Consorci
Administració Oberta
de Catalunya

Manual d'ús de l'Applet d'e-signatura

Referència:	D1312
Versió:	2.6.6
Data:	26/10/2016

Informació general

Control documental

Projecte:	Applet d'e-signatura
Entitat de destinació:	AAPP Catalanes
Títol:	Manual d'ús de l'Applet d'e-signatura
Codi de referència:	D1312
Versió:	2.6.6
Data:	26/10/2016
Fitxer:	Manual d'ús de l'applet d'e-signatura v2.6.6.doc
Eina/es d'edició:	Word 2007
Autor/s:	Albert Ciffone
Resum:	Manual d'ús i exemples de l'applet d'e-signatura.
Classificació informació – nivell d'accés:	pública

Drets d'ús

Control de versions

Versió	Descripció del canvi	Data
1.0	Creació del document	21/11/2006
1.1	Afegits paràmetres connexió: Nou paràmetre per a la configuració d'un proxy, si no s'indica res, aquest paràmetre intenta autoconfigurar-se.	28/11/2006
1.2	Afegits paràmetres selecció de certificat i canvi en el disseny dels diàlegs: Nous paràmetres que permeten filtrar per CA emissora de certificat i seleccionar certificat per CN. S'ha modificat el nom del paràmetre sig_cert_alias a selected_alias.	15/12/2006
1.3	Correcció per a que aparegui el text del botó i darrera versió de les llibreries.	4/1/2007
1.4	Afegit nou mètode de selecció de documents a signar utilitzant una llista amb els paths corresponents separats per punts i comes (;). Correcció de versions als exemples.	18/1/2007
1.5	Corregit el comportament per a la sortida amb event Javascript o per camp de formulari. La sortida per als documents que no siguin XML es codifica a Base64 (sempre que no ho estigui prèviament) per evitar comportaments estranys del navegador amb continguts binaris.	29/1/2007
1.6	Afegides noves funcionalitats i correcció de diversos bugs detectats: Es permet l'ús de múltiples resums criptogràfics (hash) separats per ";". També és possible indicar múltiples url's (absolutes) per a que l'applet en descarregui els documents i els signi posteriorment. Corregits alguns bugs de l'apartat gràfic i en l'exportació de signatures a fitxer. Nou event Javascript que permet que es capturin de cop totes les signatures generades.	30/3/2007
1.7	Refet el disseny gràfic de l'aplicació: És possible escollir el color de fons de l'applet i el logo principal, que permetrà que s'adapti millor a les aplicacions, és configurable per paràmetre. El proveïdor SunMSCapi donava problemes de compatibilitat en l'accés a claus del clauer idCAT a través del magatzem de Windows i s'ha substituït per un altre de codi obert que sí que ho permet, mantenint també el que funcionava fins ara. Afegida compatibilitat amb el clauer idCAT, mitjançant el canvi del proveïdor de Microsoft CryptoAPI.	23/5/2007

	Afegits nous formats de signatura: CAdES-BES attached i detached i CAdES-BES en un PDF. Correccions aplicades als problemes en la signatura de formularis XML (codificació UTF-8) i en la descàrrega de documents a signar.	
1.8	Afegit nou paràmetre XML per a poder seleccionar si l'algoritme de canonicalització ha de tenir en compte o no els comentaris. Possibilitat de crear signatures CMS/CAdES a partir del resum criptogràfic del document. Nou paràmetre que permet la selecció d'idioma: català / castellà. Nou filtre de certificats. Utilitzant una cadena de text present en el SubjectDN.	12/2/2008
1.8.1	Afegir instruccions per l'ús de l'applet en mode servidor.	22/5/2008
1.9	Afegir possibilitat de certificació en la signatura de documents PDF. Afegir la possibilitat d'escollir la imatge que apareix a la representació de la signatura en un document PDF. Afegir filtre per identificador de política de certificat. Fer la URL de la TSA parametritzable.	04/02/2009
1.9.1	Afegir un nou paràmetre per configurar l'espai de memòria a reservar per la signatura, quan aquesta estarà incrustada en un document PDF. Afegir nou paràmetre per escollir si es mostra o no el missatge amb el pop-up de sortida quan es grava la sortida en fitxer.	07/05/2009
1.9.2	Actualitzar les crides a l'applet en mode servidor per documents PDF.	03/08/2009
1.9.3	Compatibilitat de l'applet amb el magatzem de certificats de Firefox, per Windows i Linux.	21/09/2009
1.9.4	Nou paràmetre que afegeix la possibilitat de validació del certificat de signatura contra PSIS.	30/03/2010
1.9.5	Nou paràmetre que permet el filtre per NIF d'usuari (requereix de validació amb PSIS).	28/04/2010
1.9.5b	Afegit exemple6 – problemes seguretat.	02/09/2010

2.0.0	Implementat l'accés al keystore de MAC OS X (actualment funciona amb SAFARI). Implementat suport per al keystore Java. Afegit nou valor pel paràmetre keystore_type, de manera que l'applet seleccioni el magatzem de claus de forma automàtica. Actualitzades les llibreries de BouncyCastle a la versió 1.45. Nous paràmetres per afegir el rol del signatari i el compromís de signatura a les signatures XML i CMS. Afegit un diàleg de java a l'applet per a evitar problemes de "*fakepath*" deguts a les configuracions de seguretat que implementen els navegadors actuals. Afegida la possibilitat d'escollir una carpeta que contingui tots els documents a signar a partir del nou diàleg per a escollir fitxers. Afegit paràmetre per a evitar que l'applet retorni totes les signatures i provoqui un error de javascript en mode multiSignature quan el volum de les dades signades supera el límit permès per javascript. Corregit el nivell de XAdES quan es canvia el valor del paràmetre signature_mode de forma dinàmica. Corregit bug que no permetia crear XAdES si s'especificava policy_id i policy_hash sense especificar policy_qualifiers. Corregit bug en les signatures XAdES enveloped. En cas de no afegir l'atribut Id al SignedInfo, no era possible l'actualització posterior de la signatura a AdES-A, mitjançant PSIS. Afegits nous exemples per a les noves funcionalitats. Revisades les compatibilitats dels exemples amb diversos navegadors.	10/12/2010
2.1	Afegida la comprovació sobre el resum criptogràfic (hash). Es comprova que la longitud es correspongui amb l'algorisme especificat. Si no es correspon, es mostra una alerta informant del problema. Firefox keystore + clauer idCAT + TCAT + DNIe: S'ha modificat el codi de l'applet de manera que en utilitzar com a keystore el de Firefox, l'applet també detecti el clauer idCAT, la T-CAT o el DNIe (s'ha d'haver instal·lat prèviament el programari d'aquests). Addicionalment s'ha afegit també el	03/05/2011

	paràmetre pkcs11_addicionals_firefox. Aquest paràmetre permet afegir altres dispositius externs, a part dels ja citats, quan s'utilitza el keystore de Firefox. Els valors del paràmetre s'han d'especificar de la forma "driver_path,identificador;driver_path,identificador;...". L'identificador és important per a que internament l'applet pugui diferenciar els pkcs11. A més aquest identificador es mostrarà en el label del popup que demana el pin amb el text: "introduïu el pin per a <identificador>:". S'han afegit les dlls de Firefox, per a que també funcioni el Firefox 64 bits en Windows. S'ha canviat el provider de be.cardon pel de SunMSCapi per a Windows de 64 bits, per a poder accedir al magatzem de claus de Windows en instal·lacions de 64 bits. Corregit bug en la generació de signatures XAdES detached sobre n documents: En generar una XAdES detached sobre varis documents, l'atribut ID de les referències (*ds:Reference*) s'instanciava de forma incorrecta.	
2.2	Afegida la possibilitat de generar segells de temps de tipus EncapsulatedTimestamp per a signatures de tipus XAdES (nou paràmetre includeXMLTimestamp, de tipus boolean: true: XMLTimestamp, false: EncapsulatedTimestamp) Afegida la possibilitat de generar Signatures XAdES enveloped amb referències a un o varis nodes. Nou paràmetre uris_to_be_signed. Varies uris separades per ';' que apunten als nodes del document a signar. Si el paràmetre no està instanciat es signa tot el document. Corregit bug: Fins ara, si el magatzem de Windows disposava d'una clau pública sense clau privada, aquesta es mostrava com a seleccionable a l'hora de signar i provocava un error. S'ha afegit codi per a l'eliminació de claus públiques que no contenen clau privada del magatzem de Windows. Protecció contra certificats amb Subject o Issuer sense CN o OU. Evitar que no es carregui la pantalla de selecció en cas que el certificat no té l'emissor o l'assumpte no té CN. Afegit el suport pel magatzem de Mozilla Firefox, en cas que aquest estigui protegit amb contrasenya mestra. Afegits exemples 22 i 23 per a l'ús dels nous paràmetres (includeXMLTimestamp i uris_to_be_signed).	18/08/2011

2.2.1	Corregit problema amb Linux i Firefox 7: El problema era que l'applet no detectava el navegador, perquè el Firefox ja no es registra al classpath. Eliminada classe FileListingUtils perquè no s'utilitzava. Afegit missatge d'error adient en cas de que l'usuari introdueixi el PIN malament quan el keystore és un PKCS12, un JKS o un PKCS11. Corregit bug: Quan el servidor de segell de temps XML responia amb un error, l'applet es quedava indefinidament fent la cerca per XPath sobre la resposta. Modificada la generació de segells de temps binaris, per tal de que es puguin generar correctament segells indicant qualsevol TSP. Generada una nova versió de totes les llibreries de l'applet (applet, CMS, PDF, XML) per a afegir un atribut al MANIFEST (http://download.oracle.com/javase/6/docs/technotes/guides/jweb/mixed_code.html).	10/11/2011
2.2.2	Afegit nou paràmetre "embedded" que permet incorporar el diàleg de selecció de certificats dins de la plana html en comptes de en un popup java. Afegida les funcions javascript de retorn onReturnCerts que retorna els certificats del magatzem per a pintar-los a l'html. I la crida javascript signFromJS(alias), que especifica l'alias seleccionat per l'usuari per a poder acabar el procés de signatura. Corregit bug: Amb keystores de tipus JKS només es podien generar signatures de tipus XML. S'ha modificat la classe de BouncyCastle CMSSignedHelper per a solucionar el problema. Això suposa la generació d'una nova versió de la llibreria CMS de l'applet. Minor: S'ha modificat el comportament de retorn de la funció onMultiSignOk, de manera que ara a priori no faci falta saber el número de signatures a retornar per la funció. Retornarà un array amb totes les signatures que s'haurà de recorre en javascript. Afegit exemple24: S'afegeix un loadPanel en js que desapareix un cop carregat l'applet. Es proposa com a solució per a connexions lentes que generen esperes a l'usuari durant la carrega. Afegit exemple 25: Exemple que mostra com incorporar el	25/01/2012

	diàleg de selecció de certificats dins del propi html.	
2.3	Actualitzades les llibreries d'iText de la versió 2.1.8 a la versió 5.1.3. Això és reflecteix en el jar CATCertPDFlib1.1.1.jar que passarà a ser el CATCertPDFlib1.2.jar Afegit suport per a Windows 2003 i Windows 2008 en els casos que s'utilitzi el paràmetre keystore_type = 0, per a la resta de casos ja funcionava. Afegit control d'error en casos de sistemes operatius no suportats per a keystore_type = 0, es donarà un missatge més descriptiu a l'usuari de manera que podrà demanar-nos suport per aquell SO en cas que ho consideri oportú.	13/03/2012
2.4	S'ha generat una nova versió de la llibreria PDF i del core de l'aplicació. S'ha generat una nova versió de les llibreries anteriors i de la resta (CMS i XML) compilades sense la informació de debug per a disminuir la mida dels diferents jars (s'ha disminuït la mida total de totes les llibreries de 7MB a 5,8MB). Les versions dels nous jars són: appletCATCert2.4.jar, CATCertCMSlib1.3.1.jar, CATCertPDFlib1.3.jar, CATCertXMLlib1.2.2.jar. PDF: Opció de signatura visible en totes les planes. S'ha de passar com a paràmetre el número de plana -1. PDF: Afegir les traduccions per als tags: date, reason, location i digitally signed que apareixen a la caixa de signatura, per a que surtin en català o castellà en funció del idioma seleccionat en la configuració de l'applet. Major: Afegida la funcionalitat d'exportar el certificat en base64 del magatzem seleccionat. Es fa amb ua nova crida javascript. Aquesta crida està disponible tant amb el mode normal com amb el mode embedded. Bug fix: En algunes instal·lacions de linux al carregar els drivers de la TCAT amb el dnie connectat al PC fa q l'execució de l'applet s'aturi per un bloqueig del dispositiu. Minor: Mostra un popup amb un error, en cas que s'hagi	08/05/2012

	introduït un password incorrecte per a un pkcs11 o per al magatzem de firefox. Minor: Modificats els logos de l'eina web de signatura-e. Minor Intern: Eliminar les traces amb les excepcions que mostra en cas de carregar un pkcs11 que no està connectat al PC perquè pot donar a l'usuari la sensació de que es tracta d'un error Minor intern: Optimitzada la codificació del retorn js_event per a millorar els temps de resposta. S'afegeixen a la versió distribuïble els següents exemples: Exemple 26 signatura d'un PDF visible en totes les planes. Exemple 27 exportació d'un certificat Exemple 28 exportació d'un certificat en mode embedded	
2.5	Afegides les llibreries (SO) i el codi necessari per a poder carregar el magatzem de firefox en instal·lacions linux de 64 bits. Testejat amb ubuntu 12.04 64 bits amb la versió de java 1.6.0_35 i firefox 12 i amb ubuntu 11.04 firefox 7.0.1 i java 1.6.0_22 (Si la primera execució no funciona recordeu d'eliminar les llibreries .so que es puguin trobar a /home/<user>/CATCert/ ja que poden ser les llibreries de la versió de 32 bits de l'execució d'alguna versió anterior de l'applet que no suportava linux 64 bits) Fer que el seteig dels paràmetres del PDF pugui ser dinàmic, de forma que quan es signi p.e es pugui afegir una imatge a la signatura en funció del certificat amb el que es realitzala signatura (veure exemple29.html) Nou paràmetre pdf_dinamic_signature_image i nova funció de retorn pdfDinamicSignatureImage, la funció de retorn rebrà tots els atributs de validació del certificat del signant contra PSIS de manera que amb aquests es pugui fer el codi de la tria de la imatge. Nou paràmetre abort_pdf_sign_operation, aquest només es pot setejar via el set method, es fa així pq només te sentit setejar-lo dinamicament en funció de les operacions fetes a la funció de callback javascript (pdfDinamicSignatureImage) que seteja la imatge de forma dinàmica. Afegit el paràmetre sign_first_sign_fields, el fet de setejar aquest paràmetre amb valor true fa que l'applet a l'hora de	25/09/2012

	signar només permeti seleccionar els camps de signatura en cas que aquests siguin presents al pdf. En cas que hi hagi més d'un camp de signatura el deixarà triar, però a diferencia del funcionament normal no permetrà crear-ne un nou. Si només queda un camp de signatura disponible el triarà automàticament. En cas que no quedi lliure o no hi hagi cap camp de signatura el funcionament serà el normal i afegirà una nova signatura amb la parametrització que tingui l'applet en aquell moment (signatura visible o no, imatge, requadre, pàgina etc...). (Veure exemple30.html) Intern: Afegit sanity check, si s'activa el paràmetre sign_first_sign_fields es comprova q no s'hagi configurat signatura visible a totes les planes. En cas que s'hagi fet apareix un alert indicant que l'applet no està configurat correctament. Intern: El paràmetre pkcs11_addicionals_firefox només es seteja al readParameters i no al set. S'afegeixen a la versió distribuïble els següents exemples: Exemple 29 Exemple per al pdfDinamicSignatureImage Exemple 30 Exemple de l'atribut sign_first_sign_fields	
2.5.1	Afegit suport per a que amb keystore_type = 3 és puguin afegir diferents pkcs11's separats per ; al paràmetre pkcs11_files. Aquest paràmetre substitueix l'àntic pkcs11_file ja que permet posar un o varis pkcs11s. També substitueix el pkcs11_addicionals_firefox, que ara en cas d'anar contra el magatzem de firefox (ja sigui amb keystore_type = 0 (autodetectant el firefox) com amb keystore_type = 4 (forçant firefox)) tindrà el mateix funcionament utilitzant el pkcs11_files.	15/10/2012
2.5.2	Corregit bug intern: Problema amb l'accés al magatzem de windows a través del sunMCSAPI per a versions de java anteriors a la java 1.5 update 18.	18/11/2012
2.5.3	Nova funcionalitat per a poder extreure el certificat amb el que s'ha realitzat la signatura (veure exemple 31) En cas d'instal·lacions on hi pugui haver més d'un perfil de firefox, carrega cadascun dels magatzems associats als perfils. Àlies duplicats al magatzem de windows: El sunmscapi de java té problemes amb els àlies duplicats al magatzem de windows (http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6672015) s'ha afegit un workaround per a corregir el problema Bug conegut de java http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6506	28/01/2013

	467 reportat per l'ajuntament de Terrassa. Problema al carregar un document factory del XML des de una plana web on la plana d'error no retorna un 404. S'ha afegit un workaround per a evitar aquests problemes.	
2.5.4	Exemple 32 – java_arguments per a arrencar la JVM amb la parametrització desitjada. Exemple 33 – Comunicació java → javascript amb signatures superiors als 4MB en cas de sortida javascript (js_event=true) Suport per a windows 8. Paràmetre pkcs11_files afegida la possibilitat d'especificar elements mútuament excloents, p.e varies versions d'una mateixa llibreria. Comunicació java → javascript, implementat mode via stream per a poder retornar signatures de més de 4MB a través dels events javascript Modificada la carrega del proveïdor criptogràfic de windows (sunmsc) per a carregar la dll només en les versions de java que no la porten incorporada (versións de java <= 1.5) Reportat per aj. Terrassa. Resignats tots els jars amb el nou certificat CDP.	05/04/2013
2.5.5	Resolt problema alies duplicats al magatzem de windows per culpa del SUNMSCAPI, workaround que per a alies duplicats crea com a alies el hash del propi cert. PKCS11_files que funcioni correctament amb tots els magatzems quan el keystore_type = 0. Afegits atributs al manifest per els nous requeriments de seguretat de la màquina virtual de java. De cara a les properes release (superiors 1.7.0.45) de java els jars que no continguin aquests atributs no s'executaran Es distribueix una nova versió de tots els jars.	18/10/2013
2.6	http://192.168.166.69:3000/issues/918 L'applet actualment quan se li indica que signi documents d'una carpeta donada carrega els bytes[] d'aquest documents en una Vector<byte[]> evidentment quan la llista i la mida dels documents creix molt facilment aquesta operació pot acabar donant un problema de HEAP.	18/02/2014

	S'ha de modificar aquest comportament per a que no carregui tots els fitxers prèviament en memòria i els vagi signant un per un. http://192.168.166.69:3000/issues/1003 L'applet no genera XAdES de tipus C, però accepta un valor per a fer-ho al signature mode, el codi internament acaba realitzant una XAdES-T amb dos segells de temps. http://192.168.166.69:3000/issues/485 A banda de les crides javascript openFileDialog() i openFolderDialog() que s'exposaven per a ser invocades des de javascript per a poder seleccionar un fitxer o una carpeta amb el document a signar. S'han creat els mètodes openFileDialog(String defaultFolder), openFolderDialog(String defaultFolder) que permeten indicar el path des d'on es vol que s'obri el dialeg de selecció de fitxers. El openFileDialog() i el openFolderDialog() s'obren per defecte sobre el user home. http://192.168.166.69:3000/issues/801 S'ha afegit el paràmetre js_event_only. Aquest paràmetre és un booleà, per defecte el seu valor és false. Amb aquest paràmetre activat (true) a l'hora de retornar l'event javascript amb la signatura (onSignOk()), realitza la crida sense passar la signatura. Només té sentit instanciar aquest valor en cas que el valor js_event sigui true. http://192.168.166.69:3000/issues/836 S'ha modificat l'applet de signatura per tal d'estructurar les crides a la generació de signatures de manera que la majoria de paràmetres es setegin per defecte i evitar així crides a baix nivell per part dels clients. S'ha fet també un projecte java d'exemple per a utilitzar l'applet en mode servidor. En aquest projecte és mostra com generar de forma senzilla, una signatura de tipus XAdES_BES, una CAdES_T i un PDF amb una CAdES-BES incrustada i a banda una validació de certificat contra PSIS. A banda de forma interna, també s'ha fet una mica de neteja de codi, i s'han eliminat totes les crides que antigament s'utilitzaven per a la invocació de signatures en servidor. http://192.168.166.69:3000/issues/904 El nou paràmetre és pdf_show_adobe_sign_validation per defecte pren el valor false. Si s'indica el valor true, a la signatura es mostrarà el tick, cross o el interrogant i la descripció de l'estat (signature valid, signature invalid, signature not yet verified) per defecte que fa l'adobe a banda de l'imatge i el text que hi hagi configurat(pdf_signature_image, reason, location etc...), si no es seteja a true aquesta info extra de l'adobe no es mostrarà. http://192.168.166.69:3000/issues/1106	

	En cas de detectar un FF en l'execució amb MAC OS X s'intentarà carregar els certificats des de el perfil de l'usuari en cas que es trobi en alguna d'aquestes dues ubicacions: ~/Library/Application Support/Firefox/Profiles/<profile folder> ~/Library/Mozilla/Firefox/Profiles/<profile folder> Aquesta funcionalitat està en fase beta.	
2.6.1	S'han actualitzat per a aquesta versió 3 dels 4 jars: appletCATCert2.6.1.jar, CATCertPDFlib1.3.2.jar, CATCertXMLlib1.2.4.jar. S'ha actualitzat també el projecte d'exemple de signatura en servidor. S'han revisat i unificat tots els exemples. http://192.168.166.69:3000/issues/1911 Ultima actualització de chrome provoca que el navegador quedi bloquejat quan en el callback produït per un applet hi ha un alert. Només passa amb chrome, la resta de navegadors segueix funcionant correctament. S'han actualitzat els exemples amb el següent workaround: a través d'un setTimeout s'afegeix un petit delay a l'aparició del alert que permet que tot segueixi funcionant correctament → Chrome a eliminat el NPAPI en el que estan basats els plugins d'execució java, amb el que el canvi no aplica. http://192.168.166.69:3000/issues/2073 Canvis interns de millora de codi, principalment: JacksumAPI + PDFHash: Corregit el comportament el hash del document a signar estava hardcodejat a "sha-1" Proxy: Fet la connexió a través de proxy més robusta davant fallides. PKCS12KeyStore: Nous constructors per a poder generar instàncies des de servidor de forma més senzilla Carpeta de 32 bits i 64 bits (/32 or /64) diferents dins del $USER_HOME/CATCert per a que en una mateixa màquina es pugui executar l'applet amb jvm de 64 bits i de 32 bits indistintament Altres canvis interns. http://192.168.166.69:3000/issues/1887 Modificat el config de XMLSecurity degut ha que hi ha classes que no existeixen i en depèn quins servidors es desplegui l'applet a l'anar a buscar aquestes classes pot acabar donant un securityException si el servidor retorna un 404. http://192.168.166.69:3000/issues/1729 Afinat el missatge d'error de no hi ha certificats en funció de si s'aplica un filtre o no:	16/09/2015

	Si simplement el magatzem no disposa de certificats es mostrarà el següent missatge: ca=No hi ha certificats disponibles per a signar. es=No hay certificados disponibles para realizar la firma. Si s'està aplicant un filtre per als certificats (allowed_oid, subject_text..), en cas que no n'hi hagin es mostrarà el següent missatge: ca=No hay certificados disponibles para realizar la firma con los filtros aplicados. es=No hi ha certificats disponibles per a signar amb els filtres aplicats. http://192.168.166.69:3000/issues/1439 S'ha afegit suport per a windows server 2012. http://192.168.166.69:3000/issues/1228 Nou exemple en servidor de generar un signatura xml detached a partir d'un hash. http://192.168.166.69:3000/issues/1153 S'ha modificat el funcionament de l'applet per a afegir la possibilitat de rotar les signatures PDF (tant la imatge com la descripció). S'ha afegit el següent paràmetre a la configuració: pdf_signature_rotation. Els possibles valors són 90,180,270. El gir es fa en sentit anti-horari el nombre de graus especificats per paràmetre. https://redmine.aoc.cat/issues/3783 Suport per a windows 10.	
2.6.2	https://redmine.aoc.cat/issues/4335 Suport Firefox 64 bits en MAC OS X (versió provada MAC OS X 10.8.5, Firefox 43, JVM 1.8.0_71-b15 https://redmine.aoc.cat/issues/4156 prjAppletSignaturaServidor nou exemple signar CAdES_BES_detachedHash https://redmine.aoc.cat/issues/4303 Problema keystore_type=3 amb pkcs11_files on hi ha espais en blanc a la ruta del driver del pkcs11 https://redmine.aoc.cat/issues/3250 Solucionat outOfMemory signant molts documents des de diferents urls (doc_type = 6). https://redmine.aoc.cat/issues/4320 Afegida ruta de carrega per defecte TCAT en MAC OS X quan el magatzem és el de Firefox (/usr/local/lib/libaetpkss.dylib)	

2.6.3	https://redmine.aoc.cat/issues/4767 Arrel l'actualització de java 8 update 77hi ha bloquejos en alguns navegadors en l'execució del applet per culpa del thread de la GUI. Els panells i el navegador queden unresponsive.	
2.6.4	https://redmine.aoc.cat/issues/5243 Problemes al realitzar multiples signatures tipus CMS/PDF amb PKCS11 en java 1.8. Queda resolt modificant les llibreries de BC. Només s'ha generat nou el jar CATCertCMSlib1.3.3.jar	
2.6.5	https://redmine.aoc.cat/issues/5619 Adaptació degut als canvis interns a les modificacions del SunMSCAPI per a la versió de java 1.8 update 101. https://redmine.aoc.cat/issues/5621 Afegits paths alternatius als drivers del PKCS11 de la TCAT. Generat nou jar core AppletCATCert2.6.5.jar	
2.6.6	https://redmine.aoc.cat/issues/5701 CAdES afegit suport per signingCertificateV2 per a poder fer referència al certificat de signatura en cas que la signatura sigui en SHA2 amb RSA.	

Glossari

<APPLET> Component software que corre en el context d'un altre programa, habitualment un navegador Web.

<ASN.1> **A**bstract **S**yntax **N**otation number One. Estàndard internacional amb l'objectiu de representar dades utilitzades en protocols de comunicacions.

<CMS> **C**ryptographic **M**essage Syntax **S**tandard. Estàndard de signatura electrònica basat en el llenguatge de representació ASN.1.

<DSS> **D**igital **S**ignature **S**ervices

<OASIS> **A**dvancing **o**pen **s**tandards for the **i**nformation **s**ociety

<PDF> ***P**ortable **D**ocument **F**ormat*, Format de Document Portàtil. És una forma d'emmagatzematge de documents, desenvolupat per l'empresa Adobe.

<PSIS> **P**lataforma d'**I**dentificació **i** **S**ignatura. Plataforma de serveis de CATCert per a validació i generació de signatures, arxiu i xifrat.

<TSA> **T**ime**s**tamping **A**uthority

<XAdES> **X**ML **Ad**vanced **E**lectronic **S**ignature. Format XML de signatura digital avançada perdurable en el temps. Pot presentar-se utilitzant formes com BES (Basic Electronic Signature), T (BES + Timestamp), C (T + Revocation Information).

<XAdES-EPES> **XAdES** **E**xplicit **P**olicy based **E**lectronic **S**ignature. Signatura XAdES que segueix una política predefinida de signatura (es pot validar contra aquesta política).

<XMLDSig> **XML** **D**igital **Sig**nature. Format XML de signatura digital.

Índex

1. Introducció

L'applet d'e-signatura és un complement per a les aplicacions web que requereixen de l'ús de la signatura electrònica. El seu objectiu és el de minimitzar l'impacte de la integració de la signatura electrònica en aquest tipus d'aplicacions, permetent de forma dinàmica i senzilla generar signatures en diferents formats i presentacions. Habitualment la signatura es generarà en tres passos: selecció del document a signar, selecció de certificat i finalment introducció del PIN per a generar la signatura. Aquests passos poden variar en funció dels paràmetres utilitzats.

En aquest document es donaran les pautes de com utilitzar l'eina, els paràmetres que permeten configurar-la per a diferents entorns i objectius, i exemples HTML per als diferents casos d'ús.

A mode de resum, per a signar es poden utilitzar certificats en software, targetes criptogràfiques accessibles a través del middleware (llibreria PKCS#11) i certificats emmagatzemats al magatzem personal de Windows o de Mac OS X. El tipus de signatures que es poden generar poden ser XMLDSig (també en la seva forma avançada, XAdES), CMS i CMS incrustades en un document PDF (PDF signat).

En totes els tipus de signatura suportats existeix la possibilitat d'incrustar-hi un segell de temps, XMLTimestamp a les signatures XML, RFC3161 Timestamp a les CMS. El servei de segellat de temps que s'utilitzarà per defecte és el que proporciona CATCert dins de la plataforma PSIS (http://psis.catcert.net/psis/catcert/tsp).

Cal destacar que l'ús de l'eina web, si està signada, permet realitzar totes aquestes operacions sense necessitat de tenir accés al disc local. L'únic cas en que això succeeix de forma no controlada per l'usuari final, és en el cas d'utilitzar el repositori de certificats de Windows: es guarden a disc, normalment a la carpeta temporal *C:\Documents and Settings\username\Configuración local\Temp*, les dues dll's necessàries.

2. L'eina.

L'eina està basada en la tecnologia Java, concretament és un applet compilat per a la versió 1.5 (o posteriors) de la màquina virtual de Java. El fet de que la implementació sigui en aquesta tecnologia obre el ventall de possibilitats de sistemes operatius i navegadors que poden suportar el seu ús.

L'applet consta de 4 paquets (jar) que es descarreguen dinàmicament en funció de la configuració seleccionada. Aquests quatre paquets són (x.x fa referència a la versió):

- ***appletCATCertx.x.jar*** – Conté un paquet amb la lògica de l'applet i les diferents implementacions de signatura i magatzems de certificats; un paquet de llibreries d'Apache Commons per a connexions http, necessàries per a poder generar segells de temps (RFC3161 o XML) i el SunMSCAPI per a la integració amb el magatzem de certificats de Windows. Suporta comunicació bidireccional, utilitzant Javascript, amb la pàgina HTML que l'invoqui. Des de l'HTML es poden modificar paràmetres de forma dinàmica i per la seva banda, l'applet envia informació crítica (errors, resultat de la signatura, etc) utilitzant events Javascript.
    - *Comunicació Javascript → applet*
        - Mètode *set(name,value)* – Actualitza el paràmetre amb nom *name* al valor *value*.
        - Mètode *signFromJS()* – Inicia el procés de generació de signatura amb la configuració proporcionada pels paràmetres.
        - Mètode *openFileDialog()* – Obre el diàleg propi de l'applet per a triar el fitxer a signar, evitant els problemes de "*fakepath*" associats a la utilització de l'*input* de tipus *file* d'HTML. Realitza l'assignació automàtica del paràmetre document_to_sign i retorna una crida javascript a onFileUpload(path) amb el *path* triat, a mode d' informació. Si l'usuari cancel·la, l'operació retorna una crida javascript a la funció onFileCancel().
        - Mètode *openFileDialog(path)* – Igual que el mètode openFileDialog() però en comptes d'obrir el explorador per defecte al home de l'usuari l'obre al path indicat per paràmetre.
        - Mètode openFolderDialog() – Obre el diàleg propi de l'applet per a triar una carpeta que contingui varis documents a signar. Realitza l'assignació automàtica del paràmetre document_to_sign i retorna una crida a onFileUpload(path) amb el *path* triat, a mode d'informació. Si l'usuari cancel·la, l'operació retorna una crida javascript a la funció onFileCancel().
        - Mètode *openFolderDialog(path)* – Igual que el mètode openFolderDialog() però en comptes d'obrir el explorador per defecte a la carpeta home de l'usuari l'obre al path indicat per paràmetre.
        - Mètode signFromJS(alies) – Aquest mètode inicia el procés de generació de signatura amb la configuració proporcionada pels paràmetres. La signatura es genera amb la clau del magatzem lligada a l'àlies proporcionat per paràmetre. Aquest mètode s'utilitza quan es vol fer servir l'applet en mode embedded, i l'àlies és el seleccionat entre els que retorna l'applet amb la funció onReturnCerts(value).

- A més també existeixen les crides per a l'exportació de certificats, veure el punt Exportació de certificats.

- Comunicació applet → Javascript
  - Funció *onLoadError(msg)* – Captura el missatge *msg* que envia l'applet quan hi ha algun error en la càrrega.
  - Funció *onSignLoad()* – Captura l'event que envia l'applet quan s'ha carregat correctament.
  - Funció *onSignCancel()* – Captura l'event que envia l'applet quan detecta que l'usuari cancel·la el procés de signatura.
  - Funció *onSignError(msg)* – Captura el missatge d'error *msg* quan l'applet té problemes per a generar la signatura.
  - Funció *onSignOK(signature)* – Captura la signatura quan l'applet es configura per retornar la resposta utilitzant event Javascript.
  - Funció *onMultiSignOK(signatures)* – Captura les signatures generades per l'applet. Les retorna en un array que s'ha d'anar recorrent amb javascript per a obtenir les signatures.
  - Funció *onFileUpload(path)* – Captura el *path* seleccionat mitjançant el diàleg de l'applet.
  - Funció *onFileCancel()* – Captura la cancel·lació per part de l'usuari del diàleg de selecció de fitxer de l'applet.
  - Funció *onReturnCerts(alies)* – Captura l'event generat per l'applet quan retorna els àlies de les claus del magatzem seleccionat. Aquesta crida substitueix el onSignLoad() quan s'utilitza l'applet en mode embedded.
  - Funció *pdfDinamicSignatureImage(List)* – Captura l'event generat per l'applet a l'hora de signar un PDF amb el paràmetre pdf_dinamic_signature_image a true. Aquesta crida retorna com a paràmetres els parells de valors: nom atribut, valor de la validació contra PSIS del certificat utilitzat per a signar. Dins d'aquesta crida en funció d'aquests valors es pot setejar dinàmicament els paràmetres de signatura visible del PDF. A banda es pot avortar el procés en funció dels valors retornats fent una crida a set(abort_pdf_sign_operation, true).
  - Funció *jsEventAppender(data)* – En cas de que l'applet estigui configurat per rebre l'event javascript i la signatura resultant superi els 4MB, en comptes de fer el retorn a través de la funció *onSignOk* l'applet anirà realitzant crides successives a *jsEventAppender* amb parts de la signatura que es podran anar concatenant per exemple en un element html. Quan s'hagi passat tota la signatura, la funció rebrà com a paràmetre data el text "JS_EVENT_EXCEDS_SIZE_LIMIT" per saber que ja disposa de la signatura completa (veure apartat 6.1.9 i el Exemple33.html que es distribueix amb la documentació).

- ***CATCertCMSlibx.x.jar*** – Es tracta d'un paquet reduït de la llibreria de Bouncy-Castle per a la generació de signatures CMS. En la versió 2.0.0 de l'applet s'ha actualitzat aquesta llibreria a la versió 1.45 de BouncyCastle.

- ***CATCertXMLlibx.x.jar*** – Conté el paquet d'Apache XML. Un paquet reduït amb petites modificacions de les llibreries d'Apache Security per a que sigui possible la generació de signatures XML detached.
- ***CATCertPDFlibx.x.jar*** – Conté un paquet reduït de la llibreria iText per al tractament de documents PDF. Permetrà incrustar signatures CMS en els documents, i que aquestes siguin vàlides (i visibles) si el document s'obre amb l'eina Acrobat Reader d'Adobe.

Mode servidor

L'eina també és pot utilitzar en mode servidor per a realitzar signatures, amb l'entregable es distribueix el codi font d'un projecte Java amb exemples d'utilització de la llibreria per tal d'il·lustrar-ne l'ús.

3. Compatibilitat

Per diferents motius, principalment relacionats amb vulnerabilitats de seguretat, la tecnologia que feia de connector entre la JVM i els navegadors per a l'execució dels applets va quedant poc a poc fora d'ús. Una prova n'és per exemple que els nous dispositius com poden ser els mòbils o les tauletes no incorporen aquesta tecnologia i els applets no es poden executar en aquests entorns. Aquest problema afecta de forma transversal tots els applets no només el de signatura, el problema principal de l'applet però és la migració a altres tecnologies que tinguin una més fàcil integració amb els navegadors com podria ser el javascript, el tema és però que actualment el javascript no permet l'accés als certificats dels magatzems dels navegadors, PKCS11, etc...

En la línea que comentàvem per exemple el navegador Chrome a eliminat el suport als plugins i l'applet de signatura ja no es pot executar sobre el mateix. No és l'únic donat que el nou navegador de Microsoft l'Edge tampoc en permet l'execució. De moment els propis autors de Java (Oracle) no ofereixen alternativa i el seu consell és utilitzar Firefox o internet explorer.

A continuació és mostren les diferents taules de compatibilitat amb navegadors, sistemes operatius i versions de java:

3.1 Java 1.6 o superior

Després de varis anys compilant l'applet amb la versió 1.5 de java finalment com ja varem anunciar; l'anterior release 2.6.1 és l'última versió que donarà suport a la java 1.5 donat la política d'actualització de java i al bloqueig de versions antigues que fan els navegadors, a banda del fet de que Oracle va deixar de donar suport a la versió 1.5 ja fa més de 6 anys (EOL Octubre del 2009).

3.2 Sistemes operatius

3.2.1 Windows

XP i 2000: Donat que Microsoft ja no dona suport per aquestes versions i que Oracle tampoc dona suport per a java en aquests SO. No podem garantir l'ús de l'applet en les mateixes, en teoria ha de seguir funcionant però no donem suport.

	Windows Vista, 7, 8, 2003, 2008, 2012, 10
Magatzem personal de Windows	✓
PFX/PKCS#12	✓
Targeta (PKCS#11)	✓
Magatzem Mozilla*	✓
JKS (Keystore Java)	✓

Per a la compatibilitat, s'han realitzat proves a diferents sistemes operatius com Microsoft Windows Vista, 2003, 2008, windows 7, 8 i 10 (32 i 64 bits).

Els navegadors per als que donem suport sobre aquest sistema operatiu són firefox (versió 3.6 o superiors) i Internet explorer 8 o superiors.

3.2.2 Linux

Kernel 3.0 i posteriors: S'havia donat suport per a versions del kernel 2.4 i posteriors, al igual que en el cas de windows XP es possible que el funcionament segueixi sent correcte, però no donem suport.

	Linux (kernel 3.0+)
PFX/PKCS#12	✓
Targeta (PKCS#11)	✓
Magatzem Mozilla*	✓
JKS (Keystore Java)	✓

Les proves s'han realitzant amb versions d'Ubuntu 14.04 LTS (32 i 64 bits).

Els navegadors per als que donem suport per a linux són Firefox 3.6 o superiors.

3.2.3 MAC OS X

Per a MAC OS X només garantim el funcionament per a versions de MAC des de la 10.5.x fins a OS 10.8.5. En versions de 32 i 64 bits.

	Mac OS X (10.8.5)
PFX/PKCS#12	✓
Targeta (PKCS#11)	✓
Magatzem Mozilla*	✓
JKS (Keystore Java)	✓
Magatzem .Mac	✓

Els navegadors per als que donem suport per a MAC són Safari fins a la versió 6.0.3 i firefox fins la versió 43.

3.3 PKCS11

El gran problema, sempre parlant de la part més complexa, que és l'ús de les targetes criptogràfiques, seran les pròpies limitacions de cadascun dels diferents entorns. Per exemple, la possible manca de controladors per a certs lectors de targetes, incompatibilitat de la implementació PKCS#11 del middleware de la targeta criptogràfica (drivers), etc. Per a d'altres opcions, com el certificat software aquestes limitacions no existeixen.

Per altre banda, els sistemes de 64 bits basats en Windows disposen de distribucions de navegadors i de la JVM per a 32 i 64 bits. Actualment però les distribucions de la JVM de 64 bits tenen diverses mancances (llibreries i classes per als PKCS11 i per al proveïdor que accedeix al magatzem de Windows 64 bits). Aquestes mancances es supleixen incorporant a la distribució del applet les llibreries (dll) compilades per CATCert per a versions de 64 bits, així com les classes necessàries pel seu ús. A banda d'això pot ser que algun dels dispositius PKCS11 no disposin de les llibreries necessàries o aquestes no funcionin correctament quan s'executen com a 64 bits. A continuació és mostra una taula de les diferents execucions (32 i 64 bits) sobre un sistema de 64 bits.

El cas és que en mode keystore_type=0 (fora de windows + internet explorer) només podem garantir el correcte funcionament de la TCAT via PKCS11 en els SO i navegadors comentats anteriorment. Per a la resta de PKCS11 (DNIE, FNMT...) fora de windows i internet explorer no podem garantir el seu funcionament amb keystore_type=0, per a l'ús de les mateixes és necessari utilitzar el mode keystore_type=3 i especificar la ruta de les llibreries utilitzant els paràmetres de configuració del applet per al PKCS11 (veure l'apartat Parametres de configuració)

4. Paràmetres de configuració

Es disposa de multitud de paràmetres que permeten configurar l'eina segons les necessitats. Per tal de facilitar-ne l'ús, es llistaran agrupats segons funcionalitat, donant detalls del que cadascun implica i de com s'utilitza.

Cal destacar que tots aquests paràmetres es poden modificar dinàmicament, ja que existeix un mètode públic (*set(String name, String value)*) per actualitzar-los i que és accessible utilitzant codi Javascript en el propi codi HTML.

4.1 Paràmetres principals (obligatoris)

Els paràmetres principals s'encarreguen de definir el comportament bàsic de l'applet, és a dir, de quin magatzem es recuperaran les claus que s'utilitzaran en el procés de signatura i quina implementació de signatura s'utilitzarà.

- ***keystore_type*** – Indicarà el tipus de magatzem de certificats que s'utilitzarà en el procés de signatura. Cal utilitzar un codi numèric dels següents:

  0 – Selecció automàtica del keystore en funció del sistema operatiu i el navegador. La taula de compatibilitat és de la següent manera:

	Microsoft Windows (Vista, 7, 8, 2003, 2008,2012,10)	***Linux (kernel 3.0+)***	***Mac OS X (10.8.3)***
Internet Explorer	***Magatzem de claus de Windows***	-	-
Firefox	***Magatzem de claus de Firefox*****	***Magatzem de claus de Firefox*****	✓
Altres navegadors	***Magatzem de claus de Windows***	✘	***Magatzem de claus de Mac***

  ** Amb la versió 2.1 s'afegeix suport a dispositius externs PKCS11 conjuntament amb el magatzem de Firefox (T-CAT, idCAT, DNIe per defecte; i disposem d'un paràmetre per afegir-ne d'addicionals).

  *** A partir de la versió 2.5 suport per a Linux de 64 bits.

  1 – Magatzem de certificats del compte personal d'usuari de Windows. Per a que aquesta opció funcioni, l'applet crearà una carpeta CATCert dins de la carpeta personal de l'usuari i hi guardarà la llibreria dll nativa (sunmscapi.dll) que necessita per a poder-hi accedir.

  2 – Certificat en software (fitxer PFX, PKCS#12). Cal indicar el camí absolut al fitxer que conté el certificat (veure apartat 4.6.2).

  3 – Targeta criptogràfica, utilitzant el middleware (llibreria PKCS#11). Es poden configurar diverses llibreries a l'hora, cal però indicar el camí absolut de les mateixes (veure apartat 4.6.2 i apartat 7.1.3).

4 – Magatzem de certificats de Mozilla. I PKCS11 de la T-CAT, idCAT i DNIe (addicionalment és poden afegir més PKCS11 mitjançant el paràmetre ***pkcs11_files***). S'ha afegit suport per a poder accedir al magatzem de certificats de Mozilla en cas que aquest estigui protegit per contrasenya mestre, en cas que estigui protegit abans de mostrar els certificats per a signar apareixerà un popup demanant l'entrada del pin per a aquest magatzem. En cas de que un mateix usuari disposi de diversos perfils de firefox és carregaran tots els perfils.

5 – Magatzem de certificats personals de Java. Cal indicar el camí absolut al keystore (veure apartat 4.6.2).

6 – Magatzem de certificats personals de Mac OS X.

- ***signature_mode*** – Permet seleccionar el format de representació de la signatura electrònica que es generarà. Cal utilitzar un codi numèric dels següents:

  1 – CMS attached (signatura CMS conté el document original).

  2 – CMS detached (signatura CMS no conté el document original).

  3 – CMS detached, utilitzant hash precalculat.

  4 – PDF (CMS detached incrustada en un document PDF).

  5 – XMLDSig enveloped (document XML embolcalla la signatura).

  6 – XMLDSig enveloping (signatura XML embolcalla el document original).

  7 – XMLDSig detached (signatura XML no conté el document original)

  8 – XMLDSig detached, utilitzant hash precalculat.

  9, 10, 11 i 12 – XAdES-BES enveloped, enveloping, detached i detached amb hash precalculat. Protegeix el certificat del signant.

  13, 14, 15 i 16 – XAdES-T enveloped, enveloping, detached i detached amb hash precalculat. Afegeix un segell de temps a la forma BES.

  17, 18, 19 i 20 – XAdES-C enveloped, enveloping, detached i detached amb hash precalculat. Afegeix informació de revocació a la forma T. *Opció encara no disponible.*

  21, 22, 23 i 24 – CAdES-BES attached, detached, detached a partir del hash i detached en un PDF.

  25, 26, 27 i 28 – CAdES-T attached, detached, detached a partir del hash i detached en un PDF.

  29, 30, 31 i 32 – CAdES-C attached, detached, detached a partir del hash i detached en un PDF. *Opció encara no disponible.*

En el cas de hashs precalculats, s'ha afegit un control per a la comprovació de la validesa del hash en relació a la longitud corresponent en funció de l'algorisme de hash emprat pel seu càlcul. En cas que el hash sigui incorrecte o no es correspongui amb l'algorisme especificat, es mostra un missatge d'alerta, i s'interromp el procés de generació de la signatura.

4.2 Paràmetres visuals

Defineixen l'aspecte visual de l'applet:

- ***signButtonCaption*** – Indica el text que apareixerà en el botó (única part visible de l'applet a la pàgina web) i que inicia el procés de generació de signatura.
- ***appletBackground*** – Permet indicar el color de fons dins de l'aplicació. Consisteix en un codi de 3 valors RGB separats per punts i comes (;), com per exemple el blanc: 255;255;255 o el gris (252;252;252).
- ***appletLogo*** – Representació en Base64 de la imatge (jpeg, gif, tiff, png) que es desitja visualitzar com a logo principal, com per exemple la de l'ens o la de l'aplicació web en que s'utilitza l'applet. Veure l'annex **¡Error! No se encuentra el origen de la referencia.** i l'exemple associat. La mida recomanada per a la imatge és de 350x65 píxels. Veure l'exemple 13, descrit a l'apartat 7.2.12 per a més detalls.
- ***language*** – Indica l' idioma en que es representaran els textos de l'aplicació. L'idioma per defecte serà el català. Els idiomes suportats són:

  ca – Català

  es – Castellà
- ***embedded*** – Indica si l'applet funcionarà en el mode on la selecció de certificats estarà dins de la pròpia pàgina web o en el popup java. El valor per defecte és "false" que indica que el mode de funcionament serà en popup.

4.3 Paràmetres de xarxa

En cas d'haver de fer connexions a serveis externs, com és el cas d'utilitzar segells de temps, caldrà tenir en compte si l'usuari es troba darrere d'un proxy. La configuració per defecte detecta l'ús del proxy configurat en el navegador. Si la configuració per defecte no funciona o es desitja utilitzar un proxy diferent del que hi ha configurat, caldrà utilitzar el paràmetre següent. De moment tan sols es suporta el mecanisme d'autenticació bàsic, basat en usuari i paraula de pas.

- ***proxy_settings*** – Indica les dades del proxy que haurà d'utilitzar l'applet en el cas d'haver de fer accessos a serveis remots (servei de segellat de temps de PSIS). Cal indicar el nom del proxy i el port separats per un espai, a més de l'usuari i la paraula de pas si calen (*proxy_settings = "serverName serverPort username paraulaDePas"*).

4.4 Paràmetres de segellat de temps

Podem especificar l'adreça URL del servei de segellat de temps (TSA) desitjat.

- ***cmsts_tsa_url*** – Indica l'adreça URL del servei de segellat de temps RFC3161.
  El seu valor per defecte és el servei de segellat de temps segons el protocol RFC3161 de PSIS:
  http://psis.catcert.net/psis/catcert/tsp
- ***xmlts_tsa_url*** – Indica l'adreça URL del servei de segellat de temps XMLTimestamp.
  El seu valor per defecte és el servei de segellat de temps de PSIS per segells de temps de format XML segons l'estàndard definit per OASIS al protocol DSS:

http://psis.catcert.net/psis/catcert/dss

És molt important tenir en compte que, en cas de modificar el valor d'aquest paràmetre, cal garantir que el servei de segellat que estem seleccionant treballi segons el protocol corresponent. Les signatures CAdES es generen amb segell de temps segons el protocol RFC3161. Les XAdES, segons el protocol definit per OASIS a l'estàndard DSS.

Així doncs, si volem generar una signatura CAdES amb segell de temps d'una TSA determinada, haurem de fer servir el paràmetre cmsts_tsa_url. Si en canvi, lo que volem generar és una signatura XAdES, haurem d'especificar el valor del paràmetre xmlts_tsa_url.

En cas de treballar amb la TSA de PSIS, no cal especificar els valors d'aquests paràmetres.

4.5 Paràmetres de validació

És possible fer que el certificat del signatari es validi contra PSIS abans d'iniciar el procés de signatura.

- ***psis_validation*** – Per defecte el seu valor és "false". Per tant, per defecte el certificat del signatari no es valida contra PSIS prèviament a la generació de la signatura.

  En cas de que es desitgi que el certificat sigui validat contra PSIS abans de la generació de la signatura, caldrà posar aquest paràmetre a "true". Si el certificat és invàlid, s'obrirà una alerta amb el missatge corresponent informant de la no validesa del certificat. Si el certificat és vàlid, es procedirà amb el procés de generació de la signatura, i no es mostrarà cap missatge addicional al client.

- ***required_nif*** – En cas de que es desitgi la validació del certificat del signatari contra PSIS (*psis_validation=true*), aquest paràmetre permet afegir un filtre sobre el certificat amb què es farà al signatura. Si el signatari escull un certificat el NIF del qual no coincideix amb el retornat per PSIS en la resposta de validació, aleshores no es permetrà la generació de la signatura, i l'applet retornarà un missatge d'error indicant que el NIF no és l'esperat.

  Cal tenir en compte que per fer servir aquest paràmetre cal indicar que s'ha de fer la validació contra PSIS del certificat del signatari (*psis_validation=true*), doncs el NIF del certificat s'obté mitjançant PSIS.

4.6 Paràmetres d'entrada

4.6.1 Document

Aquests paràmetres facilitaran a l'applet què serà i com ha de tractar el document a signar.

- ***document_to_sign*** – Forma abstracta del document a signar en funció del que s'indiqui en el paràmetre *doc_type*. Es pot obviar sempre i quan s'actualitzi amb el mètode públic abans de fer la crida al mètode de generació de signatura.
- ***doc_type*** – Codi numèric que indicarà a l'eina com recuperar el document que cal signar. Cal utilitzar-ne un dels següents. Si no s'indica res, el valor per defecte és 2 (document únic).

1 *– Directori local.* Permet signar tots els documents que hi hagi en una carpeta accessible des del client on s'executa l'applet. Amb aquesta opció, cal indicar al paràmetre ***document_to_sign*** el camí absolut al directori.

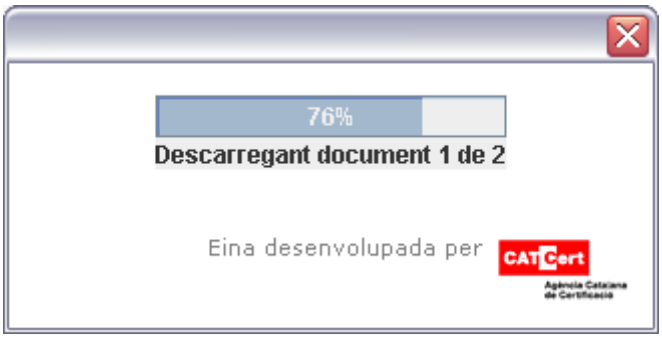
2 *– Document únic.* Cal indicar el camí absolut al document que es vol signar en el paràmetre ***document_to_sign***. Utilitzant el mètode públic per actualitzar paràmetres es pot crear un diàleg amb l'usuari per seleccionar el document a signar (veure annex 0).

3 *– Hash/s precalculat/s.* Amb aquesta opció, caldrà que el paràmetre ***document_to_sign*** contingui el valor del hash del document codificat en Base64. Es poden indicar múltiples cadenes de hash separades per punts i comes (;).

4 *– Contingut del document codificat en Base64.* Com ja indica, permet que en el paràmetre ***document_to_sign*** s'hi pugui carregar el document sencer. És una opció útil si no es pot precalcular el hash i no es disposa d'accés local al document.

5 *– Llista de camins absoluts als fitxers locals a signar*. Permetrà indicar una llista de documents locals a signar (no tenen perquè estar en la mateixa carpeta). El paràmetre ***document_to_sign*** contindrà la llista i haurà d'estar separada amb punts i comes (;).

6 *– URL/s del/s document/s a signar*. Permet indicar una o vàries urls absolutes on estan allotjats els documents a signar. En el paràmetre ***document_to_sign***, si s'indica més d'una URL cal separar-les per punts i comes (;). L'applet descarregarà el contingut dels documents i procedirà a generar-ne les signatures. Apareixerà una barra de progrés amb l'evolució de les descàrregues.



7 *– Signar un formulari*. Amb aquesta opció es pot passar com a document a signar una cadena de text generada a partir d'un formulari HTML. Es pot utilitzar qualsevol dels formats de signatura disponibles. Si, per exemple, aquesta cadena és un document XML i la signatura sol·licitada també, l'applet el parsejarà i l'inclourà com a XML en el cas de signatures XML enveloped o enveloping. Veure l'exemple 10 dels annexos (apartat 7.2.10) per a més detalls. Sobretot cal tenir en compte la codificació utilitzada en la pàgina HTML que conté l'applet (que és on es genera el text). Hauria de ser UTF-8, que és la més acceptada universalment i que manté els caràcters llatins (ñ, accents, ç, dièresi).

- ***hash_algorithm*** – Aquest paràmetre ens permet especificar l'algoritme de hash que s'aplicarà sobre el document abans de realitzar la signatura, els possibles valors són:

  1 *– SHA1*

3 – *SHA256*

5 – *SHA512*

S'ha de tenir en compte però que en funció del magatzem de certificats que s'estigui utilitzant podem trobar-nos que algun dels algoritmes no funciona correctament. Per exemple el CSP de windows XP no suporta SHA1.

4.6.2 Llibreries

Els següents paràmetres permeten indicar, en cas de signar mitjançant targeta criptogràfica o certificat en software (P12/PFX o JKS), on es troba el gestor de la targeta o el fitxer del keystore que cal utilitzar, respectivament.

Targeta criptogràfica:

- ***pkcs11_file*** – Indica el camí absolut on es troba la llibreria PKCS#11 que ha d'utilitzar l'applet per poder treballar amb la targeta criptogràfica. ***Aquest paràmetre està deprecated s'ha d'utilitzar el pkcs11_files.***
- ***pkcs11_files*** – Indica el camí absolut d'una o varies llibreries PKCS#11 que ha d'utilitzar l'applet per a poder treballar amb les targes criptogràfiques. Les rutes de les llibreries s'han d'especificar de la forma path_llibreria,ID;path_llibreria2,ID2;... L'identificador (ID) és una cadena de text arbitrària que l'applet utilitza internament per a diferenciar els diferents PKCS#11. Aquest ID també és el que es mostrarà al popup que demana el PIN en el text: "Introduïu el PIN per a (ID)":. També es poden especificar llibreries mútuament excloents per exemple per a diverses versions d'una mateixa llibreria, s'ha de fer de la forma [pathA1,pathA2,pathAN],ID1. En aquest cas l'applet anirà provant de carregar les llibreries especificades entre '[ ]' fins que en trobi una que està a disc, quan en troba una deixa de provar de carregar la resta. Evidentment es poden combinar les dues formes p.e: [pathA1,pathA2],ID1;pathB,ID2;...

Keystore:

- ***pkcs12_file*** – Indica el camí absolut al fitxer que conté el certificat en software. El format d'aquest fitxer ha de ser P12 o PFX.
- ***jks_file*** – Indica el camí absolut al keystore. El format d'aquest fitxer ha de ser JKS.
- ***pkcs11_addicionals_firefox*** – Indica el path i l'identificador de les llibreries addicionals PKCS11 que es vulguin fer servir en conjunt al magatzem de Firefox. El paràmetre s'ha d'especificar de la forma <path de la llibreria, identificador>. És possible especificar una llita de dispositius, separant-los mitjançant ";". L'identificador és important per a que internament l'applet pugui diferenciar els PKCS11. Aquest identificador es mostrarà també al label del popup que demana el pin amb el text: "introduïu el pin per a <identificador >:". ***Aquest paràmetre està deprecated s'ha d'utilitzar el pkcs11_files.***

4.6.3 Filtre

Els paràmetres que hi ha a continuació faciliten la selecció del certificat que cal utilitzar, si per exemple, l'aplicació ja el coneix (cal utilitzar l'àlies del certificat o el CommonName del SubjectDistinguishedName). Si la correspondència entre l'àlies o el CommonName és unívoca amb un sol certificat, s'utilitzarà el certificat així especificat per la generació de la

signatura, i el desplegable de selecció de certificat no es mostrarà a l'usuari. En canvi, si per exemple, existeix més d'un certificat amb el mateix CommonName, aleshores sí que es mostrarà el desplegable de selecció de certificat per tal que el client esculli el certificat amb que vol signar.

- ***selected_alias*** – Indica l'àlies del certificat que cal utilitzar en la signatura. Es comprova que existeixi en el dispositiu / magatzem seleccionat.
- ***selected_CN*** – Indica el CommonName dins del SubjectDistinguishedName del certificat que cal utilitzar en la signatura. Es comprova que existeixi en el dispositiu / magatzem seleccionat.

En cas de que es desitgi filtrar des de l'aplicació quins certificats mostrar com a disponibles, en funció de l'autoritat de certificació que ha emès el certificat, es disposa del següent paràmetre:

- ***allowed_CAs*** – Permet filtrar els certificats a mostrar en el diàleg mitjançant el CommonName de l'IssuerDistinguishedName que apareix en el certificat. Es poden indicar múltiples entrades separades per punts i comes (;). No té en compte la distinció majúscules/minúscules. Exemple: "EC-SAFP;EC-idCAT".

També tenim la possibilitat de filtrar per l'identificador de política de certificat. Haurem de fer servir el següent paràmetre:

- ***allowed_OIDs*** – Permet filtrar els certificats a mostrar en el diàleg mitjançant l'identificador de la directiva de certificat que apareix a l'extensió "Bases del certificat". Es poden indicar múltiples entrades separades per punts i comes (;).

També es pot aplicar un filtre sobre els certificats a mostrar al diàleg de selecció utilitzant una cadena de text lliure:

- ***subject_Text*** – Filtra els certificats a mostrar en el diàleg de selecció utilitzant una cadena de text que pot estar present en qualsevol dels camps del SubjectDistinguishedName del certificat. No té en compte la distinció majúscules/minúscules. Exemple: "Director General".

4.7 Paràmetres de sortida

Amb aquests paràmetres es controla com obtenir el resultat del procés de signatura. Hi ha 3 mètodes i es poden utilitzar a la vegada (no cal limitar-se a un forçosament). Per defecte estan tots desactivats, cosa que obliga a incloure com a paràmetre, com a mínim, un d'ells.

4.7.1 Sortida amb event Javascript

- ***js_event*** – Per activar la sortida utilitzant un event Javascript cal incloure aquest paràmetre amb valor *true*. El valor per defecte és *false*. La sortida es captura creant una funció Javascript amb el nom *onSignOK(signature)*. En el cas de generar vàries signatures, es poden capturar una a una o capturar-les totes amb una la funció *onMultiSignOK(signature1, signature2, ..., signatureN)*. Són les funcions que cridarà l'applet per a retornar les signatures. Es codificarà la sortida a Base64 si encara no ho està i si no es tracta d'un XML. El motiu de la conversió a Base64 és evitar problemes amb l'enviament del contingut (salts de línia o problemes de codificació).
- ***js_event_only*** – En cas que estigui activat el paràmetre js_event, aquest paràmetre indica si amb el retorn a la crida javascript *onSignOK(signature)* es retornarà la

signatura o només la crida a l'event. Per defecte aquest paràmetre pren el valor false. Si s'indica el valor true es farà només la crida indicant que ha finalitzat però sense passar la signatura.

- ***js_multisignature_only_event*** – En el cas de signatura massiva, la crida a l'event onMultiSignOk pot no suportar el volum de totes les signatures que s'han de mostrar al navegador degut a limitacions javascript. Activant aquest paràmetre s'evita que la crida a l'event onMultiSignOk retorni les signatures, cosa que permet finalitzar correctament l'operació en cas de que el volum conjunt de les signatures excedeixi la limitació javascript (6,5 MBytes aproximadament).

4.7.2 Sortida a document local

- ***local_file*** – Per crear el document que conté la signatura en un fitxer local cal incloure aquest paràmetre amb valor *true*. El valor per defecte és *false*.
- ***output_filename*** – Indica el camí absolut del fitxer en el que es desarà la signatura sempre que el paràmetre *local_file* estigui activat. Si es signa més d'un document i es desitja un nom per a cada signatura, cal indicar els noms separats per punts i comes (;).

Si *local_file* està activat però no s'especifica el nom del fitxer de sortida, aquest prendrà el nom del document original i li afegirà la cadena "*_signat.ext*", on *ext* indicarà el tipus de document (veure apartat 4.7.4). En tots els casos on no es disposi d'un path vàlid per afegir l'extensió comentada, es desarà el document signat a la carpeta CATCert que es crea (en cas de que no existeixi) a la carpeta personal de l'usuari. El nom del document prendrà la forma *document_x_signat.ext* on x serà un enter començant pel 0.

- ***local_file_result_message*** – Permet escollir si es presenta o no el missatge de sortida en pop-up informant del path on s'han guardat els documents signats. Per defecte el seu valor és "true". Si no es desitja que aparegui el pop-up de sortida, cal posar aquest paràmetre a "false".

4.7.3 Sortida emplenant un camp de formulari

- ***form_fill*** – Cal utilitzar aquest paràmetre i donar-li valor *true* si es desitja que el resultat de la signatura actualitzi un camp del formulari de l'HTML que invoca l'applet. Per defecte el valor és *false*. Es recodificarà la sortida a Base64 per als continguts que no siguin XML o que ja hagin estat codificats prèviament a Base64.
- ***form_fill_form*** – Indica el nom que té el formulari dins de l'HTML. Si no s'especifica pren el valor per defecte, *appletCATCertForm*.
- ***form_fill_field*** – Nom del control/camp de l'HTML que cal actualitzar amb el valor de la signatura.

4.7.4 Format de sortida

- ***output_mode*** – Amb aquest paràmetre es pot decidir el format de representació de la signatura. En principi tan sols té sentit en el cas de signatures CMS en que es pot triar entre codificació binaria o Base64. Per defecte pren el valor de Base64. Per a signatures XML o PDF el format de sortida no es pot modificar (cal tenir en compte en el cas de PDF que s'autocodificarà a Base64 sempre que s'utilitzi un event Javascript o un camp del formulari com a mètode de sortida). El paràmetre es configura amb un valor numèric:

  1 – Codificació binaria.

2 – Codificació Base64, opció per defecte en signatures CMS/CAdES.

3 – XML, opció per defecte en signatures XML.

4 – PDF, opció per defecte signant documents PDF.

4.8 Paràmetres signatura XML

Paràmetres específics per a les signatures XML (XMLdsig i XAdES en les formes suportades).

- ***n_enveloping*** – En el cas d'utilitzar qualsevol de les formes de signatura XML enveloping i havent de signar múltiples documents, ja sigui documents locals o remots utilitzant la seva URL, és possible generar una única signatura que inclogui referències a tots els documents signats.
- ***n_detached*** – Igual que en el cas anterior però per a les signatures XML detached, permet generar una signatura XML amb referències a tots els documents signats. La diferència respecte la opció anterior és la possibilitat de poder utilitzar també els resums criptogràfics (hash) precalculats dels documents a signar.
- ***canonicalizationWithComments* –** Indica si l'algoritme de canonicalització emprat en la generació de la signatura XML tindrà en compte comentaris o no. Per defecte pren el valor *false*, i per tant ometrà els comentaris (opció requerida per l'W3C). En cas de voler el contrari, posar el valor del paràmetre a *true*.
- ***includeXMLTimestamp* –** En cas que la signatura incorpori un segell de temps, indica si aquest ha de ser del tipus XMLTimeStamp o EncapsulatedTimeStamp. Per defecte pren el valor *true* que indica que el segell serà de tipus XMLTimeStamp, si es vol que el tipus sigui EncapsulatedTimeStamp s'ha de posar el valor del paràmetre a *false*.
- ***uris_to_be_signed* –** En cas de signatures de tipus XAdES enveloped, és possible mitjançant aquest paràmetre, especificar el node o nodes a signar, en lloc de signar tot el document. Els identificadors s'han d'especificar separats per ";" i cal que es corresponguin amb el valor del atribut "id" dels nodes del document sobre els que es vol realitzar la signatura. En cas de no especificar res, es signarà tot el document. En cas que s'especifiquin identificadors de nodes que no existeixen al document, es mostrarà un missatge d'error indicant que els atributs no existeixen al document a signar.

4.9 Paràmetres de signatures AdES-EPES

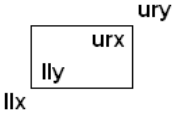
- ***signature_policy*** – Permet incloure la política contra la que s'haurà de validar la signatura generada. El valor del paràmetre haurà de ser l'OID de la política de signatura (implica l'ús del paràmetre signature_policy_hash). Paràmetre disponible tant per XAdES com per CAdES (convertim la forma de la signatura a EPES).
- ***signature_policy_hash*** – El valor d'aquest paràmetre conté el hash codificat en Base64 del document XML que descriu la política de signatura contra la que es validarà la signatura generada (ve a ser una mena de control de versió). Aplica a XAdES i CAdES.
- ***signature_policy_qualifier*** – Qualificador de l'identificador de la política de signatura.

- ***signer_role*** – Permet incloure el rol del signatari (ClaimedRole) com a element signat dins de la signatura. Aplica a XAdES i CAdES.
- ***commitment_identifier*** – Permet especificar el compromís de signatura. Aplica a XAdES i CAdES.
- ***commitment_description*** – Descripció del compromís de signatura, en cas que aquest s'hagi especificat. És un paràmetre opcional, és a dir, es pot especificar compromís sense descripció. Disponible només per signatures XAdES.
- ***commitment_object_reference*** – Referència a l'atribut sobre el que s'aplica el compromís de signatura. En cas de no especificar res, el compromís s'aplica sobre tots els atributs. Disponible només per signatures XAdES.
- ***protectKeyInfo*** – Valor booleà que indica si addicionalment a part de les dades i les signatureProperties requerides a signar s'ha de signar també l'element <KeyInfo> (conté la informació de la clau amb la que s'ha realitzat la signatura) . Per defecte pren el valor false.

4.10 Paràmetres signatura CMS / PDF

Paràmetres opcionals i específics per a signatures CMS i incrustades (CMS detached) en documents PDF:

- ***TimeStamp_CMS_signature*** – Com el seu nom indica, permet afegir un segell de temps a les signatures CMS i per extensió a les que s'incrusten en els PDF. Per activar-ho cal posar el valor del paràmetre a *true*. Per defecte el valor és *false*. El servei de segellat de temps (TSA) utilitzat per defecte és el que s'ofereix a PSIS (http://psis.catcert.net/psis/catcert/tsp). En cas de voler utilitzar un altre, fer servir el paràmetre *cmsts_tsa_url* definit a l'apartat 4.4.
- ***pdf_signature_field*** – Si el document PDF a signar disposa de camps de signatura buits, és possible indicar el nom del camp que es desitja emplenar. D'aquesta manera s'evita que en el diàleg que apareix a l'hora de signar un document PDF s'hagi d'escollir aquesta opció.
- ***pdf_visible_signature*** – Permet indicar a l'applet que la signatura que es crearà al document PDF sigui invisible (valor a *false*). Per defecte el valor és *true* (visible). Si hi ha camps de signatura, aquest paràmetre no té influència i s'intentarà emplenar un dels camps buits.
- ***pdf_signature_rectangle*** – Quan no hi ha camps de signatura i la signatura ha de ser visible, hi ha l'opció de seleccionar on es crearà: coordenades de la pàgina i número de pàgina. El valor d'aquest paràmetre per defecte és *100 100 200 200 1*. Les coordenades s'indiquen de forma numèrica i separades per espais: llx lly urx ury page_nr.



És possible indicar que la signatura es coloqui a l'última plana del document PDF especificant el valor "0" a "nr".

Addicionalment també és possible realitzar només una signatura en el document però fer que aquesta sigui visible a totes les planes del mateix. Per a fer-ho s'ha de especificar el valor "-1" a "nr".

- ***pdf_signature_rotation*** – Permet rotar el camp de signatura visible dins del PDF, rotant la imatge i el text del mateix. Els possibles valors són 90,180,270. El gir es fa en sentit anti-horari el nombre de graus especificats per paràmetre.

- ***pdf_reason*** – Permet indicar el motiu de la signatura (camp propi d'Adobe). L'ús d'aquest paràmetre deshabilita aquesta opció en el diàleg de signatura de documents PDF.
- ***pdf_location*** – Permet indicar una localització (camp propi d'Adobe). Com en el cas anterior, si s'utilitza el paràmetre, desapareix del diàleg de signatura de documents PDF.
- ***pdf_certification_level*** – Permet especificar el nivell de certificació de la signatura d'un document PDF. Els possibles valors d'aquest atribut són:
  0 : Document no certificat (opció per defecte).
  1 : Document certificat. No es permeten canvis.
  2 : Document certificat. Es permet l'emplenament de formularis.
  3 : Document certificat. Es permet l'emplenament de formularis, i anotacions.
- ***pdf_signature_image*** – Ens permet escollir la imatge que apareix a la signatura d'un document PDF. El valor d'aquest paràmetre haurà de ser la codificació en Base64 del fitxer imatge.
- ***pdf_reserved_space*** – Permet especificar l'espai de memòria a reservar per la signatura dins del document PDF. Cal indicar aquest valor en KBytes. Internament, i donat que al document PDF la signatura s'incrusta codificada en hexadecimal, l'espai real que es reserva és, aproximadament, del doble. És a dir, que la mida del document augmentarà aproximadament en un valor del doble de l'indicat en aquest paràmetre. Per defecte, el valor que pren aquest paràmetre és:

  - Signatura CMS: 26 KB
  - Signatura CAdES: 500 KB

- ***pdf_dinamic_signature_image*** – Per defecte aquest paràmetre té valor: false. Posar aquest paràmetre com a true, força a l'applet a generar la crida javascript *pdfDinamicSignatureImage* retornant els atributs del certificat i el seu valor al fer la validació del certificat del signant contra PSIS, dins d'aquest mètode i en funció de validar el valor d'aquests atributs es pot setejar de forma dinàmica els paràmetres de visualització de la signatura com p.e exemple la imatge.
- ***abort_pdf_sign_operation*** – Aquest paràmetre només es pot setejar de forma dinàmica i només té efecte en cas que el paràmetre *pdf_dinamic_signature_image* estigui activat. S'utilitza per avortar l'operació en cas que algun dels atributs retornats per la funció javascript *pdfDinamicSignatureImage* no sigui l'esperat. Per defecte el valor és false.
- ***sign_first_sign_fields*** – Aquest paràmetre força que en cas que el PDF disposi de camps de signatura només es pugui triar un d'aquests camps, a diferencia del funcionament normal que a banda de deixar-te triar els camps et permet crear una nova signatura. Si existeixen diversos camps de signatura l'activació d'aquest paràmetre només et permetrà triar el camp desitjat, si només hi ha un camp de signatura el seleccionarà de forma automàtica, si no n'hi ha cap t'afegirà un camp de signatura amb els paràmetres de visualització configurats igual que ho faria amb el funcionament normal.
- ***pdf_show_adobe_sign_validation*** – Per defecte pren el valor false. Si s'indica el valor true, a la signatura es mostrarà el tick, cross o el interrogant i la descripció de l'estat (signature valid, signature invalid, signature not yet verified) per defecte que fa l'adobe a banda de l'imatge i el text que hi hagi configurat(pdf_signature_image,

reason, location etc...), si no es seteja a true aquesta info extra de l'adobe no es mostrarà.

4.11 Paràmetres propis JVM

Els applets java disposen del següent paràmetre propi que permet configurar la parametrització de l'arrencada de la pròpia JVM:

- ***java_arguments*** – Com a valor d'aquest argument es poden passar els paràmetres d'arrencada de la JVM com es faria per exemple amb un tomcat. Es pot especificar per exemple la mida del heap (-Xmx –Xms), setejar una propietat del sistema (-Dfoo="text"). Aquests paràmetres es poden ajustar en cas de necessitat per a millorar el rendiment de l'execució en alguns casos.

5. Exportació de certificats

A banda de signar l'eina incorpora la possibilitat d'extreure certificats dels diferents magatzems (Windows, Firefox, MACOSX, dispositius PKCS11). Els certificats retornats per l'applet estaran codificats en base64.

Igual que amb el mode de signatura, la exportació també és pot realitzar en mode embedded. Per a fer ús d'aquesta funcionalitat existeixen aquestes dues crides javascript:

- *Comunicació Javascript → applet*
  - Mètode *exportCertFromJS()* – Inicia el procés de exportació dels certificats. Apareixerà el pop-up de selecció i un cop seleccionat el certificat l'applet el retornarà com a paràmetre codificat en base64 amb la crida javascript onCertExported(cert).
  - Mètode *exportCertFromJS(alies)* – Aquesta crida es realitza amb l'applet en mode embedded (paràmetre embedded = true), i permet exportar el certificat en base64 corresponent a l'àlies passat com a paràmetre. Aquesta crida retornarà la crida javascript onCertExported(cert) on cert serà el certificat codificat en base64.
- Comunicació applet → Javascript
  - Funció *onCertExported(cert)* – Captura el retorn de l'applet a la crida exportCertFromJS i rep com a paràmetre cert el certificat exportat en base64.

Per a més informació sobre el funcionament d'aquests mètodes es poden consultar els exemples 27 i 28.

A banda de la funció d'exportació que emula el funcionament de la signatura, s'ha afegit la possibilitat de recuperar el certificat que s'ha utilitzat per a signar sense la necessitat de triar-lo d'una llista sinó invocant directament l'applet per a que ens retorni el certificat amb el que s'ha fet l'última operació de signatura.

Per a l'ús d'aquesta funcionalitat els mètodes de comunicació entre l'applet i la plana web són:

- *Comunicació Javascript → applet*
  - Mètode *getSignCertificate()* – Recupera el certificat que s'ha utilitzat per a realitzar la última signatura amb l'applet.
- Comunicació applet → Javascript
  - Funció *getSignCert(cert)* – Captura el retorn de l'applet a la crida getSignCertificate() i rep com a paràmetre (cert) el certificat del signant exportat en base64. En cas que encara no s'hagi realitzat cap signatura amb l'applet, al paràmetre "cert" vindrà el missatge informant que es necessari realitzar una signatura abans de poder recuperar el certificat del signant.

6. Instal·lació i ús de l'eina

6.1 Instal·lació

La instal·lació és senzilla. Tan sols cal copiar les 4 llibreries en el servidor Web i referenciar-les (camí absolut o relatiu) correctament des de l'HTML. Veure exemples en l'annex 7.1.

Per a que tot funcioni correctament, ***cal signar les 4 llibreries*** (les del paquet ja ho estan), ja que necessiten accedir a recursos locals del client (memòria i certificats).

Degut a una limitació del nou Internet Explorer 7 (i IE6 amb les darreres actualitzacions), l'applet apareixerà emmarcat i requerirà ser activat per l'usuari de forma manual, tant si es carrega normalment com utilitzant un control Javascript. Per a poder evitar aquest comportament, molest per a l'usuari final, cal seguir les indicacions que proposa Microsoft en el següent document:

http://msdn.microsoft.com/library/default.asp?url=/workshop/author/dhtml/overview/activating_activex.asp.

Les solucions proposades, bàsicament el que intenten és muntar el codi de l'applet en una funció que es troba en un script fora de la pàgina HTML. Un exemple senzill de com solucionar el problema es pot veure dins dels annexos a l'apartat **¡Error! No se encuentra el origen de la referencia.** o en l'exemple 11 del paquet de lliure distribució. Aquesta limitació no existeix a la resta de navegadors, com per exemple Firefox.

6.2 Ús

Un cop el client rep l'HTML, el navegador procedirà a descarregar de forma dinàmica les llibreries que siguin necessàries segons la configuració.

Quan s'inicia el procés de signatura, depenent del magatzem de certificats que s'hagi seleccionat, apareixerà una finestra demanant el PIN per a poder accedir al magatzem i mostrar els certificats que hi ha disponibles (cas de tots els magatzems excepte el de Windows) i a continuació el diàleg de selecció de certificat i el corresponent avís legal informant que s'està a punt de generar una signatura electrònica. En els 2 casos amb el color i logo indicats. En el cas del magatzem de Windows, primer es selecciona el certificat i posteriorment, el propi component de seguretat de Windows gestiona l'accés al magatzem sol·licitant el PIN.

Existeix un cas especial, que és el de la signatura de documents PDF. L'eina permet incloure informació pròpia d'aquest tipus de documents. Si no s'ha indicat prèviament (veure 4.6), caldrà omplir un formulari com els següents.

Si hi ha camps de signatura disponibles...

...si no n'hi ha

6.3 Applet en mode servidor

Podem fer servir l'applet en mode servidor per la generació de signatures, atacant directament les seves llibreries.

Es poden generar els següents tipus de signatures:

- CMS attached / detached
- CAdES attached / detached: CAdES-BES / CAdES-T.
- XMLDSig detached / enveloping / enveloped
- XAdES detached / enveloping / enveloped: XAdES-BES / XAdES-T

També es poden signar documents PDF (signatura CMS o CAdES) fent servir l'applet en mode servidor.

Amb el paquet es distribueix un projecte eclipse amb exemples per a la generació de signatures directament des de codi utilitzant les llibreries de l'applet.

7. Annexos

7.1 Exemples HTML

7.1.1 Tag <object> i funcions Javascript

Per a incloure l'applet en una pàgina HTML cal utilitzar la següent estructura dins del <*BODY*>. Conté els paràmetres per duplicat, ja que els navegadors basats en Mozilla utilitzen el que conté el tag *embed*. Els paràmetres fora d'aquest tag són els que interpreta Internet Explorer.

```
<object
classid = "clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
width = "130" height = "25"
id = "appletCATCert">
        <param name = "code" value = "org.catcert.AppletSignatura">
        <param name = "archive" value = "appletCATCert1.7.jar, CATCertXMLlib1.1.jar, CATCertCMSlib1.0.jar,
CATCertPDFlib1.0.jar">
        <param name = "mayscript" value = "true">
        <param name = "scriptable" value = "true">
        <param name = "type" value = "application/x-java-applet;version=1.5">
        <param name = "keystore_type" value = "1">
        <param name = "signature_mode" value = "12">
        <param name = "js_event" value = "true">
        <comment>
                <embed
                        type = "application/x-java-applet;version=1.5"
                        code = "org.catcert.AppletSignatura" archive = "appletCATCert1.7.jar,
CATCertXMLlib1.1.jar, CATCertCMSlib1.0.jar, CATCertPDFlib1.0.jar"
                        scriptable = "true"
                        width = "130" height = "25"
                        name = "appletCATCert"
                        mayscript = "true"
                        scriptable = "true"
                        keystore_type = "1"
                        signature_mode = "12"
                        js_event = "true">
                        <noembed>
                        Atenció! Estàs intentant executar un applet i el navegador no t'ho permet.
                        Possibles raons:<br/>
                        - No està permès executar-los per la política de seguretat.<br/>
                        - Aquest navegador no disposa del Java Plug-in per poder executar applets.<br/>
                        <a href = "http://java.sun.com/products/plugin/downloads/index.html">
                        Aconsegueix la darrera versió del Java Plug-in aquí.
                        </a>
                        </noembed>
                </embed>
        </comment>
</object>
```

Per a poder utilitzar els mètodes públics de l'applet descrits a l'apartat 2 caldrà utilitzar *document.appletcatcert.mètode*. A continuació hi ha un exemple de les funcions que es poden utilitzar per a interactuar amb l'applet. Aquestes funcions hauran d'anar preferiblement dins del tag <*HEAD*> de l'HTML.

```
<SCRIPT LANGUAGE = "JavaScript">
```

```
<!--
function onSignOK(signature) {
        alert ('Signatura generada correctament:\n' + signature);
}
function onMultiSignOK(signature1, signature2,...) {
        alert ('Signatures generades correctament:\n' + signature1 + '\n' + signature2 + ...);
}
function onSignCancel() {
        alert('Procés cancel·lat per l\'usuari');
}
function onSignError(msg) {
        alert('Error durant la generació de la signatura: \n' + msg);
}
function onSignLoad() {
        alert('Applet de signatura carregat correctament');
}
function onLoadError(msg) {
        alert('Error durant la càrrega de l\'applet:\n' + msg);
}
function setAppletParam(name,value) {
        document.appletcatcert.set(name,value);
}
function sign() {
        document.appletcatcert.signFromJS();
}

// -->
</SCRIPT>
```

7.1.2 Exemple diàleg usuari per a seleccionar el document a signar

Codi a incloure en el <*BODY*>.

```
<input type ="button" value="examinar..." onclick="inputFileOnChange();">
<input type = "xhidden"  id= "path" name= "path" size= "100" >
```

Codi a incloure en el <*SCRIPT*>.

```
function inputFileOnChange(){
        try{
                document.appletcatcert.openFileDialog();
        }catch(Exception){
                // chrome, safari i opera
                document.appletcatcert[1].openFileDialog();
        }
}
function onFileUpload(path){
        document.getElementById('path').value = path;
}
```

7.1.3 Exemple utilitzant targeta criptogràfica (PKCS#11)

Paràmetres a modificar respecte a l'exemple 7.1.1 en la part corresponent a Internet Explorer:

```
<param name = "keystore_type" value = "3">
<param name = "pkcs11_files" value = " C:\WINDOWS\system32\aetpkss1.dll,TCAT">
```

Paràmetres a modificar respecte a l'exemple 7.1.1 en la part corresponent a Mozilla:

```
keystore_type = "3"
pkcs11_files = " C:\WINDOWS\system32\aetpkss1.dll,TCAT">
```

També es poden especificar diverses llibreries separant-les per ';' de la següent forma, en aquest cas p.e la TCAT i la tarja de la FNMT.

```
pkcs11_files = " C:\WINDOWS\system32\aetpkss1.dll,TCAT;C:\Windows\System32\FNMT.dll,FNMT"
```

En cas de que puguin existir diverses versions d'una mateixa llibreria en el sistema es poden especificar de manera que el sistema només carregui una de les especificades (la primera que trobi per l'ordre que s'han configurat; s'han de posar entre '[versio1,versio2...]' (en aquest cas provaria de carregar la aetpkss2.dll si la carrega no continuaria, cas que no provaria de carregar la aetpkss1.dll):

```
pkcs11_files = "[C:\WINDOWS\system32\aetpkss2.dll ,C:\WINDOWS\system32\aetpkss1.dll],TCAT"
```

Evidentment es poden combinar les dues formes, diferents llibreries per a diferents dispositius amb diferents versions d'una llibreria:

```
pkcs11_files = "[C:\WINDOWS\system32\aetpkss2.dll ,C:\WINDOWS\system32\aetpkss1.dll],TCAT
        ;C:\WINDOWS\system32\aetpkss1.dll,TCAT;C:\Windows\System32\FNMT.dll,FNMT"
```

7.1.4 Botó applet invisible; crida utilitzant botó HTML

Cal indicar que a l'<*object*> que les dimensions de l'applet seran nul·les (width = "0" height = "0"). El codi a incloure en el <*BODY*> serà el següent:

```
<input type = "button" name = "sig_button" value = "Signa" onclick = "sign()">
```

7.1.5 Exemple complet

Signatura XAdES-BES detached on l'entrada és el hash del document a signar i la sortida utilitzant un event Javascript i omplint un camp de formulari. El magatzem de certificats serà el personal de l'usuari a Windows.

```
<HTML>
<HEAD>
<TITLE> Test appletCATCert </TITLE>
<META NAME="Author" CONTENT="Oscar Burgos">
<META NAME="Description" CONTENT="Pàgina de Test de l'applet de CATCert">
<SCRIPT LANGUAGE = "JavaScript">
```

```
<!--
function onSignOK(signature) {
        alert ('Signatura generada correctament:\n' + signature);
}
function onSignCancel() {
        alert('Procés cancel·lat per l\'usuari');
}
function onSignError(msg) {
        alert('Error durant la generació de la signatura: \n' + msg);
}
function onSignLoad() {
        alert('Applet de signatura carregat correctament');
}
function onLoadError(msg) {
        alert('Error durant la càrrega de l\'applet:\n' + msg);
}
// -->
</SCRIPT>
</HEAD>
<BODY>

Signatura XAdES-BES detached

<form name="appletCATCertForm">
<table align="center" border="1" cellspacing="0" cellpadding="10">
<tr><td>
<table border="0">
<tr>
        <td>Signatura generada:</td>
        <td><textarea name="result" style="width:350px" cols="80" rows="10"></textarea></td>
</tr>
</table>
</td></tr>
<tr><td align = "center">
<object
classid = "clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
codebase = http://java.sun.com/update/1.5.0/jinstall-1_5-windows-i586.cab#Version=5,0,0,5
width = "130" height = "25"
id = "appletCATCert">
        <param name = "code" value = "org.catcert.AppletSignatura">
        <param name = "archive" value = "appletCATCert1.7.jar, CATCertXMLlib1.1.jar, CATCertCMSlib1.0.jar,
CATCertPDFlib1.0.jar">
        <param name = "mayscript" value = "true">
        <param name = "scriptable" value = "true">
        <param name = "type" value = "application/x-java-applet;version=1.5">
        <param name = "keystore_type" value = "1">
        <param name = "signature_mode" value = "12">
        <param name = "doc_type" value = "3">
        <param name = "document_to_sign" value = "wu6nT5Z9b0gu7jBzri+8v6fysVc=">
        <param name = "form_fill" value = "true">
        <param name = "form_fill_form" value = "appletCATcertForm">
        <param name = "form_fill_field" value = "result">
        <param name = "signButtonCaption" value = "Signa">
        <comment>
                <embed
                        type = "application/x-java-applet;version=1.5"
                        code = "org.catcert.AppletSignatura" archive = "appletCATCert1.7.jar,
CATCertXMLlib1.1.jar, CATCertCMSlib1.0.jar, CATCertPDFlib1.0.jar"
                        scriptable = "true"
                        pluginspage = http://java.sun.com/products/plugin/index.html#download
                        width = "130" height = "25"
                        name = "appletCATCert"
                        mayscript = "true"
                        scriptable = "true"
```

```
                        keystore_type = "1"
                        signature_mode = "12"
                        doc_type = "3"
                        document_to_sign = "wu6nT5Z9b0gu7jBzri+8v6fysVc="
                        form_fill = "true"
                        form_fill_form = "appletCATCertForm"
                        form_fill_field = "result"
                        signButtonCaption  = "Signa">
                        <noembed>
                        Atenció! Estàs intentant executar un applet i el navegador no t'ho permet.
                        Possibles raons:<br/>
                        - No està permès executar-los per la política de seguretat.<br/>
                        - Aquest navegador no disposa del Java Plug-in per poder executar applets.<br/>
                        <a href = "http://java.sun.com/products/plugin/downloads/index.html">
                        Aconsegeuix la darrera versió del Java Plug-in aquí.
                        </a>
                        </noembed>
                </embed>
        </comment>
</object>
</td></tr>
</table>
</form>
</BODY>
</HTML>
```

7.1.6 Exemple de solució en aplicacions Web amb autenticació de client

En el cas d'utilitzar l'applet en una aplicació Web que requereixi autenticació de client, és aconsellable col·locar les llibreries en una carpeta virtual que no estigui securitzada. El motiu de separar les llibreries de la part segura de l'aplicació és evitar que el plugin de Java, que és l'aplicació que s'ocupa de la descàrrega de les llibreries, sol·liciti vàries vegades el certificat de client. Això succeeix perquè, per motius de seguretat, el plugin no reaprofita les connexions obertes pel navegador i per tant cal tornar a fer l'autenticació tantes vegades com llibreries a descarregar. La solució, a part de la intervenció en el servidor, requereix l'ús del paràmetre CODEBASE.

7.1.7 Exemple de funcionament en mode embedded

A continuació és descriu el flux d'execució que segueix l'applet quan s'utilitza en mode embedded. En el exemple realitzarem una signatura de tipus XAdES-T detached sobre un hash precalculat, el magatzem de certificats serà en funció del sistema operatiu i el navegador i mostrarem el resultat en un event javascript. L'exemple complet és pot veure en l'exemple 25.

La configuració de l'applet quedarà de la següent forma:

```
<object
       classid = "clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
       codebase      =      "http://java.sun.com/update/1.5.0/jinstall-1_5-windows-
i586.cab#Version=5,0,0,5"
       width="0" height="0" id="appletcatcert">
       <param name = "code" value = "org.catcert.AppletSignatura">
       <param     name     =     "archive"     value     =     "appletCATCert2.2.2.jar,
CATCertXMLlib1.2.1.jar, CATCertCMSlib1.2.1.jar, CATCertPDFlib1.1.1.jar">
       <param name = "mayscript" value = "true">
       <param name = "scriptable" value = "true">
       <param name = "type" value = "application/x-java-applet;version=1.5">
       <param name = "keystore_type" value = "0">
       <param name = "signature_mode" value = "16">
       <param name = "doc_type" value = "3">
       <param name = "js_event" value = "true">
       <param name = "embedded" value = "true">
       <param name = "document_to_sign" value = "TrESBuui6RAvQUf/7Az9XqeLf+I=">
       <comment>
        <embed
       type = "application/x-java-applet;version=1.5"
       code  =  "org.catcert.AppletSignatura"  archive  =  "appletCATCert2.2.2.jar,
CATCertXMLlib1.2.1.jar, CATCertCMSlib1.2.1.jar, CATCertPDFlib1.1.1.jar"
       scriptable = "true"
       pluginspage = "http://java.sun.com/products/plugin/index.html#download"
       width = "0" height = "0"
       name = "appletcatcert"
       mayscript = "true"
       scriptable = "true"
       keystore_type = "0"
       signature_mode = "16"
       doc_type = "3"
       js_event ="true"
       embedded="true"
       document_to_sign="TrESBuui6RAvQUf/7Az9XqeLf+I=">
       <noembed>
       ...
       </noembed>
       </embed>
    </comment>
</object>
```

keystore_type = 0 indica que el magatzem anirà en funció del sistema operatiu i el navegador.

signature_mode = 16 Indica que es una XAdES T detached sobre un hash precalculat.

doc_type = 3 Indica que el document ha signar serà un hash

js_event = true Indica que la signatura realitzada es retornarà en un event javascript.

embedded = true Indica que el mode de funcionament serà embedded.

document_to_sign Indica el hash que signarem.

Un cop fet això quan es carregui l'applet, retornarà la crida javascript ***onReturnCerts(alies)*** passant per paràmetre els àlies disponibles al magatzem. Aleshores el que haurem de fer es pintar-los en pantalla. Podeu veure un exemple d'això buscant la funció ***onReturnCert(alies)*** al fitxer *applet_callbacks.js* inclòs als exemples. Al exemple només hi ha una proposta de com realitzar l'operació però hi ha llibertat total en la forma de com plasmar els alies en l'html.

Juntament amb l'operació anterior, a l'hora de signar serà necessari recuperar l'alies seleccionat per a passar-lo a l'applet per a que pugui saber amb quin certificat es vol realitzar la signatura. Això també dependrà totalment de com s'ha decidit implementar la selecció de l'alies en el html.

7.1.8 Exemple de retorn per parts via javascript

Com s'ha comentat anteriorment en la pròpia documentació, la comunicació java -> javascript presenta problemes de funcionament quan els paràmetres que s'envien superen els 4MB, per a resoldre això el que s'ha fet és crear un nou mètode que en cas que el event js estigui activat (js_event = true) i la signatura resultant superi els 4MB en comptes de retornar-se sencera a través del *onSignOk,* es farà per parts a través del *jsEventAppender,* si a priori se sap que les signatures no excediran aquesta mida o que el retorn no es vol via js no es necessari implementar aquesta funció.

A continuació es defineix el flux i els elements necessaris per a implementar aquesta funció. Com que el problema es retornar la signatura sencera l'applet anirà retornant la signatura per parts, i la funció javascript s'encarregarà d'anar concatenant el resultat a un element de la pròpia plana web:

A la plana on tenim l'applet creem p.e el següent element i especifiquem la següent funció js:

```
<!-- element on s'anira desant la signatura en parts en cas que superi els 4MB (js_event = true) -->
<input type="hidden" id="appletCATCertReturnId"/>
```

```
/**
Funció que es crida des de l'applet amb les parts de la signatura. Aquesta crida es realitza si la signatura resultant excedeix els 4MB, sinó la signatura arriba al signOnOK
**/
function jsEventAppender(data){
        if(data == "JS_EVENT_EXCEDS_SIZE_LIMIT"){
                alert(document.getElementById("appletCATCertReturnId").value);
        }else{
                document.getElementById('appletCATCertReturnId').value +=  data;
        }
}
```

En les successives crides que rebem a jsEventAppender com paràmetre data arriben trossos de la signatura que anem concatenant a dins d'un element de l'html. Com a última crida per saber que ja s'ha passat tota la signatura rebrem la cadena "JS_EVENT_EXCEDS_SIZE_LIMIT", aleshores ja sabrem que a l'objecte tenim la signatura sencera i podrem fer el que necessitem.

7.2 Llistat d'exemples del paquet de lliure distribució

S'han unificat tots els exemples. A tots els exemples s'ha afegit un panell modal durant la càrrega i la generació de les signatures per a deixar clar a l'usuari que s'està realitzant una operació. S'ha afegit també un control amb un missatge d'error en cas que l'applet no carregui en un temps determinat. S'ha encapsulat els *alert()* dels callbacks degut a un problema de Chrome (no afecta a la resta de navegadors), el workaround proposat funciona correctament (veure *appletAlert.alert(msg)* dels exemples).

S'han generat dos fitxers *.js* un que conté tots els callbacks que pot produïr l'applet amb crides cap a *javascript* i un altre amb els objecte del applet encapsulat per a poder realitzar les crides sobre el mateix.

S'han unificat tots els exemples de *unix/windows/macosx* de manera que tots facin servir *keystore_type = 0* per a la detecció automàtica del magatzem.

Arrel d'aquests canvis s'han eliminat varis exemples que ja no tenien sentit.

Tots els exemples utilitzen els estils de *bootstrap*[1] i *jQuery*[2], no es garanteix el total funcionament dels elements gràfics dels exemples per a versions de IE8 o inferiors.

7.2.1 Exemple 1

Es genera una signatura XAdES-BES detached. El document a signar és el resum criptogràfic del document a signar. Es genera un event Javascript que cal capturar per a recuperar la signatura. L'applet es crida utilitzant un botó HTML que utilitzarà Javascript per a invocar-lo.

7.2.2 Exemple 2

Signatura d'un document PDF amb la configuració estàndard (signatura a la primera pàgina i visible). El document a signar és un document PDF (amb 2 camps de signatura ja preparats) incrustat a l'HTML, codificat en Base64. Els camps propietaris del PDF, motiu ("Aprovant el document") i lloc ("Barcelona"), s'han fixat per paràmetre. La signatura contindrà un segell de temps, visible des de l'Adobe Reader. El document signat serà un fitxer local amb nom C:\test_sig.pdf. L'applet es crida directament al carregar la pàgina HTML (onload = sign()).

7.2.3 Exemple 3

Signatura d'un document PDF amb la configuració estàndard. La signatura contindrà un segell de temps, visible des de l'Adobe Reader. El document a signar es selecciona amb un diàleg i es passa a l'applet utilitzant Javascript. La sortida serà un fitxer local amb el mateix nom que l'original, afegint l'extensió "_signat.pdf". L'applet es crida utilitzant el seu propi botó, i se li indica el text a mostrar: "Signa".

7.2.4 Exemple 4

Signatura XAdES-BES detached d'un document (pot ser de qualsevol tipus) utilitzant els certificats obtinguts de la targeta de CATCert. S'accedeix per tant directament a la targeta utilitzant la seva llibreria PKCS#11. El document a signar es selecciona amb un diàleg i es passa a l'applet utilitzant Javascript. La sortida, que serà una signatura XML, es pintarà a l'àrea de text reservada a la pàgina HTML.

[1] http://getbootstrap.com/

[2] http://jquery.com/

7.2.5 Exemple 5

Signatura CMS attached codificada en Base64 (per defecte) de qualsevol tipus de document. El document a signar es selecciona amb un diàleg i es passa a l'applet utilitzant Javascript. La signatura incorporarà un segell de temps. La sortida serà un fitxer local amb el mateix nom que l'original, afegint l'extensió "_signat.p7s".

7.2.6 Exemple 6

Signatura XAdES-T enveloping del document seleccionat al diàleg. La sortida serà un fitxer local amb el mateix nom que l'original, afegint l'extensió "_signat.xml".

7.2.7 Exemple 7

Signatura d'un document PDF amb la configuració estàndard. El document a signar està incrustat a l'HTML, codificat en Base64. La sortida es recollirà capturant l'event Javascript que envia l'applet un cop realitzada la signatura. Donat que es tracta d'un document binari, el document de sortida estarà codificat en Base64. L'applet es crida directament al carregar la pàgina HTML (onload = sign()).

7.2.8 Exemple 8

Signatura XAdES-BES detached. Els documents a signar són 3 resums criptogràfics (hash) dels documents a signar. Les signatures generades cal capturar-les utilitzant els events Javascript que genera l'applet per a cada signatura. Es rep un event per a la signatura de cada document i un event final amb totes les signatures.

7.2.9 Exemple 9

Signatura d'un document PDF amb la configuració estàndard. Els 2 documents (PDF) a signar es troben a servidors remots, i s'indica la URL de cada un d'ells. L'applet els descarregarà i els signarà posteriorment. Es guardaran els PDF signats a la carpeta personal de l'usuari.

7.2.10 Exemple 10

Signatura XAdES-T enveloped. Es signarà un XML generat amb les dades del formulari de l'HTML (tal com es fa habitualment amb l'eina Formsign). Primer el previsualitzarem i després el signarem. La signatura generada es pintarà a un dels camps del formulari.

7.2.11 Exemple 11

Signatura CAdES-BES detached. Es signarà un text generat amb les dades del formulari de l'HTML (tal com es fa habitualment amb l'eina Formsign). Primer el previsualitzarem i després el signarem. La signatura generada es pintarà a un dels camps del formulari codificada en Base64.

7.2.12 Exemple 12

Exemple igual que el presentat a l'exemple 1. En aquest, a part d'utilitzar l'script que habilita automàticament l'applet, s'indica el logo principal que cal utilitzar en les finestres de l'aplicació i el color de fons s'ha fixat a blanc.

7.2.13 Exemple 13

Signatura CAdES-BES detached. Es signa el document a partir del seu hash, que es passa per paràmetre (veure codi font). La signatura apareixerà en el camp del formulari codificada en BASE64 (comportament per defecte). A més s'aplica un filtre en el selector de certificats,

que farà que es mostrin tan sols aquells que continguin el text "catcert" dins del SubjectDistinguishedName.

7.2.14 Exemple 14

Signatura XAdES-EPES que segueix l'estàndard CCI respecte al format d'e-factura acceptat per l' AEAT. Es poden veure més detalls sobre el format a http://www.asociacioncci.es/Paginas/eFactura_AEAT-CCI.aspx.
Per a que la signatura compleixi amb l'estàndard, incorpora els paràmetres "signature_type a 9" (signatura XAdES-BES enveloped), "signature_policy", "signature_policy_hash", "signature_policy_qualifier", "signer_role" (pot prendre els valors Emisor/Receptor/Tercero) i "protectKeyInfo". Faria falta completar la signatura a la forma XAdES-T en un màxim de 3 dies des del moment de la seva generació (utilitzant la plataforma PSIS) per acabar de complir amb la normativa.

7.2.15 Exemple 15

Signatura CAdES-EPES (CAdES-BES amb política de signatura) en un PDF. El document a signar ha des ser un PDF. Es selecciona amb un diàleg i es passa a l'applet utilitzant Javascript. Per tal de que la signatura avançada sigui de la forma EPES, cal indicar la política de signatura i el hash corresponent. La signatura incorporarà un segell de temps que l'Adobe Reader podrà reconèixer. Aquesta signatura podria ser completada i reinsertada en el document PDF sense que es perdi la integritat.

7.2.16 Exemple 16

Signatura XAdES-T enveloped utilitzant els certificats d'un keystore JKS. Es signarà un XML generat amb les dades del formulari de l'HTML (tal com es fa habitualment amb l'eina FormSign). Primer el previsualitzarem i després el signarem. La signatura generada es pintarà en un dels camps del formulari. S'adjunta el JKS d'exemple al paquet de distribució.

7.2.17 Exemple 17

Signatura de varis documents PDF en una carpeta. La signatura contindrà un segell de temps, visible amb l'Adobe Reader. La carpeta amb els documents a signar es selecciona amb un diàleg i es passa a l'applet utilitzant Javascript. La sortida seran els fitxers signats ubicats a la mateixa carpeta d'origen, amb el mateix nom que l'original, afegint l'extensió "_signat.pdf". L'applet es crida utilitzant el seu propi botó, i se l'indica el text a mostrar: "Signa".

7.2.18 Exemple 18

Signatura XAdES-BES detached ~~utilitzant el certificat del clauer de CATCert~~ (***CATCert ja no dona suport ni fa desenvolupaments del clauer IDCAT per tant nosaltres tot i no haver eliminat res del codi de l'applet ja no donem suport al mateix***). El document a signar és un resum criptogràfics (hash) dels documents a signar. La signatura generada cal capturar-la utilitzant els events Javascript que genera l'applet per a la signatura. Es rep un event per a la signatura del document.

7.2.19 Exemple 19

Signatura XAdES-EPES detached. El document a signar és un resum criptogràfic (hash) del document a signar. La signatura generada es guardarà en un fitxer en local en la mateixa ubicació que el fitxer original i amb el mateix nom amb extensió "_signat". S'especificarà el signature_policy, el signature_policy_hash, el commitment_identifier i el signer_role (veure el codi HTML).

7.2.20 Exemple 20

Signatura XAdES-BES enveloped. Permet especificar les referències al/s node/s a signar. La signatura generada es guardarà en un fitxer en local a la mateixa ubicació que el fitxer original, amb el mateix nom amb extensió "_signat".

7.2.21 Exemple 21

Signatura XAdES-T enveloped. Es signarà un XML generat amb les dades del formulari de l'HTML (tal com es fa habitualment amb l'eina Formsign). Primer el previsualitzarem i després el signarem. La signatura generada es pintarà a un dels camps del formulari. El tipus de segell de la signatura serà EncapsulatedTimeStamp.

7.2.22 Exemple 22

Exemple que mostra el diàleg de selecció de certificats incorporat dins de la plana html, de manera que la tria de certificats es fa des d'un <selection> de la pròpia plana i no a través d'un pop-up. També mostra el panell de carrega durant la carrega igual que l'exemple 24. Realitza una signatura sobre un hash precalculat.

7.2.23 Exemple 23

Exemple de signatura visible en un PDF. Genera una CAdES-BES detached dins d'un PDF amb la signatura visible a totes les planes. S'ha d'indicar el paràmetre pdf_signature_rectangle passant com a número de plana -1.

7.2.24 Exemple 24

Exemple de exportació d'un certificat del magatzem configurat. Apareix el diàleg de selecció de certificat, però en comptes de generar una signatura retorna el certificat seleccionat en base64.

7.2.25 Exemple 25

Igual que l'exemple 24 però en mode de funcionament embedded. Es a dir el dialeg de selecció de certificats no apareix en un popup java sinó que està incrustat en un <selection> dins de la plana html.

7.2.26 Exemple 26

Permet realitzar la signatura sobre un PDF amb el paràmetre *pdf_dinamic_signature_image* activat de manera que el seteig de la imatge del camp de signatura es fa de forma dinàmica a la funció javascript *pdfDinamicSignatureImage.*

7.2.27 Exemple 27

Permet realitzar la signatura sobre un PDF amb 2 camps de signatura amb el paràmetre *sign_first_sign_fields* activat, de manera que a diferencia del funcionament normal només es podran triar els camps de signatura i no es podrà crear un camp nou.

7.2.28 Exemple 28

Permet realitzar una signatura sobre un PDF i recuperar el certificat del signant utilitzat per a la realització de la signatura.

7.2.29 Exemple 29

Mateix exemple que el 28, però incorpora a la configuració el paràmetre java_arguments que permet passar paràmetres de configuració per a l'arrencada de la JVM. En aquest cas es

configura la mida del heap per a poder signar documents més gran de 35Mb en mode attached. Aquest exemple es mostra per a resoldre una casuística molt concreta, ja que no és recomanable realitzar signatures attached sobre documents tant grans per tot el que comporta (mida a disc, rendiment en les validacions i tractament de la signatura etc...).

7.2.30 Exemple 30

Permet realitzar una signatura sobre un PDF amb sortida javascript quan la signatura resultant excedeix els 4MB. El cas és que s'ha hagut d'implementar una solució específica donat que en els applets la comunicació java → javascript no funciona correctament quan es passen paràmetres que accedeixen aproximadament els 4MB.

7.2.31 Taula de compatibilitat dels exemples

En les següents taules es mostren els entorns (sistema operatiu, navegador i versions de la JVM) sobre els quals s'han fet proves de funcionament dels exemples anteriors.

Windows 7 professional service pack 1 (64 bits):
*Les proves s'han realitzat sobre la versió 1.8.0_101-b13 (32 bits) de la JVM.

Test Exemples Applet	Explorer 11.0.9600.18426	**Firefox 47.0.1**
Exemple 1	OK	OK
Exemple 2	OK	OK
Exemple 3	OK	OK
Exemple 4	OK	OK
Exemple 5	OK	OK
Exemple 6	OK	OK
Exemple 7	OK	OK
Exemple 8	OK	OK
Exemple 9	OK	OK
Exemple 10	OK	OK
Exemple 11	OK	OK
Exemple 12	OK	OK
Exemple 13	OK	OK
Exemple 14	OK	OK
Exemple 15	OK	OK
Exemple 16	OK	OK
Exemple 17	OK	OK
Exemple 18	OK	OK
Exemple 19	OK	OK
Exemple 20	OK	OK
Exemple 21	OK	OK
Exemple 22	OK	OK
Exemple 23	OK	OK
Exemple 24	OK	OK
Exemple 25	OK	OK
Exemple 26	OK	OK
Exemple 27	OK	OK
Exemple 28	OK	OK
Exemple 29	OK	OK
Exemple 30	OK	OK

Ubuntu 14.04 LTS (32 bits):

*: Les proves s'han realitzat amb una JVM 1.8.0_77-b03.

Test Exemples Applet	Firefox 45.0
Exemple 1	OK
Exemple 2	OK
Exemple 3	OK
Exemple 4	OK
Exemple 5	OK
Exemple 6	OK
Exemple 7	OK
Exemple 8	OK
Exemple 9	OK
Exemple 10	OK
Exemple 11	OK
Exemple 12	OK
Exemple 13	OK
Exemple 14	OK
Exemple 15	OK
Exemple 16	OK
Exemple 17	OK
Exemple 18	OK
Exemple 19	OK
Exemple 20	OK
Exemple 21	OK
Exemple 22	OK
Exemple 23	OK
Exemple 24	OK
Exemple 25	OK
Exemple 26	OK
Exemple 27	OK
Exemple 28	OK
Exemple 29	OK
Exemple 30	OK

MAC OS X 10.8.5 (64 bits):

*: Les proves s'han realitzat amb una JVM 1.8.0_77-b03 64 bits.

**: Per a provar applets sobre html en fitxers locals amb safari s'ha d'activar el menu de developers del propi safari i desactivar les restriccions sobre fitxers locals: http://www.intertech.com/Blog/running-local-java-applets-on-a-mac/. A banda d'això després és necessari entrar a les preferencies -> seguretat -> gestionar ajustes de sitios web... i s'ha de seleccionar en el plugin java -> Documents locals -> Executar en mode NO segur.

Test Exemples Applet	Firefox 43.0.4	Safari 6.1.1**
Exemple 1	OK	OK
Exemple 2	OK	OK
Exemple 3	OK	OK
Exemple 4	OK	OK
Exemple 5	OK	OK
Exemple 6	OK	OK
Exemple 7	OK	OK
Exemple 8	OK	OK
Exemple 9	OK	OK
Exemple 10	OK	OK
Exemple 11	OK	OK
Exemple 12	OK	OK
Exemple 13	OK	OK
Exemple 14	OK	OK
Exemple 15	OK	OK
Exemple 16	OK	OK
Exemple 17	OK	OK
Exemple 18	OK	OK
Exemple 19	OK	OK
Exemple 20	OK	OK
Exemple 21	OK	OK
Exemple 22	OK	OK
Exemple 23	OK	OK
Exemple 24	OK	OK
Exemple 25	OK	OK
Exemple 26	OK	OK
Exemple 27	OK	OK
Exemple 28	OK	OK
Exemple 29	OK	OK
Exemple 30	OK	OK