



---

# Compound Request/Response Profile of the OASIS Digital Signature Service (DSS)

Working Draft 01, 27 December 2005

**Document identifier:**

**Location:**

<http://www.oasis-open.org/committees/dss>

**Editor:**

Carlos González-Cadenas, netfocus

**Contributors:**

Marta Cruellas, CATCert

Francesc Oliveras, CATCert

Ignacio Alamillo, CATCert

**Abstract:**

This document defines XML request/response protocols for requesting compound DSS-based operations to a server.

**Status:**

This is a **Working Draft** produced by the OASIS Digital Signature Service Technical Committee. Committee members should send comments on this draft to [dss@lists.oasis-open.org](mailto:dss@lists.oasis-open.org).

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Digital Signature Service TC web page at <http://www.oasis-open.org/committees/dss/ipr.php>.

---

## Table of Contents

29	1	Introduction .....	3
30	1.1	Notation .....	3
31	1.2	Schema Organization and Namespaces.....	3
32	2	Profile Features.....	4
33	2.1	Identifier.....	4
34	2.2	Scope .....	4
35	2.3	Relationship to Other Profiles .....	4
36	2.4	Signature Object.....	4
37	2.5	Transport Binding.....	5
38	2.6	Security Binding .....	5
39	3	The Compound Request/Response Protocol .....	6
40	3.1	Element <CompoundRequest>.....	6
41	3.2	Element <CompoundResponse> .....	7
42	3.3	Optional Inputs and Outputs .....	7
43	3.3.1	Common DSS Optional Inputs and Outputs.....	9
44	3.3.2	DSS Optional Inputs and Outputs related to signing operations .....	10
45	3.3.3	DSS Optional Inputs and Outputs related to verifying operations .....	11
46	3.4	Result Codes.....	11
47	4	References.....	13
48	4.1	Normative .....	13
49		Appendix A. Revision History .....	14
50		Appendix B. Notices .....	15
51			

---

## 52 1 Introduction

### 53 1.1 Notation

54 The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD",  
55 "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be  
56 interpreted as described in IETF RFC 2119 [RFC 2119]. These keywords are capitalized when  
57 used to unambiguously specify requirements over protocol features and behavior that affect the  
58 interoperability and security of implementations. When these words are not capitalized, they are  
59 meant in their natural-language sense.

60 This specification uses the following typographical conventions in text:  
61 <CompoundProfileElement>, <ns:ForeignElement>, Attribute, **Datatype**,  
62 OtherCode.

63 `Listings of Compound Profile schemas appear like this.`

### 64 1.2 Schema Organization and Namespaces

65 The structures described in this specification are contained in the schema file [**Compound-XSD**].  
66 All schema listings in the current document are excerpts from the schema file. In the case of a  
67 disagreement between the schema file and this document, the schema file takes precedence.

68 This schema is associated with the following XML namespace:

69 `urn:oasis:names:tc:dss:1.0:profiles:compound`

70 If a future version of this specification is needed, it will use a different namespace.

71 Conventional XML namespace prefixes are used in the schema:

- 72
- 73 • The prefix `dss`: stands for the DSS core namespace [**Core-XSD**].
  - 74 • The prefix `ds`: stands for the W3C XML Signature namespace [**XMLSig**].
  - 75 • The prefix `xs`: stands for the W3C XML Schema namespace [**Schema1**].
  - The prefix `compound`: or no prefix defaults to the namespace of the present document.

---

## 76 2 Profile Features

### 77 2.1 Identifier

78 urn:oasis:names:tc:dss:1.0:profiles:compound

### 79 2.2 Scope

80 At the moment, with the capabilities included in the DSS Core protocols **[DSSCore]**, only one  
81 server operation (i.e. signature production, verification or archival) can be carried per request.

82 There are some situations, mainly in batch environments, where large numbers of operations  
83 (mainly signature creations) have to be performed. Several real examples of this need can be  
84 found in different business sectors

- 85 • Invoicing, where a company generates and signs its invoices monthly, normally in a batch  
86 process.
- 87 • Electronic Procurement, where an organization generates and signs orders and other  
88 related documents normally in batch processes.
- 89 • Email Advertising, where an organization generates and signs electronic mails in batch.
- 90 • ...

91 In these cases, it would be very convenient to be able to carry multiple operations per request,  
92 leading to a more efficient (especially in network roundtrips) and manageable (the tracking of the  
93 *transactions* against the DSS server is less complicated) model.

94 This profile is concrete, can be directly implemented, and MAY be further profiled.

### 95 2.3 Relationship to Other Profiles

96 This profile is based directly on the **[DSSCore]**.

97 This profile can be combined with the Asynchronous Profile **[Async-DSS]**, especially when  
98 dealing with large requests that cannot be processed in a synchronous manner.

### 99 2.4 Signature Object

100 The signature object can only include signatures, therefore excluding other objects like  
101 timestamps and certificates.

102 **2.5 Transport Binding**

103 This profile does not constrain any transport binding defined in **[DSSCore]**.

104 **2.6 Security Binding**

105 This profile does not constrain any transport binding defined in **[DSSCore]**.

106

## 3 The Compound Request/Response Protocol

107

### 3.1 Element <CompoundRequest>

108

The <CompoundRequest> element is used to carry and wrap together the individual DSS-protocol requests (i.e. sign, verify or archive requests, among others that could be defined in the future) to the server for processing.

109

110

Clients can freely mix different protocol requests within the <CompoundRequest> element. Servers (depending of their implementation) MAY process these mixed requests, or MAY refuse them using the minor code `MixedRequestsNotSupported`.

111

112

113

Only one level of request-nesting is supported by this profile, therefore compound requests MUST not be carried inside other compound requests.

114

<dss:OptionalInputs> [Optional]

115

Optional inputs MAY be present in order to customize how the server processes the compound request, or to carry information shared to the different requests (i.e. key information).

116

117

<Requests> [Required]

118

Contains one or more individual requests (i.e. <dss:SignRequest>, <dss:VerifyRequest>, among others).

119

120

The `RequestID` and `Profile` attribute of the <CompoundRequest> element MUST be processed as described in the <dss:RequestBaseType> element of **[DSSCore]**.

121

122

The `RequestID` attribute for the inner requests MUST be present in every request included, in order to enable safe correlation between the individual requests and their responses.

123

124

Processing rules for the individual requests SHOULD follow the ones described for processing individual requests, except in those cases where a <CompoundRequest>' optional input modifies the behaviour, as described below in this document.

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

```
<xs:element name="CompoundRequest">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="dss:OptionalInputs" minOccurs="0"/>
      <xs:element name="Requests" type="dss:AnyType"/>
    </xs:sequence>
    <xs:attribute name="RequestID" type="xs:string" use="optional"/>
    <xs:attribute name="Profile" type="xs:anyURI" use="optional"/>
  </xs:complexType>
</xs:element>
```

## 141 3.2 Element <CompoundResponse>

142 The <CompoundResponse> element is used used to carry and wrap together the individual  
143 DSS-protocol responses (i.e. sign, verify or archive responses, among others that could be  
144 defined in the future) to the client after processing.

145 <Responses> [Optional]

146 Contains one or more individual requests (i.e. <dss:SignRequest>,  
147 <dss:VerifyRequest>, among others). This element MUST be present when the request  
148 is successfully processed.

149 As the <CompoundResponse> is an element whose type is derived by extension from  
150 <dss:ResponseBaseType>, processing rules described in <dss:ResponseBaseType>, as  
151 described in the element of [DSSCore], MUST be observed by the server when generating the  
152 responses, with the following additions

- 153 • The server doesn't need to order the inner responses in request order, as the  
154 RequestID of the inner requests MUST be included in all of them.
- 155 • The result of the compound response can only be Success if every inner request has  
156 also Success into its dss:ResultMajor element. Note that other conditions in the  
157 construction of the result of the compound response MAY apply (that is, we can have all  
158 the inner requests processed successfully and not have a Success result for the  
159 compound response).

```
160 <xs:element name="CompoundResponse">  
161   <xs:complexType>  
162     <xs:complexContent>  
163       <xs:extension base="dss:ResponseBaseType">  
164         <xs:sequence>  
165           <xs:element name="Responses" type="dss:AnyType" minOccurs="0"/>  
166         </xs:sequence>  
167       </xs:extension>  
168     </xs:complexContent>  
169   </xs:complexType>  
170 </xs:element>
```

## 171 3.3 Optional Inputs and Outputs

172 Apart from the optional inputs and outputs present in the inner requests and responses, the  
173 compound requests and responses MAY carry the optional inputs and outputs described below.

174 When considering a compound request, some of them are allowed to appear only in the  
175 compound request itself, some of them are allowed to appear only in the inner requests and  
176 some of them are allowed to appear both in the compound and the inner requests.

177 In order to describe these situations, this profile introduces a classification scheme for optional  
178 inputs and outputs, and provides a recommended classification under this scheme for several  
179 useful optional inputs.

180 The classification scheme is comprised by two categories

- 181 • **Exclusive Optional Inputs**, that is, these optional inputs that can't appear both in the  
182 compound and the inner requests for a given compound request.

183 • **Inclusive Optional Inputs**, that is, these optional inputs that can appear both in the  
184 compound and the inner requests for a given compound request.

185 These inclusive optional inputs MAY be further qualified using a **combining algorithm**  
186 from the ones described below. The combining algorithm determines how the server  
187 processes the requests for the  $2^2$  possible cases

188 ○ **None (default)**

189 No combination. The optional inputs are processed independently.

190 The following figure includes the four possible inputs and their outputs, where C  
191 represents the compound request, I represents a single inner request, CR  
192 represents the optional input processed by the server as the compound request's  
193 optional input and IR represents the optional input processed by the server as the  
194 inner request's optional input.

195 A represents any instance of a valid optional input. Ai and Aj represent two  
196 different instances of the optional input A.

197

C	I	CR	IR
X	A	X	A
A	X	A	X
Ai	Aj	Ai	Aj
Aj	Ai	Aj	Ai

198

199 ○ **Inherit**, that is, when the optional inputs are included in the compound request  
200 but not in the inner requests, these inner requests inherit the optional inputs from  
201 the parent one. When an optional input is present both in the compound and the  
202 inner request, this rule behaves like **None**.

C	I	CR	IR
X	A	X	A
A	X	A	A
Ai	Aj	Ai	Aj
Aj	Ai	Aj	Ai

203

204 ○ **Override**, that is, every request takes the optional input from the compound  
205 request, when present.

C	I	CR	IR
X	A	X	A

A	X	A	A
A <sub>i</sub>	A <sub>j</sub>	A <sub>i</sub>	A <sub>i</sub>
A <sub>j</sub>	A <sub>i</sub>	A <sub>j</sub>	A <sub>j</sub>

206

207

208

209

- **Union**, that is, every pair of optional inputs present in the compound and the inner requests are combined using an union operation. How the union operation behaves is dependent from the optional input in question.

C	I	CR	IR
X	A	X	A
A	X	A	A
A <sub>i</sub>	A <sub>j</sub>	A <sub>i</sub>	A <sub>i</sub> U A <sub>j</sub>
A <sub>j</sub>	A <sub>i</sub>	A <sub>j</sub>	A <sub>j</sub> U A <sub>i</sub>

210

211

212

213

- **Intersection**, that is, every pair of optional inputs present in the compound and the inner requests are combined using an intersection operation. How the intersection operation behaves is dependent from the optional input in question.

C	I	CR	IR
X	A	X	X
A	X	A	X
A <sub>i</sub>	A <sub>j</sub>	A <sub>i</sub>	A <sub>i</sub> ∩ A <sub>j</sub>
A <sub>j</sub>	A <sub>i</sub>	A <sub>j</sub>	A <sub>j</sub> ∩ A <sub>i</sub>

214

215

216

When any combination is not supported by the server, the request MUST be refused using the minor code `NotSupported`, as defined in **[DSSCore]**.

217

### 3.3.1 Common DSS Optional Inputs and Outputs

218

#### 3.3.1.1 Optional Input `<dss:ServicePolicy>`

219

As described in the `<dss:ServicePolicy>` element of **[DSSCore]**.

220

221

This optional input MUST be considered as **inclusive**, as the service policies of the compound and the inner requests MAY be different.

### 222 **3.3.1.2 Optional Input <dss:ClaimedIdentity>**

223 The presence of this optional input in a <CompoundRequest> is appropriate when an individual  
224 entity requests multiple operations to a server at a time, not needing to duplicate the  
225 <dss:ClaimedIdentity> element in every inner request, and allowing the server to perform  
226 only one authentication process that remains valid for the whole transaction.

227 There are other cases, i.e. when the requesters of the operations are not the clients themselves,  
228 but a proxy service acting on their behalf, where the approach described above MAY not be  
229 suitable. In these cases, the <dss:ClaimedIdentity> element of the <CompoundRequest>  
230 SHOULD not be present, in favour of the <dss:ClaimedIdentity> elements of the inner  
231 requests.

232 Therefore, this optional input MUST be considered as **exclusive**.

### 233 **3.3.1.3 Optional Input <dss:Language>**

234 As described in the <dss:Language> element of [DSSCore].

235 The <dss:Language> element included in the compound request and the ones included in the  
236 inner requests MAY be different. This is specially useful when a proxy service is creating the  
237 compound request in behalf of several individual clients.

238 This optional input MUST be considered as inclusive. It's up to the implementations to use one of  
239 the **None** or **Inherit** combining algorithms.

### 240 **3.3.1.4 Optional Input <dss:AdditionalProfile>**

241 As described in the <dss:AdditionalProfile> element of [DSSCore].

242 This optional input MUST be considered as **inclusive**, as the compound and/or the inner  
243 requests MAY need different additional profiles to work properly.

244 This profile can be used in combination with the Asynchronous profile [Async-DSS], in order to  
245 correctly process large batch-style transactions.

### 246 **3.3.1.5 Optional Input <dss:Schemas>**

247 As described in the <dss:Schemas> element of [DSSCore].

248 In order to minimize duplicate schema elements (and therefore minimize the length of the  
249 request), it's recommended to consider this optional input as **exclusive**, and to include the  
250 schemas in the compound request.

## 251 **3.3.2 DSS Optional Inputs and Outputs related to signing operations**

252 The following optional inputs and outputs SHOULD be considered when the compound request  
253 contain one or more signature creation operations. If these optional inputs are present, and no  
254 <dss:SignRequest> elements are present, the server MAY decide to ignore them or to refuse  
255 the request using the minor code `NotSupported`.

### 256 **3.3.2.1 Optional Input <dss:KeySelector>**

257 As described in the <dss:KeySelector> element of [DSSCore].

258 The presence of this optional input in a `<CompoundRequest>` is appropriate when an individual  
259 entity requests multiple signing operations to a server at a time using the same key (i.e. a  
260 company signing invoices with its private key).

261 In other cases (i.e. several entities signing through a proxy service or a single entity signing with  
262 different keys), these approach MAY not be suitable, and the individual `<dss:KeySelector>`  
263 elements from the individual requests SHOULD be used instead of the one of the compound  
264 request.

265 Therefore, this optional input MUST be considered as **exclusive**.

### 266 **3.3.2.2 Optional Input `<dss:SignatureType>`**

267 As described in the `<dss:SignatureType>` element of [DSSCore].

268 The presence of this optional input in a `<CompoundRequest>` is appropriate when an individual  
269 entity requests multiple signing operations to a server. In these cases, the desired signature type  
270 will be frequently the same.

271 In other cases (i.e. several entities signing through a proxy service or a single entity signing with  
272 different keys), these approach MAY not be suitable, and the individual `<dss:SignatureType>`  
273 elements from the individual requests SHOULD be used instead of the one of the compound  
274 request.

275 Therefore, this optional input MUST be considered as **exclusive**.

## 276 **3.3.3 DSS Optional Inputs and Outputs related to verifying operations**

### 277 **3.3.3.1 Optional Input `<dss:AdditionalKeyInfo>`**

278 As described in the `<dss:AdditionalKeyInfo>` element of [DSSCore].

279 The presence of this optional input in a `<CompoundRequest>` may be very useful to carry  
280 information that are common to the majority or all the requests, like intermediate CA Certificates.

281 Additionally, there are another information whose scope is tied to a specific individual request (i.e.  
282 CRLs or OCSP responses). In these cases, the `<dss:AdditionalKeyInfo>` element of the  
283 inner request SHOULD be used.

284 Therefore, this optional input MUST be considered as **inclusive**, using the **Union** combining  
285 algorithm.

286 The union operation takes every element included in the optional input of the compound request  
287 and every element in the one included in the individual request and combines them eliminating  
288 duplicates (if any).

## 289 **3.4 Result Codes**

290 The URN used for the `<dss:ResultMajor>` elements is described in [DSSCore]. The URN  
291 used for the `<dss:ResultMinor>` elements MUST be  
292 `urn:oasis:names:tc:dss:1.0:profiles:compound:resultminor:` followed by the  
293 codes described below.

<dss:ResultMajor>	<dss:ResultMinor>	Description
ResponderError	MixedRequestsNotSupported	The server does not support mixed compound requests (i.e. compound requests whose inner requests are of different types)

294

---

295 **4 References**

296 **4.1 Normative**

297 [TO BE DONE]

298

299

300

301

302

---

## Appendix A. Revision History

Rev	Date	By Whom	What
wd01	26/12/2005	Carlos González-Cadenas	Initial Version

304

---

## Appendix B. Notices

305 OASIS takes no position regarding the validity or scope of any intellectual property or other rights  
306 that might be claimed to pertain to the implementation or use of the technology described in this  
307 document or the extent to which any license under such rights might or might not be available;  
308 neither does it represent that it has made any effort to identify any such rights. Information on  
309 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS  
310 website. Copies of claims of rights made available for publication and any assurances of licenses  
311 to be made available, or the result of an attempt made to obtain a general license or permission  
312 for the use of such proprietary rights by implementors or users of this specification, can be  
313 obtained from the OASIS Executive Director.

314 OASIS invites any interested party to bring to its attention any copyrights, patents or patent  
315 applications, or other proprietary rights which may cover technology that may be required to  
316 implement this specification. Please address the information to the OASIS Executive Director.

317 Copyright © OASIS Open 2003. *All Rights Reserved.*

318 This document and translations of it may be copied and furnished to others, and derivative works  
319 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,  
320 published and distributed, in whole or in part, without restriction of any kind, provided that the  
321 above copyright notice and this paragraph are included on all such copies and derivative works.  
322 However, this document itself does not be modified in any way, such as by removing the  
323 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS  
324 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual  
325 Property Rights document must be followed, or as required to translate it into languages other  
326 than English.

327 The limited permissions granted above are perpetual and will not be revoked by OASIS or its  
328 successors or assigns.

329 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
330 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO  
331 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY  
332 RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A  
333 PARTICULAR PURPOSE.

334