RIPEMD160_RSA_PKCS, CKM_SHA1_RSA_PKCS, CKM_MD5_RSA_PKCS

- Als ze door de elektronische ID-kaart worden ondersteund, worden de volgende handtekeningmechanismen ook door de middleware ondersteund : CKM_RSA_PKCS_PSS, CKM_SHA1_RSA_PKCS_PSS

Voor digests :

CKM_SHA_1, CKM_RIPEMD160, CKM_MD5

3.3.3. Slot- en token-informatie

Voor elke PIN zal er een virtueel slot/token zijn (in het geval van de Belgische elektronische identiteitskaart betekent dit dus 1 slot/token).

De publieke sleutels, private sleutels en certificaten die bij elkaar horen zullen hetzelfde CKA_ID-objectattribuut hebben.

3.3.4. Gedrag in het geval van een pinpad-lezer

In dit geval zal CK_TOKEN_INFO over de flag CKF_PROTECTED_AUTHENTICATION_PATH beschikken en de applicatie moet een NULL PIN geven met C_Login. Als er een C_Login wordt opgeroepen, zal de PKCS#11 library de gebruiker een dialoogvenster voorstellen met het verzoek de PIN in te geven op het pinpad om een handtekening of identificatie te plaatsen.

3.3.5. Gedrag met een niet-verwerpingssleutel

Als er met deze sleutel een handtekening wordt gevraagd, zal de PKCS#11 library zelf een GUI tonen om de gebruiker te vragen zijn PIN in te geven, of om de gebruiker te vragen zijn PIN in de pinpad-lezer in te geven.

4. Referenties

[ISO/IEC 7816-1] Identification Cards - Integrated Circuit Cards with Contacts Part1 : Physical Characteristics

[ISO/IEC 7816-2] Identification Cards - Integrated Circuit Cards with Contacts Part2 : Dimensions and Location of Contacts

[ISO/IEC 7816-3] Identification Cards - Integrated Circuit Cards with Contacts Part3 : Electronic Signals and Transmission Protocols

[ISO/IEC 7816-4] Identification Cards - Integrated Circuit Cards with Contacts Part4 : Inter-industry Commands for Interchange

[ISO/IEC 7816-5] Identification Cards - Integrated Circuit Cards with Contacts Part5 : Numbering System for Application Identifiers

[CEN/CWA 14170] Electronic Signature - Security Requirements Secure Signature Creation Applications

[CEN/CWA 14171] Electronic Signature - Security Requirements Secure Signature Verification Applications

[RSAS/PKCS#11] Public Key Cryptographic Standards Cryptographic Token Interface Standard

[RSAS/PKCS#15] Public Key Cryptographic Standards Cryptographic Token Information Standard

[EID/READERS 2.7.3] Belgian Electronic Identity Card Readers Technical Compatibility v 2.7.3

[EID/CRYPTO 1.4.0] Belgian Electronic Identity Card Security & Crypto Middleware v 1.4.0

[EID/IDENTITY 1.0.0] Belgian Electronic Identity Card Data & Identity Middleware v 1.0.0

[EID/CARD 2.0.0] Belgian Electronic Identity Card Card and Applet Specifications v 2.0.0

[EN45014] Conformity Assessment Generic Criteria (CE)

[EN60950] Information Processing Equipment Security

[EN50081] EMC (Electro-Magnetic Compatibility) Emission Generic Criteria

[EN50081] EMC (Electro-Magnetic Compatibility) Immunity Generic Criteria

[EN55022] REC (Radio-Electric Compatibility) Perturbations Generic Criteria

Gezien om te worden gevoegd bij Ons besluit van 7 december 2006 tot vaststelling van de specificaties en registratieprocedure van de leesapparatuur voor de elektronische identiteitskaart en tot wijziging van het koninklijk besluit van 13 februari 1998 houdende specificaties van de leesapparatuur voor de sociale identiteitskaart.

ALBERT

Van Koningswege :

De Minister van Binnenlandse Zaken,
P. DEWAEL

De Minister van Economie,
M. VERWILGHEN

De Minister van Sociale Zaken,
R. DEMOTTE

De Minister van Middenstand en Landbouw,
Mevr. S. LARUELLE

De Minister van Werk,

P. VANVELTHOVEN

Nota's

(1) Sommige kaarten bevatten alleen de twee voornamen samen; in dat geval worden ze beide in dit veld opgegeven en zal het veld tweede voornaam leeg zijn.

(2) Sommige kaarten bevatten alleen de twee voornamen samen; in dat geval worden ze beide in dit veld opgegeven en zal het veld tweede voornaam leeg zijn.

(3) Dit pakket kan zich in een archief bevinden zoals een ZIP-, TAR- of GZ-bestand

Bijlage II

MODELFORMULIER VOOR DE REGITSRATIE VAN EEN LEESAPPARAAT VOOR DE BELGISCHE ELEKTRONISCHE IDENTITEITSKAART (eID)

Gebruiksaanwijzing

Dit document bevat de inventaris van de verschillende documenten en de opgave van de gegevens die moeten worden ingevuld en aan de overheid moeten worden overgemaakt met het oog op het bekomen van een registratienummer voor een leesapparaat voor de Belgische elektronische identiteitskaart. Alle beschreven punten en gevraagde informatie moeten zo volledig mogelijk worden beantwoord.

Het model van het registratieformulier kan worden gedownload op de volgende URL : http ://eid.belgium.be

Het volledig en duidelijk ingevuld registratieformulier met inbegrip van alle noodzakelijk bijlagen moet worden opgestuurd :

a) per post naar het volgende adres :

Federale Overheidsdienst Informatie-en Commnicatietechnologie

Maria-Theresiastraat 1-3

1000 Brussel

b) per e-mail :

registratie@fedict.be

I. IDENTIFICATIE VAN DE AANVRAGER

De identificatie van de aanvrager van de registratie van de volledige oplossing, dit wil zeggen diegene die verantwoordelijk is voor de integratie van de oplossing.

Volledige benaming onderneming : .....

.....

Adres maatschappelijke zetel : .....

.....

Juridische vorm : .....

.....

Contactpersoon : .....

.....

Telefoon van de contactpersoon : .....

.....

e-mail : .....

.....

II. OMSCHRIJVING VAN DE OPLOSSING

In dit onderdeel wordt een omschrijving gegeven van de oplossing die ter registratie wordt ingediend met aanduiding van de essentiële eigenschappen en de omgeving ervan.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

III. Normatieve specificaties

In dit onderdeel moet worden aangeduid of de vereiste normen worden gerespecteerd en moet voor elke rubriek het bewijs daarvan als bijlage worden gevoegd, hetzij op grond van een gelijkvormigheidattest afgeleverd door een certificatie-instelling, hetzij op grond van een schriftelijke verklaring van eenvormigheid.

*** TABEL ***

IV. RESULTATEN VAN DE TESTSCENARIOS

V. HANDBOEKEN

Gedaan te, ... op .....

Naam : .....

Handtekening : .....

Gezien om te worden gevoegd bij Ons besluit van 7 december 2006 tot vaststelling van de specificaties en registratieprocedure van de leesapparatuur voor de elektronische identiteitskaart en tot wijziging van het koninklijk besluit van 13 februari 1998 houdende specificaties van de leesapparatuur voor de sociale identiteitskaart.

ALBERT

Van Koningswege :

De Minister van Binnenlandse Zaken,

P. DEWAEL

De Minister van Economie,

M. VERWILGHEN

De Minister van Sociale Zaken,

R. DEMOTTE

De Minister van Middenstand en Landbouw,

Mevr. S. LARUELLE

De Minister van Werk,

P. VANVELTHOVEN

ANNEXE III

Modèle de label et conditions d'utilisation.

1. Le label visé à l'article 8 de cet l'arrêté royal :

- est strictement et limitativement lié à un appareil de lecture qui, pour ce qui est de la carte d'identité électronique, a satisfait et satisfait aux normes et exigences techniques et fonctionnelles imposées;

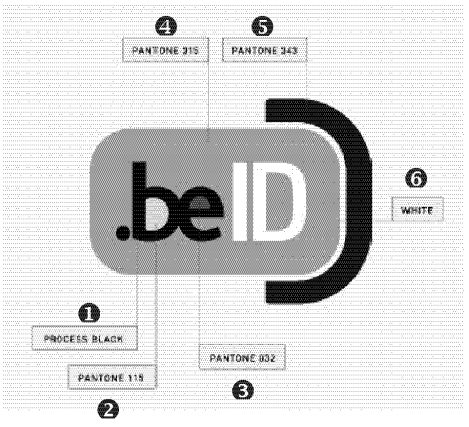
- témoigne, sur l'initiative du fournisseur, de la conformité de l'appareil de lecture sur lequel il est apposé aux dites normes et exigences;

- est de nature à éliminer dans le chef de l'utilisateur le doute quant à la compatibilité des appareils de lecture avec la carte d'identité électronique;

- vise à soutenir la démarche volontariste des fournisseurs se faisant enregistrer.

2. Le label est figuré par le logo ici reproduit :

version en couleurs : version monochrome :



2.1. La version en couleurs est obligatoirement composée, selon les indications susmentionnées, des couleurs spécifiques suivantes :

- (1)

Pantone: Process Black

CMYK : C 0, M 0, Y 0, K 100

RGB : R 0, G 0, B 0

- (2)

Pantone : 115

CMYK : C 0, M 8,5, Y 79, K 0

RGB : R 255, G 233, B 0

- (3)

Pantone : 032

CMYK : C 0, M 91, Y 87, K 0

RGB : R 246, G 0, B 0

- (4)

Pantone : 338

CMYK : C 47, M 0, Y 32, K 0

RGB : R 135, G 207, B 163

- (5)

Pantone : 343

CMYK : C 98, M 0, Y 72, K 61

RGB : R 2, G 56, B 37

- (6)

Blanc

CMYK : C 0, M 0, Y 0, K 0

RGB : R 255, G 255, B 255

La version en couleurs doit être préférée à la version monochrome; elle doit être reproduite sur un fond blanc.

2.2. La version monochrome, lorsque la version en couleur n'est pas possible, est obligatoirement composée de noir et de blanc, avec une trame à 35% du noir pour la partie grisée.

Elle doit être reproduite sur un fond blanc.

3. Le label doit être reproduit, selon le modèle supra, tel quel, sans altération de la forme, des couleurs, de la lisibilité et des proportions.

BIJLAGE III

Labelmodel en gebruiksvoorwaarden.

1. Het label bedoeld in artikel 8 van dit koninklijk besluit :

- is strikt en beperkend verbonden met een leesapparaat dat, voor wat de elektronische identiteitskaart betreft, heeft voldaan en voldoet aan de opgelegde functionele en technische normen en vereisten;

- getuigt, op initiatief van de leverancier, van de conformiteit van het leesapparaat waarop het wordt aangebracht met bovenvermelde normen en vereisten;

- neemt bij de gebruiker elke twijfel weg rond de compatibiliteit van de leesapparaten met de elektronische identiteitskaart;

- heeft tot doel de voluntaristische houding te ondersteunen van de leveranciers die zich laten registreren.

2. Het label wordt afgebeeld door het logo dat hier wordt weergegeven :

kleurenversie : monochrome versie :

2.1. De kleurenversie is, volgens bovenvermelde aanwijzingen, verplicht samengesteld uit de volgende specifieke kleuren :

- (1)

Pantone : Process Black

CMYK : C 0, M 0, Y 0, K 100

RGB : R 0, G 0, B 0

- (2)

Pantone : 115

CMYK : C 0, M 8,5, Y 79, K 0

RGB : R 255, G 233, B 0

- (3)

Pantone : 032

CMYK : C 0, M 91, Y 87, K 0

RGB : R 246, G 0, B 0

- (4)

Pantone : 338

CMYK : C 47, M 0, Y 32, K 0

RGB : R 135, G 207, B 163

- (5)

Pantone : 343

CMYK : C 98, M 0, Y 72, K 61

RGB : R 2, G 56, B 37

- (6)

Wit

CMYK : C 0, M 0, Y 0, K 0

RGB : R 255, G 255, B 255

De kleurenversie geniet de voorkeur boven de monochrome versie; zij moet worden weergegeven op een witte achtergrond.

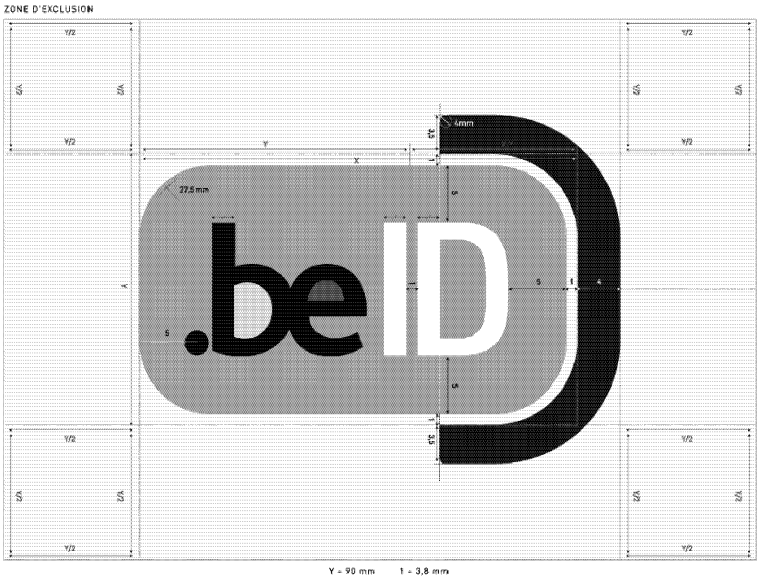
2.2. De monochrome versie, wanneer de kleurenversie niet mogelijk is, bestaat verplicht uit zwart en wit, met een raster à 35% van het zwart voor het grijze gedeelte.

Zij moet worden weergegeven op een witte achtergrond.

3. Het label moet, volgens bovenstaand model, als dusdanig worden weergegeven, zonder wijziging van vorm, kleuren, leesbaarheid en afmetingen.

4. Le logo figurant le label est entouré d'une zone d'exclusion, dans laquelle ne peut figurer aucun élément quelle qu'en soit la nature; sa reproduction respecte cette zone telle qu'ici définie :

4. Het logo dat het label vormt wordt omringd door een vrije ruimte, waarin geen enkel element van om het even welke aard mag voorkomen; de reproductie ervan neemt deze ruimte in acht zoals hier omschreven :



Si le logo, en couleurs ou monochrome, devait être reproduit sur un support d'une couleur autre que le blanc, il conviendrait alors que la zone d'exclusion soit blanche.

Il peut, par exemple, s'agir d'un autocollant où le logo est reproduit sur un fond blanc épousant la zone d'exclusion.

5. Le label est placé sur l'appareil de lecture considéré de manière visible et de préférence sur sa face avant.

Pour garantir sa visibilité, le label (logo hors zone d'exclusion) doit avoir une dimension proportionnée à l'appareil considéré, avec une valeur de Y (voir schéma) de minimum 10mm.

Le label ne peut pas dominer visuellement le logo ou la dénomination du fournisseur ou de l'appareil de lecture considéré.

6. Le label peut également être reproduit, aux même conditions que celles ici précisées, sur la publicité, l'emballage et la documentation de l'appareil de lecture considéré et ce pour autant que ces éléments présentent le label en le rattachant, directement et de manière explicite, audit appareil et à lui seul.

A cet égard, un fournisseur ne peut pas se prévaloir de l'obtention du label indépendamment de l'appareil auquel il a été attribué.

Lorsque l'appareil de lecture considéré est proposé à la vente sous un emballage, ou un conditionnement, qui ne permet pas de voir le label apposé sur ledit appareil, le label doit être reproduit sur l'emballage ou le conditionnement.

7. Le label et les éléments nécessaires à sa reproduction peuvent être obtenus par téléchargement électronique, à l'adresse www.fedict.be.

Vu pour être annexé à Notre arrête du 7 décembre 2006 fixant les specifications et la procedure d'enregistrement des appareils de lecture pour la carte d'identité éléctronique et modifiant l'arrêté royal du 13 février 1998 relatif aux specifications des appareils de lecture de la carte d'identité sociale.

ALBERT

Par le Roi :

Le Ministre de l'Intérieur,
P. DEWAEL

Le Ministre de l'Economie,
M. VERWILGHEN

Le Ministre des Affaires sociales,
R. DEMOTTE

La Ministre des Classes moyennes et de l'Agriculture,
Mme S. LARUELLE

Le Ministre de l'Emploi,
P. VANVELTHOVEN

Indien het logo, in kleurenversie of monochrome versie, op een andere achtergrond wordt weergegeven dan een witte achtergrond dan is het aangewezen een vrije ruimte in het wit te voorzien.

Het kan bijvoorbeeld gaan om een zelfklever waar het logo wordt weergegeven op een witte achtergrond, die precies de vorm aanneemt van de vrije ruimte.

5. Het label wordt duidelijk zichtbaar en bij voorkeur op de voorkant van het betrokken leesapparaat aangebracht.

Om de zichtbaarheid ervan te verzekeren moet het label (logo buiten vrije ruimte) een afmeting hebben die in verhouding is met het betrokken apparaat, met een waarde Y (zie schema) van minimum 10mm.

Het label mag visueel gezien het logo of de benaming van de leverancier of van het betrokken leesapparaat niet overheersen.

6. Het label mag ook, op dezelfde voorwaarden als hier beschreven, worden weergegeven op de reclame voor, de verpakking en de documentatie van het betrokken leesapparaat en dit voor zover deze elementen het label presenteren, rechtstreeks en uitdrukkelijk in verbinding met dit apparaat en enkel dit apparaat.

Een leverancier kan zich dus niet beroepen op het bekomen van het label, los van het toestel waaraan het werd toegekend.

Wanneer het betrokken leesapparaat wordt verkocht in een verpakking, die belet dat men het label op het apparaat kan zien, dan moet het label op de verpakking worden weergegeven.

7. Het label en de elementen voor de reproductie ervan kunnen bekomen worden door elektronisch teleladen op het adres www.fedict.be.

Gezien om te worden gevoegd bij Ons besluit van 7 december 2006 tot vaststelling van de specificaties en registratieprocedure van de leesapparatuur voor de elektronische identiteitskaart en tot wijziging van het koninklijk besluit van 13 februari 1998 houdende specificaties van de leesapparatuur voor de sociale identiteitskaart.

ALBERT

Van Koningswege :

De Minister van Binnenlandse Zaken,
P. DEWAEL

De Minister van Economie,
M. VERWILGHEN

De Minister van Sociale Zaken,
R. DEMOTTE

De Minister van Middenstand en Landbouw,
Mevr. S. LARUELLE

De Minister van Werk,
P. VANVELTHOVEN

Moniteur belge, rue de Louvain 40-42, 1000 Bruxelles. – *Belgisch Staatsblad*, Leuvenseweg 40-42, 1000 Brussel.
Conseiller/Adviseur : A. VAN DAMME