

Modus: Alle Zeilen
Linker Ursprungsordner: W:\BCAST\OMA\CRs\own\OMA-IOP-TTCN-2008-0059-CR_Registration\modified TTCN-3 sources
Rechter Ursprungsordner: W:\BCAST\OMA\CRs\own\OMA-IOP-TTCN-2008-0059-CR_Registration\orig

Datei: AtsBCast_Main_Functions.ttcn

1	/**	=	1	/**
2	*		2	*
3	* @author		3	* @author
4	* ETSI CTI BCAST CON Project		4	* ETSI CTI BCAST CON Project
5	* @version		5	* @version
6	* \$Id\$		6	* \$Id\$
7	* @desc		7	* @desc
8	* This module specifies functions which create either BCAST control or		8	* This module specifies functions which create either BCAST control or
9	* Upper Tester components and start library BCAST or Upper Tester functions		9	* Upper Tester components and start library BCAST or Upper Tester functions
10	* on them.		10	* on them.
11	*/		11	*/
12	module AtsBCast_Main_Functions {		12	module AtsBCast_Main_Functions {
13	import from LibBCast_Common_TypesAndValues all;		13	import from LibBCast_Common_TypesAndValues all;
14	import from LibBCast_ServiceGuide_TypesAndValues all;		14	import from LibBCast_ServiceGuide_TypesAndValues all;
15	import from AtsBCast_ServiceGuide_Functions all;		15	import from AtsBCast_ServiceGuide_Functions all;
16	import from LibBCast_Common_Templates all;		16	import from LibBCast_Common_Templates all;
17	import from LibBCast_ServicePrimitives_TypesAndValues all;		17	import from LibBCast_ServicePrimitives_TypesAndValues all;
18	import from LibBCast_UpperTester_Functions all;		18	import from LibBCast_UpperTester_Functions all;
19	import from AtsBCast_TestSystem all;		19	import from AtsBCast_TestSystem all;
20	import from AtsBCast_ModuleParameters all;		20	import from AtsBCast_ModuleParameters all;
21	import from LibBCast_ModuleParameters all;		21	import from LibBCast_ModuleParameters all;
22	import from LibCommon_Time all;		22	import from LibCommon_Time all;
23	import from LibCommon_BasicTypesAndValues all;		23	import from LibCommon_BasicTypesAndValues all;
24	import from LibBCast_UpperTesterPrimitives_TypesAndValues all;		24	import from LibBCast_UpperTesterPrimitives_TypesAndValues all;
25	import from LibBCast_BCastNetworkControl_Functions all;		25	import from LibBCast_BCastNetworkControl_Functions all;
26	import from LibBCast_BCastNetworkData_Functions all;		26	import from LibBCast_BCastNetworkData_Functions all;
27	import from LibBCast_UpperTester_Functions all;		27	import from LibBCast_UpperTester_Functions all;
28	// change 1 (WK18): for content protection testst)		28	// change 1 (WK18): for content protection testst)
29	// change 2 (WK23): GBA authentication/authorization extensions for content protection tests		29	// change 2 (WK23): GBA authentication/authorization extensions for content protection tests
30	import from LibCommon_GBA_BSF {		30	import from LibCommon_GBA_BSF {
31	function BSF_ServerSimulation;		31	function BSF_ServerSimulation;
32	function f_bsf_setBtid;		32	function f_bsf_setBtid;
33	function f_bsf_init;		33	function f_bsf_init;
34	}		34	}
35	// change 1 (WK18): for content protection tests		35	// change 1 (WK18): for content protection tests
36	// change 1 (WK23): Service Request extensions for content protection tests		36	// change 1 (WK23): Service Request extensions for content protection tests
37	import from LibCommon_BSM {		37	import from LibCommon_BSM {
38	function BSM_ServerSimulation;		38	function BSM_ServerSimulation;
39	function f_bsm_setProtectionKeyId;		39	function f_bsm_setProtectionKeyId;
40	function f_bsm_setBtid;		40	function f_bsm_setBtid;
41	function f_bsm_setProtectionKeysAndValues;		41	function f_bsm_setProtectionKeysAndValues;
42	}		42	}
43	import from LibCommon_GBA_BSF_TypesAndValues {		43	import from LibCommon_GBA_BSF_TypesAndValues {
44	group GBA;		44	group GBA;
45	}		45	}
46			46	
47	// change 1 (WK34): Service Protection: Mikey		47	// change 1 (WK34): Service Protection: Mikey
48	import from LibCommon_Mikey_TypesAndValues all;		48	import from LibCommon_Mikey_TypesAndValues all;
49	import from LibCommon_Mikey all;		49	import from LibCommon_Mikey all;
50			50	
51	// change 3 (WK34): SService Protection: secure streaming service		51	// change 3 (WK34): SService Protection: secure streaming service
52	import from LibCommon_DataStrings all;		52	import from LibCommon_DataStrings all;
53			53	
54	group bcastCtrlMainFunctions {		54	group bcastCtrlMainFunctions {
55			55	
56	// change 01 (WK 6): global time definition		56	// change 01 (WK 6): global time definition
57	/**		57	/**
58	*		58	*
59	* @desc		59	* @desc
60	* initialize the start and end time. The start time will be		60	* initialize the start and end time. The start time will be
61	* set to a value which is test execution plus the start offset in seconds and the		61	* set to a value which is test execution plus the start offset in seconds and the
62	* endtime will be set to a value equal to the start time plus the specified duration in		62	* endtime will be set to a value equal to the start time plus the specified duration in
63	* @param		63	* @param
64	* p_startOffset Start Offset in seconds		64	* p_startOffset Start Offset in seconds
65	* @param		65	* @param
66	* p_duration Duration in seconds between the start and the end		66	* p_duration Duration in seconds between the start and the end

Datei: AtsBCast_Main_Functions.ttcn (Fortsetzung)

67	*/	67	*/
68	function f_main_ctrl_InitStartEndTime(integer p_startOffset, integer p_duration) runs on B	68	function f_main_ctrl_InitStartEndTime(integer p_startOffset, integer p_duration) runs on B
69	v_startTime := f_getNTPTime() + p_startOffset;	69	v_startTime := f_getNTPTime() + p_startOffset;
70	v_endTime := v_startTime + p_duration;	70	v_endTime := v_startTime + p_duration;
71	vc_CTRL.start (f_ctrl_InitStartEndTime(v_startTime,v_endTime));	71	vc_CTRL.start (f_ctrl_InitStartEndTime(v_startTime,v_endTime));
72	vc_CTRL.done;	72	vc_CTRL.done;
73	vc_DATA.start (f_data_InitStartEndTime(v_startTime,v_endTime)); // change 15 (WK18	73	vc_DATA.start (f_data_InitStartEndTime(v_startTime,v_endTime)); // change 15 (WK18
74	vc_DATA.done; // change 15 (WK18	74	vc_DATA.done; // change 15 (WK18
75	}	75	}
76		76	
77	/**	77	/**
78	*	78	*
79	* @desc	79	* @desc
80	* This function creates a test component and starts a function that	80	* This function creates a test component and starts a function that
81	* waits for the receiving of a http subscription	81	* waits for the receiving of a http subscription
82	* indication.	82	* indication.
83	* @param	83	* @param
84	* p_serviceName The service name for the subscription	84	* p_serviceName The service name for the subscription
85	* @verdict	85	* @verdict
86	* pass The substription was successfull received for the given	86	* pass The substription was successfull received for the given
87	* service.	87	* service.
88	* @verdict	88	* @verdict
89	* fail A message was received which is not expected.	89	* fail A message was received which is not expected.
90	* @verdict	90	* @verdict
91	* inconc A guard timer expires.	91	* inconc A guard timer expires.
92	*/	92	*/
93	function f_main_ctrl_awaitingSubscription(charstring p_serviceName) runs on BCastMainCompor	93	function f_main_ctrl_awaitingSubscription(charstring p_serviceName) runs on BCastMainCompor
94	vc_CTRL.start (f_ctrl_awaitingHttpSubscriptionIndication(p_serviceName));	94	vc_CTRL.start (f_ctrl_awaitingHttpSubscriptionIndication(p_serviceName));
95	vc_CTRL.done;	95	vc_CTRL.done;
96	}	96	}
97		97	
98	/**	98	/**
99	*	99	*
100	* @desc	100	* @desc
101	* This function creates a test component and starts a function that	101	* This function creates a test component and starts a function that
102	* waits for a mms indication.	102	* waits for a mms indication.
103	* @param	103	* @param
104	* content the content which should be inside of the mms.	104	* content the content which should be inside of the mms.
105	* @verdict	105	* @verdict
106	* pass The indication was successfull received for the given	106	* pass The indication was successfull received for the given
107	* service.	107	* service.
108	* @verdict	108	* @verdict
109	* fail A message was received which is not expected.	109	* fail A message was received which is not expected.
110	* @verdict	110	* @verdict
111	* inconc A guard timer expires.	111	* inconc A guard timer expires.
112	*/	112	*/
113	function f_main_ctrl_awaitingMMS(charstring content) runs on BCastMainComponent {	113	function f_main_ctrl_awaitingMMS(charstring content) runs on BCastMainComponent {
114		114	
115	vc_CTRL.start (f_ctrl_awaitingMMS(content));	115	vc_CTRL.start (f_ctrl_awaitingMMS(content));
116	vc_CTRL.done;	116	vc_CTRL.done;
117	}	117	}
118		118	
119	/**	119	/**
120	*	120	*
121	* @desc	121	* @desc
122	* This function creates a test component and starts a function that	122	* This function creates a test component and starts a function that
123	* waits for a sms indication.	123	* waits for a sms indication.
124	* @param	124	* @param
125	* content the content which should be inside of the mms.	125	* content the content which should be inside of the mms.
126	* @verdict	126	* @verdict
127	* pass The indication was successfull received for the given	127	* pass The indication was successfull received for the given
128	* service.	128	* service.
129	* @verdict	129	* @verdict
130	* fail A message was received which is not expected.	130	* fail A message was received which is not expected.
131	* @verdict	131	* @verdict
132	* inconc A guard timer expires.	132	* inconc A guard timer expires.
133	*/	133	*/
134	function f_main_ctrl_awaitingSMS(charstring content) runs on BCastMainComponent {	134	function f_main_ctrl_awaitingSMS(charstring content) runs on BCastMainComponent {
135	vc_CTRL.start (f_ctrl_awaitingSMS(content));	135	vc_CTRL.start (f_ctrl_awaitingSMS(content));
136	vc_CTRL.done;	136	vc_CTRL.done;
137	}	137	}

Datei: AtsBCast_Main_Functions.ttcn (Fortsetzung)

138			138	
139	/**		139	/**
140	*		140	*
141	* @desc		141	* @desc
142	* This function creates a test component and starts a function that		142	* This function creates a test component and starts a function that
143	* broadcasts a service guide over the flute protocol		143	* broadcasts a service guide over the flute protocol
144	* @param		144	* @param
145	* p_SGDU The ServiceGuideDeliveryUnit which should be		145	* p_SGDU The ServiceGuideDeliveryUnit which should be
146	* broadcasted		146	* broadcasted
147	* @param		147	* @param
148	* p_Encoding The encoding type for compression in case that the		148	* p_Encoding The encoding type for compression in case that the
149	* service guide should be compressed otherwise this parameter		149	* service guide should be compressed otherwise this parameter
150	* should be omitted.		150	* should be omitted.
151	* @param		151	* @param
152	* p_FluteUpdateID The Flute Instance ID of the Flute Session		152	* p_FluteUpdateID The Flute Instance ID of the Flute Session
153	* @verdict		153	* @verdict
154	* pass in case the broadcast of the service guide was		154	* pass in case the broadcast of the service guide was
155	* successful		155	* successful
156	* @verdict		156	* @verdict
157	* fail in case that the broadcast was not successful		157	* fail in case that the broadcast was not successful
158	* @verdict		158	* @verdict
159	* inconc in case of the timer runs out of time		159	* inconc in case of the timer runs out of time
160	*/		160	*/
161	// CR (WK 34/2008): Flute Update Signalling		161	// CR (WK 34/2008): Flute Update Signalling
162	function f_main_ctrl_broadcastServiceGuide(ServiceGuideDeliveryUnit p_SGDU,		162	function f_main_ctrl_broadcastServiceGuide(ServiceGuideDeliveryUnit p_SGDU,
163	template charstring p_Encoding,		163	template charstring p_Encoding,
164	UInt32 p_FluteUpdateID) runs on BCastMainComponen		164	UInt32 p_FluteUpdateID) runs on BCastMainComponen
165	vc_CTRL.start (165	vc_CTRL.start (
166	f_ctrl_broadcastServiceGuide(166	f_ctrl_broadcastServiceGuide(
167	p_SGDU,		167	p_SGDU,
168	PX_SGDU_ID,		168	PX_SGDU_ID,
169	PX_SGDD_ID,		169	PX_SGDD_ID,
170	p_Encoding,		170	p_Encoding,
171	p_FluteUpdateID)		171	p_FluteUpdateID)
172);		172);
173	vc_CTRL.done;		173	vc_CTRL.done;
174	}		174	}
175			175	
176	/**		176	/**
177	*		177	*
178	* @desc		178	* @desc
179	* This functions expect a http post request in order to deliver		179	* This functions expect a http post request in order to deliver
180	* the service guide over the interaction channel.		180	* the service guide over the interaction channel.
181	* @param		181	* @param
182	* p_SGDU The Service Guide Delivery Unit to send		182	* p_SGDU The Service Guide Delivery Unit to send
183	* @verdict		183	* @verdict
184	* pass the service guide could successful delivered over the		184	* pass the service guide could successful delivered over the
185	* interaction channel		185	* interaction channel
186	* @verdict		186	* @verdict
187	* fail the service fuide could not successful deflivered.		187	* fail the service fuide could not successful deflivered.
188	* @verdict		188	* @verdict
189	* incon the guard timer runs out of time.		189	* incon the guard timer runs out of time.
190	*/		190	*/
191	function f_main_ctrl_ServiceGuideOnRequest(ServiceGuideDeliveryUnit p_SGDU) runs on BCastMa		191	function f_main_ctrl_ServiceGuideOnRequest(ServiceGuideDeliveryUnit p_SGDU) runs on BCastMa
192			192	
193	vc_UTC.start(f_ut_GetServiceGuide(PX_SGDD_ID));		193	vc_UTC.start(f_ut_GetServiceGuide(PX_SGDD_ID));
194			194	
195	vc_CTRL.start (195	vc_CTRL.start (
196	f_ctrl_ServiceGuideViaHttp(196	f_ctrl_ServiceGuideViaHttp(
197	p_SGDU,		197	p_SGDU,
198	PX_SGDU_ID,		198	PX_SGDU_ID,
199	PX_SGDD_ID));		199	PX_SGDD_ID));
200	vc_CTRL.done;		200	vc_CTRL.done;
201			201	
202	vc_UTC.done;		202	vc_UTC.done;
203	}		203	}
204			204	
205	// change 1 (WK18): for content protection tests		205	// change 1 (WK18): for content protection tests
206	// change 1 (WK46/08): Clerical naming changes	+-		
207	/**	=	206	/**
208	*		207	*

Datei: AtsBCast_Main_Functions.ttcn (Fortsetzung)

209	* @desc		208	* @desc
210	* This function starts a BSF Server including a HTTP Digest challenge.		209	* This function starts a BSF Server including a HTTP Digest challenge.
211	* The server is implemented as parallel component.		210	* The server is implemented as parallel component.
212	* @verdict		211	* @verdict
213	* pass HTTP Digest (client response value matches XRES) challenge is successful.		212	* pass HTTP Digest (client response value matches XRES) challenge is successful.
214	* @verdict		213	* @verdict
215	* fail client response value does not match XRES		214	* fail client response value does not match XRES
216	* @verdict		215	* @verdict
217	* incon the guard timer runs out of time.		216	* incon the guard timer runs out of time.
218	*/		217	*/
219	function f_main_ctrl_GBA_Bootstrapping(boolean p_isParallelComponent) runs on BCastMainComp	<>	218	function f_main_ctrl_GBA_Bootstrapping(boolean p_isParallelComponent) runs on BCastMainCo
220	vc_BSF.start(BSF_ServerSimulation(p_isParallelComponent));	=	219	vc_BSF.start(BSF_ServerSimulation(p_isParallelComponent));
221	if (not p_isParallelComponent) {		220	if (not p_isParallelComponent) {
222	vc_BSF.done;		221	vc_BSF.done;
223	}		222	}
224	}		223	}
225			224	
226	// change 1 (WK23): Service Request extensions for content protection tests		225	// change 1 (WK23): Service Request extensions for content protection tests
227	/**		226	/**
228	* @desc		227	* @desc
229	* This function starts a BSM Server including a HTTP Digest challenge.		228	* This function starts a BSM Server including a HTTP Digest challenge.
230	* (The server could be implemented as parallel component.)		229	* (The server could be implemented as parallel component.)
231	* @verdict		230	* @verdict
232	* pass HTTP Digest (client response value matches XRES) challenge is successful.		231	* pass HTTP Digest (client response value matches XRES) challenge is successful.
233	* @verdict		232	* @verdict
234	* fail client response value does not match XRES		233	* fail client response value does not match XRES
235	* @verdict		234	* @verdict
236	* incon the guard timer runs out of time.		235	* incon the guard timer runs out of time.
237	*/		236	*/
238	function f_main_crtl_BSM_Procedures(boolean p_activateUserServiceRegistration, boolean p_Us		237	function f_main_crtl_BSM_Procedures(boolean p_activateUserServiceRegistration, boolean p_Us
239	if (p_activateUserServiceRegistration or p_activateServiceRequest) {		238	if (p_activateUserServiceRegistration or p_activateServiceRequest) {
240	vc_BSM.start(BSM_ServerSimulation(p_activateUserServiceRegistration, p_UsrA		239	vc_BSM.start(BSM_ServerSimulation(p_activateUserServiceRegistration, p_UsrA
241	if ((not p_UsrActAsParallelComponent) or (not p_SrActAsParallelComponent))		240	if ((not p_UsrActAsParallelComponent) or (not p_SrActAsParallelComponent))
242	vc_BSM.done;		241	vc_BSM.done;
243	}		242	}
244	}		243	}
245	}		244	}
246			245	
247	// change 2 (WK23): GBA authentication/authorization extensions for content protection test		246	// change 2 (WK23): GBA authentication/authorization extensions for content protection test
248	/**		247	/**
249	* @desc This function sets the protection key id in the BSM component.		248	* @desc This function sets the protection key id in the BSM component.
250	* @param p_protectionKeyId		249	* @param p_protectionKeyId
251	* @verdict		250	* @verdict
252	*/		251	*/
253	function f_main_crtl_SetProtectionKeyId(hexstring p_protectionKeyId) runs on BCastMainComp		252	function f_main_crtl_SetProtectionKeyId(hexstring p_protectionKeyId) runs on BCastMainComp
254	vc_BSM.start(f_bsm_setProtectionKeyId(p_protectionKeyId));		253	vc_BSM.start(f_bsm_setProtectionKeyId(p_protectionKeyId));
255	vc_BSM.done;		254	vc_BSM.done;
256	}		255	}
257			256	
258	// change 2 (WK23): GBA authentication/authorization extensions for content protection test		257	// change 2 (WK23): GBA authentication/authorization extensions for content protection test
259	/**		258	/**
260	* @desc Sets the B-TID in the BSM and BSF component.		259	* @desc Sets the B-TID in the BSM and BSF component.
261	*/		260	*/
262	function f_main_crtl_SetBootstrappingTransactionId(charstring p_btid) runs on BCastMainComp		261	function f_main_crtl_SetBootstrappingTransactionId(charstring p_btid) runs on BCastMainComp
263	vc_BSM.start(f_bsm_setBtid(p_btid));		262	vc_BSM.start(f_bsm_setBtid(p_btid));
264	vc_BSM.done;		263	vc_BSM.done;
265	vc_BSF.start(f_bsf_setBtid(p_btid));		264	vc_BSF.start(f_bsf_setBtid(p_btid));
266	vc_BSF.done;		265	vc_BSF.done;
267	}		266	}
268			267	
269	//change 2 (WK23): GBA authentication/authorization extensions for content protection tests		268	//change 2 (WK23): GBA authentication/authorization extensions for content protection tests
270	/**		269	/**
271	* @desc This method generates the KS_(ext/int)_NAF and sets this keys the in BSM and BSF		270	* @desc This method generates the KS_(ext/int)_NAF and sets this keys the in BSM and BSF
272	* @return the KS_(ext)_NAF or KS_(int)_NAF depending on the GBA type.		271	* @return the KS_(ext)_NAF or KS_(int)_NAF depending on the GBA type.
273	* @verdict		272	* @verdict
274	*/		273	*/
275	function f_main_crtl_InitProtectionKeysAndValues() runs on BCastMainComponent return octets		274	function f_main_crtl_InitProtectionKeysAndValues() runs on BCastMainComponent return octets
276	var octetstring ks_ext_NAF;		275	var octetstring ks_ext_NAF;
277	var octetstring ks_int_NAF;		276	var octetstring ks_int_NAF;
278	vc_BSF.start(f_bsf_init(ks_ext_NAF, ks_int_NAF));		277	vc_BSF.start(f_bsf_init(ks_ext_NAF, ks_int_NAF));
279	vc_BSF.done;		278	vc_BSF.done;

Datei: AtsBCast_Main_Functions.ttcn (Fortsetzung)

280	vc_BSM.start(f_bsm_setProtectionKeysAndValues(ks_ext_NAF));	279	vc_BSM.start(f_bsm_setProtectionKeysAndValues(ks_ext_NAF));
281	vc_BSM.done;	280	vc_BSM.done;
282		281	
283	if (PX_GBA == e_gba_u) {	282	if (PX_GBA == e_gba_u) {
284	// if UICC supports MBMS as well	283	// if UICC supports MBMS as well
285	return ks_int_NAF;	284	return ks_int_NAF;
286	// TODO implements 'else if' for support of MBMS	285	// TODO implements 'else if' for support of MBMS
287	}	286	}
288	else {	287	else {
289	// GBA_ME case: returns ks_NAF (equal to ke_ext_NAF)	288	// GBA_ME case: returns ks_NAF (equal to ke_ext_NAF)
290	return ks_ext_NAF;	289	return ks_ext_NAF;
291	}	290	}
292		291	
293	}	292	}
294		293	
295	// change 1 (WK34): Service Protection: Mikey (definition, sending, receiving ...)	294	// change 1 (WK34): Service Protection: Mikey (definition, sending, receiving ...)
296	/**	295	/**
297	* @desc This function sends any MIKEY message via UDP.	296	* @desc This function sends any MIKEY message via UDP.
298	* @param p_mikeyMessage	297	* @param p_mikeyMessage
299	* @verdict	298	* @verdict
300	*/	299	*/
301	function f_main_crtl_sendMikey(MikeyMessage p_mikeyMessage) runs on BCastMainComponent {	300	function f_main_crtl_sendMikey(MikeyMessage p_mikeyMessage) runs on BCastMainComponent {
302	vc_Mikey.start(f_send_MikeyMessage(p_mikeyMessage));	301	vc_Mikey.start(f_send_MikeyMessage(p_mikeyMessage));
303	vc_Mikey.done;	302	vc_Mikey.done;
304	}	303	}
305		304	
306	// change 1 (WK34): Service Protection: Mikey (definition, sending, receiving ...)	305	// change 1 (WK34): Service Protection: Mikey (definition, sending, receiving ...)
307	/**	306	/**
308	* @desc This function sends any MIKEY message to a multicast IP address.	307	* @desc This function sends any MIKEY message to a multicast IP address.
309	* @param p_mikeyMessage	308	* @param p_mikeyMessage
310	* @verdict	309	* @verdict
311	*/	310	*/
312	function f_main_crtl_broadcastMikey(MikeyMessage p_mikeyMessage) runs on BCastMainComponent	311	function f_main_crtl_broadcastMikey(MikeyMessage p_mikeyMessage) runs on BCastMainComponent
313	vc_Mikey.start(f_broadcast_MikeyMessage(p_mikeyMessage));	312	vc_Mikey.start(f_broadcast_MikeyMessage(p_mikeyMessage));
314	vc_Mikey.done;	313	vc_Mikey.done;
315	}	314	}
316	}	315	}
317		316	
318	group bcastDataMainFunctions {	317	group bcastDataMainFunctions {
319	// CR (WK 34/2008): Flute Update Signalling	318	// CR (WK 34/2008): Flute Update Signalling
320	/**	319	/**
321	*	320	*
322	* @desc This function trigger the file server to deliver files to the terminal.	321	* @desc This function trigger the file server to deliver files to the terminal.
323	* @param p_TransferId the transfer id	322	* @param p_TransferId the transfer id
324	* @param p_FluteUpdateID The Flute Instance ID of the Flute Session.	323	* @param p_FluteUpdateID The Flute Instance ID of the Flute Session.
325	* @param p_Sdp the used sdp	324	* @param p_Sdp the used sdp
326	* @param p_FileList the list of files which should delivered to the terminal	325	* @param p_FileList the list of files which should delivered to the terminal
327	* @verdict	326	* @verdict
328	* pass In case that the file transfer request was successful	327	* pass In case that the file transfer request was successful
329	* @verdict	328	* @verdict
330	* fail A wrong message was received.	329	* fail A wrong message was received.
331	* @verdict	330	* @verdict
332	* inconc A guard timer expires.	331	* inconc A guard timer expires.
333	*/	332	*/
334	function f_main_data_startFileTransfer(UInt p_TransferId,	333	function f_main_data_startFileTransfer(UInt p_TransferId,
335	UInt32 p_FluteUpdateID,	334	UInt32 p_FluteUpdateID,
336	charstring p_Sdp,	335	charstring p_Sdp,
337	TransferFileList p_FileList) runs on BCastMainComponent	336	TransferFileList p_FileList) runs on BCastMainComponent
338	vc_DATA.start (f_data_startFileTransfer(p_TransferId, p_FluteUpdateID, p_Sdp, p_FileList));	337	vc_DATA.start (f_data_startFileTransfer(p_TransferId, p_FluteUpdateID, p_Sdp, p_FileList));
339	vc_DATA.done;	338	vc_DATA.done;
340	}	339	}
341		340	
342	/**	341	/**
343	*	342	*
344	* @desc This function triggers the file server to stop a file transfer operation	343	* @desc This function triggers the file server to stop a file transfer operation
345	* @param p_TransferId the id of the transfer to be stopped	344	* @param p_TransferId the id of the transfer to be stopped
346	* @verdict	345	* @verdict
347	* pass in case that the stop streaming request was successful	346	* pass in case that the stop streaming request was successful
348	* @verdict	347	* @verdict
349	* fail A wrong message was received.	348	* fail A wrong message was received.
350	* @verdict	349	* @verdict

Datei: AtsBCast_Main_Functions.ttcn (Fortsetzung)

351	* inconc A guard timer expires.	350	* inconc A guard timer expires.
352	*/	351	*/
353	function f_main_data_stopFileTransfer(UInt p_TransferId) runs on BCastMainComponent {	352	function f_main_data_stopFileTransfer(UInt p_TransferId) runs on BCastMainComponent {
354	vc_DATA.start (f_data_stopFileTransfer(p_TransferId));	353	vc_DATA.start (f_data_stopFileTransfer(p_TransferId));
355	vc_DATA.done;	354	vc_DATA.done;
356	}	355	}
357		356	
358	// change 14 (WK18): dynamic setting of destination IP	357	// change 14 (WK18): dynamic setting of destination IP
359	/**	358	/**
360	*	359	*
361	* @desc This function triggers the streaming server to stream a file.	360	* @desc This function triggers the streaming server to stream a file.
362	* @param p_StreamId The stream id	361	* @param p_StreamId The stream id
363	* @param p_ContentList The list of files which should be streamed	362	* @param p_ContentList The list of files which should be streamed
364	* @param p_SDP The used SDP	363	* @param p_SDP The used SDP
365	* @param p_DstIp The destination IP address	364	* @param p_DstIp The destination IP address
366	* @verdict	365	* @verdict
367	* pass In case that the streaming request was successful	366	* pass In case that the streaming request was successful
368	* @verdict	367	* @verdict
369	* fail A wrong message was received.	368	* fail A wrong message was received.
370	* @verdict	369	* @verdict
371	* inconc A guard timer expires.	370	* inconc A guard timer expires.
372	*/	371	*/
373	function f_main_data_startStreamingFile(372	function f_main_data_startStreamingFile(
374	UInt p_StreamId,	373	UInt p_StreamId,
375	charstring p_FileName,	374	charstring p_FileName,
376	charstring p_SDP,	375	charstring p_SDP,
377	charstring p_DstIP) runs on BCastMainComponent {	376	charstring p_DstIP) runs on BCastMainComponent {
378	vc_DATA.start (f_data_startStreaming(p_StreamId, p_FileName, p_SDP, p_DstIP));	377	vc_DATA.start (f_data_startStreaming(p_StreamId, p_FileName, p_SDP, p_DstIP));
379	vc_DATA.done;	378	vc_DATA.done;
380	}	379	}
381		380	
382	/**	381	/**
383	*	382	*
384	* @desc This function trigger the streaming server to stop the streaming of a file	383	* @desc This function trigger the streaming server to stop the streaming of a file
385	* @param p_StreamId the id of the stream which should stopped	384	* @param p_StreamId the id of the stream which should stopped
386	* @verdict	385	* @verdict
387	* pass in case that the stop streaming request was successful	386	* pass in case that the stop streaming request was successful
388	* @verdict	387	* @verdict
389	* fail A wrong message was received.	388	* fail A wrong message was received.
390	* @verdict	389	* @verdict
391	* inconc A guard timer expires.	390	* inconc A guard timer expires.
392	*/	391	*/
393	function f_main_data_stopStreaming(UInt p_StreamId) runs on BCastMainComponent {	392	function f_main_data_stopStreaming(UInt p_StreamId) runs on BCastMainComponent {
394	vc_DATA.start (f_data_stopStreaming(p_StreamId));	393	vc_DATA.start (f_data_stopStreaming(p_StreamId));
395	vc_DATA.done;	394	vc_DATA.done;
396	}	395	}
397		396	
398	// change 3 (WK34): Service Protection: secure streaming service	397	// change 3 (WK34): Service Protection: secure streaming service
399	/**	398	/**
400	*	399	*
401	* @desc This function triggers the secure streaming server to stream a file.	400	* @desc This function triggers the secure streaming server to stream a file.
402	* @param p_StreamId The stream id	401	* @param p_StreamId The stream id
403	* @param p_ContentList The list of files which should be streamed	402	* @param p_ContentList The list of files which should be streamed
404	* @param p_SDP The used SDP	403	* @param p_SDP The used SDP
405	* @param p_DstIp The destination IP address	404	* @param p_DstIp The destination IP address
406	* @verdict	405	* @verdict
407	* pass In case that the streaming request was successful	406	* pass In case that the streaming request was successful
408	* @verdict	407	* @verdict
409	* fail A wrong message was received.	408	* fail A wrong message was received.
410	* @verdict	409	* @verdict
411	* inconc A guard timer expires.	410	* inconc A guard timer expires.
412	*/	411	*/
413	function f_main_data_secure_startStreamingFile(UInt p_StreamId, charstring p_FileName, char	412	function f_main_data_secure_startStreamingFile(UInt p_StreamId, charstring p_FileName, char
414	vc_DATA.start(f_data_startStreaming(p_StreamId, p_FileName, p_SDP, PX_SRC_IP));	413	vc_DATA.start(f_data_startStreaming(p_StreamId, p_FileName, p_SDP, PX_SRC_IP));
415	vc_DATA.done;	414	vc_DATA.done;
416	// encrypting service streams out the content	415	// encrypting service streams out the content
417	vc_DATA.start(f_data_secure_startStreaming(p_DstIP, p_ssrc));	416	vc_DATA.start(f_data_secure_startStreaming(p_DstIP, p_ssrc));
418	vc_DATA.done;	417	vc_DATA.done;
419	}	418	}
420		419	
421	// change 3 (WK34): Service Protection: secure streaming service	420	// change 3 (WK34): Service Protection: secure streaming service

Datei: AtsBCast_Main_Functions.ttcn (Fortsetzung)

422	/**	421	/**
423	*	422	*
424	* @desc This function triggers the sceure streaming server to stop the streaming of a file	423	* @desc This function triggers the sceure streaming server to stop the streaming of a file
425	* @param p_StreamId the id of the stream which should stopped	424	* @param p_StreamId the id of the stream which should stopped
426	* @verdict	425	* @verdict
427	* pass in case that the stop streaming request was successful	426	* pass in case that the stop streaming request was successful
428	* @verdict	427	* @verdict
429	* fail A wrong message was received.	428	* fail A wrong message was received.
430	* @verdict	429	* @verdict
431	* inconc A guard timer expires.	430	* inconc A guard timer expires.
432	*/	431	*/
433	function f_main_data_secure_stopStreaming(UInt p_StreamId) runs on BCastMainComponent {	432	function f_main_data_secure_stopStreaming(UInt p_StreamId) runs on BCastMainComponent {
434	vc_DATA.start(f_data_secure_stopStreaming());	433	vc_DATA.start(f_data_secure_stopStreaming());
435	vc_DATA.done;	434	vc_DATA.done;
436	vc_DATA.start(f_data_stopStreaming(p_StreamId));	435	vc_DATA.start(f_data_stopStreaming(p_StreamId));
437	vc_DATA.done;	436	vc_DATA.done;
438	}	437	}
439		438	
440	function f_main_data_secure_setKeys(in ListOfSTKMKeys p_keys) runs on BCastMainComponent {	439	function f_main_data_secure_setKeys(in ListOfSTKMKeys p_keys) runs on BCastMainComponent {
441	vc_DATA.start(f_data_secure_setKeys(p_keys));	440	vc_DATA.start(f_data_secure_setKeys(p_keys));
442	vc_DATA.done;	441	vc_DATA.done;
443	}	442	}
444		443	
445	function f_main_data_secure_activateKey(in SecureStreamingKey p_key2Activate, in integer p	444	function f_main_data_secure_activateKey(in SecureStreamingKey p_key2Activate, in integer p
446	vc_DATA.start(f_data_secure_activateKey(p_key2Activate, p_keyNumberInList));	445	vc_DATA.start(f_data_secure_activateKey(p_key2Activate, p_keyNumberInList));
447	vc_DATA.done;	446	vc_DATA.done;
448	}	447	}
449	}	448	}
450		449	
451	group upperTesterMainFunctions {	450	group upperTesterMainFunctions {
452		451	
453	/**	452	/**
454	*	453	*
455	* @desc	454	* @desc
456	* This function creates a test component and starts a function that	455	* This function creates a test component and starts a function that
457	* sends a select language request from the upper tester component.	456	* sends a select language request from the upper tester component.
458	* @param	457	* @param
459	* p_langType the language type (audio or text)	458	* p_langType the language type (audio or text)
460	* @param	459	* @param
461	* p_langId the country id for the specific language	460	* p_langId the country id for the specific language
462	* @verdict	461	* @verdict
463	* pass The request was successful	462	* pass The request was successful
464	* @verdict	463	* @verdict
465	* fail A wrong message was received	464	* fail A wrong message was received
466	* @verdict	465	* @verdict
467	* inconc A guard timer expires	466	* inconc A guard timer expires
468	*/	467	*/
469	function f_main_ut_SelectLanguage(LibBCast_UpperTesterPrimitives_TypesAndValues.LanguageTyp	468	function f_main_ut_SelectLanguage(LibBCast_UpperTesterPrimitives_TypesAndValues.LanguageTyp
470		469	
471	vc_UTC.start (f_ut_sendSelectLanguageRequest(p_langType, p_langId));	470	vc_UTC.start (f_ut_sendSelectLanguageRequest(p_langType, p_langId));
472	vc_UTC.done;	471	vc_UTC.done;
473		472	
474	}	473	}
475		474	
476	/**	475	/**
477	*	476	*
478	* @desc	477	* @desc
479	* This function creates a test component and starts a function that	478	* This function creates a test component and starts a function that
480	* sends an update service guide request from the upper tester component.	479	* sends an update service guide request from the upper tester component.
481	* @verdict	480	* @verdict
482	* pass The request was successful	481	* pass The request was successful
483	* @verdict	482	* @verdict
484	* fail A wrong message was received	483	* fail A wrong message was received
485	* @verdict	484	* @verdict
486	* inconc A guard timer expires	485	* inconc A guard timer expires
487	*/	486	*/
488	function f_main_ut_UpdateServiceGuide() runs on BCastMainComponent {	487	function f_main_ut_UpdateServiceGuide() runs on BCastMainComponent {
489	vc_UTC.start (f_ut_sendUpdateServiceGuideRequest());	488	vc_UTC.start (f_ut_sendUpdateServiceGuideRequest());
490	vc_UTC.done;	489	vc_UTC.done;
491		490	
492	}	491	}

Datei: AtsBCast_Main_Functions.ttcn (Fortsetzung)

493		492	
494		493	
495	/**	494	/**
496	*	495	*
497	* @desc	496	* @desc
498	* This function creates a test component and starts a function that	497	* This function creates a test component and starts a function that
499	* sends a power on request from the upper tester component	498	* sends a power on request from the upper tester component
500	* @verdict	499	* @verdict
501	* pass The request was successful	500	* pass The request was successful
502	* @verdict	501	* @verdict
503	* fail A wrong message was received	502	* fail A wrong message was received
504	* @verdict	503	* @verdict
505	* inconc A guard timer expires	504	* inconc A guard timer expires
506	*/	505	*/
507	function f_main_ut_PowerOn() runs on BCastMainComponent {	506	function f_main_ut_PowerOn() runs on BCastMainComponent {
508	vc_UTC.start (f_ut_sendPowerOnRequest());	507	vc_UTC.start (f_ut_sendPowerOnRequest());
509	vc_UTC.done;	508	vc_UTC.done;
510	}	509	}
511		510	
512	/**	511	/**
513	*	512	*
514	* @desc	513	* @desc
515	* This function creates a test component and starts a function that	514	* This function creates a test component and starts a function that
516	* sends a power off request from the upper tester component	515	* sends a power off request from the upper tester component
517	* @verdict	516	* @verdict
518	* pass The request was successful	517	* pass The request was successful
519	* @verdict	518	* @verdict
520	* fail A wrong message was received	519	* fail A wrong message was received
521	* @verdict	520	* @verdict
522	* inconc A guard timer expires	521	* inconc A guard timer expires
523	*/	522	*/
524	function f_main_ut_PowerOff() runs on BCastMainComponent {	523	function f_main_ut_PowerOff() runs on BCastMainComponent {
525	vc_UTC.start (f_ut_sendPowerOffRequest());	524	vc_UTC.start (f_ut_sendPowerOffRequest());
526	vc_UTC.done;	525	vc_UTC.done;
527	}	526	}
528		527	
529	/**	528	/**
530	*	529	*
531	* @desc	530	* @desc
532	* This function creates a test component and starts a function that	531	* This function creates a test component and starts a function that
533	* sends a run BCAST application request from the upper tester component	532	* sends a run BCAST application request from the upper tester component
534	* @verdict	533	* @verdict
535	* pass The request was successful	534	* pass The request was successful
536	* @verdict	535	* @verdict
537	* fail A wrong message was received	536	* fail A wrong message was received
538	* @verdict	537	* @verdict
539	* inconc A guard timer expires	538	* inconc A guard timer expires
540	*/	539	*/
541	function f_main_ut_RunBCastApplication() runs on BCastMainComponent {	540	function f_main_ut_RunBCastApplication() runs on BCastMainComponent {
542	vc_UTC.start (f_ut_RunBCastApplication());	541	vc_UTC.start (f_ut_RunBCastApplication());
543	vc_UTC.done;	542	vc_UTC.done;
544	}	543	}
545		544	
546		545	
547	/**	546	/**
548	*	547	*
549	* @desc	548	* @desc
550	* This function creates a test component and starts a function that	549	* This function creates a test component and starts a function that
551	* sends a clear service guide cache request from the upper	550	* sends a clear service guide cache request from the upper
552	* tester component.	551	* tester component.
553	* @verdict	552	* @verdict
554	* pass in case the clear service guide cache request was	553	* pass in case the clear service guide cache request was
555	* successful.	554	* successful.
556	* @verdict	555	* @verdict
557	* inconc case that the request was not successful or the timer	556	* inconc case that the request was not successful or the timer
558	* runs out of time.	557	* runs out of time.
559	*/	558	*/
560	function f_main_ut_ClearServiceGuide() runs on BCastMainComponent {	559	function f_main_ut_ClearServiceGuide() runs on BCastMainComponent {
561	vc_UTC.start (f_ut_ClearServiceGuideCache());	560	vc_UTC.start (f_ut_ClearServiceGuideCache());
562	vc_UTC.done;	561	vc_UTC.done;
563		562	

Datei: AtsBCast_Main_Functions.ttcn (Fortsetzung)

564	}	563	}
565		564	
566	/**	565	/**
567	*	566	*
568	* @desc	567	* @desc
569	* This function creates a test component and starts a function that	568	* This function creates a test component and starts a function that
570	* sends a service check request from the upper tester component and expects	569	* sends a service check request from the upper tester component and expects
571	* a success response back.	570	* a success response back.
572	* @param	571	* @param
573	* p_content The name of the content displayed to the user, e.g., program 1	572	* p_content The name of the content displayed to the user, e.g., program 1
574	* @verdict	573	* @verdict
575	* pass in case service check request was successful.	574	* pass in case service check request was successful.
576	* @verdict	575	* @verdict
577	* inconc case that the request was not successful or the timer	576	* inconc case that the request was not successful or the timer
578	* runs out of time.	577	* runs out of time.
579	*/	578	*/
580	function f_main_ut_CheckService(charstring p_content) runs on BCastMainComponent {	579	function f_main_ut_CheckService(charstring p_content) runs on BCastMainComponent {
581	vc_UTC.start(f_ut_CheckService(p_content));	580	vc_UTC.start(f_ut_CheckService(p_content));
582	vc_UTC.done;	581	vc_UTC.done;
583		582	
584	}	583	}
585		584	
586	/**	585	/**
587	*	586	*
588	* @desc	587	* @desc
589	* This function creates a test component and starts a function that	588	* This function creates a test component and starts a function that
590	* sends a Use Service Request from the upper tester component and expects a	589	* sends a Use Service Request from the upper tester component and expects a
591	* success response back.	590	* success response back.
592	* @param	591	* @param
593	* p_service The name of the service to be used (e.g., viewed) by	592	* p_service The name of the service to be used (e.g., viewed) by
594	* the user, e.g., TV Channel 1	593	* the user, e.g., TV Channel 1
595	* @verdict	594	* @verdict
596	* pass in case service check request was successful.	595	* pass in case service check request was successful.
597	* @verdict	596	* @verdict
598	* inconc case that the request was not successful or the timer	597	* inconc case that the request was not successful or the timer
599	* runs out of time.	598	* runs out of time.
600	*/	599	*/
601	function f_main_ut_UseService(charstring p_service) runs on BCastMainComponent {	600	function f_main_ut_UseService(charstring p_service) runs on BCastMainComponent {
602	vc_UTC.start(f_ut_UseService(p_service));	601	vc_UTC.start(f_ut_UseService(p_service));
603	vc_UTC.done;	602	vc_UTC.done;
604		603	
605	}	604	}
606		605	
607	/**	606	/**
608	*	607	*
609	* @desc	608	* @desc
610	* This function creates a test component and starts a function that	609	* This function creates a test component and starts a function that
611	* sends a Get Service Guide Request from the upper tester component and expects a	610	* sends a Get Service Guide Request from the upper tester component and expects a
612	* success response back.	611	* success response back.
613	* @param	612	* @param
614	* p_ServiceGuideIdentifier The name of the service guide to retrieved	613	* p_ServiceGuideIdentifier The name of the service guide to retrieved
615	* @verdict	614	* @verdict
616	* pass in case service check request was successful.	615	* pass in case service check request was successful.
617	* @verdict	616	* @verdict
618	* inconc case that the request was not successful or the timer	617	* inconc case that the request was not successful or the timer
619	* runs out of time.	618	* runs out of time.
620	*/	619	*/
621	function f_main_ut_GetServiceGuide(charstring p_ServiceGuideIdentifier) runs on BCastMainComponent {	620	function f_main_ut_GetServiceGuide(charstring p_ServiceGuideIdentifier) runs on BCastMainComponent {
622	vc_UTC.start(f_ut_GetServiceGuide(p_ServiceGuideIdentifier));	621	vc_UTC.start(f_ut_GetServiceGuide(p_ServiceGuideIdentifier));
623	vc_UTC.done;	622	vc_UTC.done;
624		623	
625	}	624	}
626		625	
627	/**	626	/**
628	*	627	*
629	* @desc	628	* @desc
630	* This function creates a test component and starts a function that sends a select service	629	* This function creates a test component and starts a function that sends a select service
631	* upper tester.	630	* upper tester.
632	* @param	631	* @param
633	* p_service The name of the service to be selected by the user, e.g., TV Channel 1	632	* p_service The name of the service to be selected by the user, e.g., TV Channel 1
634	* @verdict	633	* @verdict

Datei: AtsBCast_Main_Functions.ttcn (Fortsetzung)

635	<pre> * pass The request was successful 636 * @verdict 637 * fail A wrong message was received 638 * @verdict 639 * inconc A guard timer expires 640 */ 641 function f_main_ut_SelectService(charstring p_service) runs on BCastMainComponent { 642 vc_UTC.start(f_ut_SelectService(p_service)); 643 vc_UTC.done; 644 } 645 646 /** 647 * 648 * @desc 649 * This function creates a test component and starts a function that 650 * sends a check content request from the upper tester component. 651 * @param 652 * p_content The name of the content displayed to the user 653 * @param 654 * p_start Start of broadcasting of content displayed to the 655 * user, e.g., Program 1 656 * @param 657 * p_end End of broadcasting of content displayed to the user 658 * @verdict 659 * pass The request was successful 660 * @verdict 661 * fail A wrong message was received 662 * @verdict 663 * inconc A guard timer expires 664 */ 665 function f_main_ut_CheckContent(charstring p_content, template DateTime p_start, template I 666 vc_UTC.start(f_ut_CheckContent(p_content, p_start, p_end)); 667 vc_UTC.done; 668 } 669 670 /** 671 * 672 * @desc 673 * This function creates a test component and starts a function that 674 * sends a check interactivity request from the upper tester component. 675 * @param 676 * p_interactivity Name of the interactivity displayed to the user, e.g., vote 677 * @verdict 678 * pass The request was successful 679 * @verdict 680 * fail A wrong message was received 681 * @verdict 682 * inconc A guard timer expires 683 */ 684 function f_main_ut_CheckInteractivity(charstring p_interactivity) runs on BCastMainComponen 685 vc_UTC.start(f_ut_CheckInteractivity(p_interactivity)); 686 vc_UTC.done; 687 } 688 689 /** 690 * 691 * @desc 692 * This function creates a test component and starts a function that 693 * sends a select interactivity request from the upper tester component. 694 * @param 695 * p_interactivity Name of the interactivity to be selected to the user, e.g., vote 696 * @verdict 697 * pass The request was successful 698 * @verdict 699 * fail A wrong message was received 700 * @verdict 701 * inconc A guard timer expires 702 */ 703 function f_main_ut_SelectInteractivity(charstring p_interactivity) runs on BCastMainComponen 704 vc_UTC.start(f_ut_SelectInteractivity(p_interactivity));</pre>	634	<pre> * pass The request was successful 635 * @verdict 636 * fail A wrong message was received 637 * @verdict 638 * inconc A guard timer expires 639 */ 640 function f_main_ut_SelectService(charstring p_service) runs on BCastMainComponent { 641 vc_UTC.start(f_ut_SelectService(p_service)); 642 vc_UTC.done; 643 } 644 645 /** 646 * 647 * @desc 648 * This function creates a test component and starts a function that 649 * sends a check content request from the upper tester component. 650 * @param 651 * p_content The name of the content displayed to the user 652 * @param 653 * p_start Start of broadcasting of content displayed to the 654 * user, e.g., Program 1 655 * @param 656 * p_end End of broadcasting of content displayed to the user 657 * @verdict 658 * pass The request was successful 659 * @verdict 660 * fail A wrong message was received 661 * @verdict 662 * inconc A guard timer expires 663 */ 664 function f_main_ut_CheckContent(charstring p_content, template DateTime p_start, template I 665 vc_UTC.start(f_ut_CheckContent(p_content, p_start, p_end)); 666 vc_UTC.done; 667 } 668 669 /** 670 * 671 * @desc 672 * This function creates a test component and starts a function that 673 * sends a check interactivity request from the upper tester component. 674 * @param 675 * p_interactivity Name of the interactivity displayed to the user, e.g., vote 676 * @verdict 677 * pass The request was successful 678 * @verdict 679 * fail A wrong message was received 680 * @verdict 681 * inconc A guard timer expires 682 */ 683 function f_main_ut_CheckInteractivity(charstring p_interactivity) runs on BCastMainComponen 684 vc_UTC.start(f_ut_CheckInteractivity(p_interactivity)); 685 vc_UTC.done; 686 } 687 688 /** 689 * 690 * @desc 691 * This function creates a test component and starts a function that 692 * sends a select interactivity request from the upper tester component. 693 * @param 694 * p_interactivity Name of the interactivity to be selected to the user, e.g., vote 695 * @verdict 696 * pass The request was successful 697 * @verdict 698 * fail A wrong message was received 699 * @verdict 700 * inconc A guard timer expires 701 */ 702 function f_main_ut_SelectInteractivity(charstring p_interactivity) runs on BCastMainComponen 703 vc_UTC.start(f_ut_SelectInteractivity(p_interactivity));</pre>
-----	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Datei: AtsBCast_Main_Functions.ttcn (Fortsetzung)

706	vc_UTC.done;	705	vc_UTC.done;
707	}	706	}
708		707	
709	/**	708	/**
710	*	709	*
711	* @desc	710	* @desc
712	* This function creates a test component and starts a function that	711	* This function creates a test component and starts a function that
713	* sends a check interactivity choices request from the upper tester component.	712	* sends a check interactivity choices request from the upper tester component.
714	* @param	713	* @param
715	* p_choices Name of the interactivity choices displayed to the	714	* p_choices Name of the interactivity choices displayed to the
716	* user	715	* user
717	* @verdict	716	* @verdict
718	* pass The request was successful	717	* pass The request was successful
719	* @verdict	718	* @verdict
720	* fail A wrong message was received	719	* fail A wrong message was received
721	* @verdict	720	* @verdict
722	* inconc A guard timer expires	721	* inconc A guard timer expires
723	*/	722	*/
724	function f_main_ut_CheckInteractivityChoices(LanguageStringList p_ChoiceTexts) runs on BCas	723	function f_main_ut_CheckInteractivityChoices(LanguageStringList p_ChoiceTexts) runs on BCas
725	vc_UTC.start(f_ut_CheckInteractivityChoice(p_ChoiceTexts));	724	vc_UTC.start(f_ut_CheckInteractivityChoice(p_ChoiceTexts));
726	vc_UTC.done;	725	vc_UTC.done;
727	}	726	}
728		727	
729	/**	728	/**
730	*	729	*
731	* @desc	730	* @desc
732	* This function creates a test component and starts a function that	731	* This function creates a test component and starts a function that
733	* sends a select interactivity coice request to	732	* sends a select interactivity coice request to
734	* the upper tester.	733	* the upper tester.
735	* @param	734	* @param
736	* p_choice Name of the interactivity choice to be selected by the	735	* p_choice Name of the interactivity choice to be selected by the
737	* user, e.g., choice a	736	* user, e.g., choice a
738	* @verdict	737	* @verdict
739	* pass The request was successful	738	* pass The request was successful
740	* @verdict	739	* @verdict
741	* fail A wrong message was received	740	* fail A wrong message was received
742	* @verdict	741	* @verdict
743	* inconc A guard timer expires	742	* inconc A guard timer expires
744	*/	743	*/
745	function f_main_ut_SelectInteractivityChoice(charstring p_choice) runs on BCastMainComponen	744	function f_main_ut_SelectInteractivityChoice(charstring p_choice) runs on BCastMainComponen
746	vc_UTC.start(f_ut_SelectInteractivityChoice(p_choice));	745	vc_UTC.start(f_ut_SelectInteractivityChoice(p_choice));
747	vc_UTC.done;	746	vc_UTC.done;
748	}	747	}
749		748	
750	/**	749	/**
751	*	750	*
752	* @desc	751	* @desc
753	* This function creates a test component and starts a function that	752	* This function creates a test component and starts a function that
754	* sends a check audio language request from the upper tester component.	753	* sends a check audio language request from the upper tester component.
755	* @param	754	* @param
756	* p_content The identifer of the content displayed to the user, e.g., Program 1	755	* p_content The identifer of the content displayed to the user, e.g., Program 1
757	* @param	756	* @param
758	* p_type Type of language setting to be changed by the user	757	* p_type Type of language setting to be changed by the user
759	* @param	758	* @param
760	* p_lang Language that the user shall hear	759	* p_lang Language that the user shall hear
761	* @verdict	760	* @verdict
762	* pass The request was successful	761	* pass The request was successful
763	* @verdict	762	* @verdict
764	* fail A wrong message was received	763	* fail A wrong message was received
765	* @verdict	764	* @verdict
766	* inconc A guard timer expires	765	* inconc A guard timer expires
767	*/	766	*/
768	function f_main_ut_CheckLanguage(charstring p_content, LibBCast_UpperTesterPrimitives_Types	767	function f_main_ut_CheckLanguage(charstring p_content, LibBCast_UpperTesterPrimitives_Types
769	vc_UTC.start(f_ut_CheckLanguage(p_content, p_type, p_lang));	768	vc_UTC.start(f_ut_CheckLanguage(p_content, p_type, p_lang));
770	vc_UTC.done;	769	vc_UTC.done;
771	}	770	}
772		771	
773	/**	772	/**
774	*	773	*
775	* @desc	774	* @desc
776	* This function creates a test component and starts a function that	775	* This function creates a test component and starts a function that

Datei: AtsBCast_Main_Functions.ttcn (Fortsetzung)

777	<pre> * sends a check browser request from the upper tester component.</pre>	776	<pre> * sends a check browser request from the upper tester component.</pre>
778	<pre> * @param</pre>	777	<pre> * @param</pre>
779	<pre> * p_xhtml Reference or description of the content of the xhtml</pre>	778	<pre> * p_xhtml Reference or description of the content of the xhtml</pre>
780	<pre> * file to be displayed to the user</pre>	779	<pre> * file to be displayed to the user</pre>
781	<pre> * @verdict</pre>	780	<pre> * @verdict</pre>
782	<pre> * pass The request was successful</pre>	781	<pre> * pass The request was successful</pre>
783	<pre> * @verdict</pre>	782	<pre> * @verdict</pre>
784	<pre> * fail A wrong message was received</pre>	783	<pre> * fail A wrong message was received</pre>
785	<pre> * @verdict</pre>	784	<pre> * @verdict</pre>
786	<pre> * inconc A guard timer expires</pre>	785	<pre> * inconc A guard timer expires</pre>
787	<pre> */</pre>	786	<pre> */</pre>
788	<pre>function f_main_ut_CheckBrowser(charstring p_xhtml) runs on BCastMainComponent {</pre>	787	<pre>function f_main_ut_CheckBrowser(charstring p_xhtml) runs on BCastMainComponent {</pre>
789	<pre> vc_UTC.start(f_ut_CheckBrowser(p_xhtml));</pre>	788	<pre> vc_UTC.start(f_ut_CheckBrowser(p_xhtml));</pre>
790	<pre> vc_UTC.done;</pre>	789	<pre> vc_UTC.done;</pre>
791	<pre>}</pre>	790	<pre>}</pre>
792		791	
793	<pre>/**</pre>	792	<pre>/**</pre>
794	<pre> *</pre>	793	<pre> *</pre>
795	<pre> * @desc</pre>	794	<pre> * @desc</pre>
796	<pre> * This function creates a test component and starts a function that</pre>	795	<pre> * This function creates a test component and starts a function that</pre>
797	<pre> * sends a check preview request from the upper tester component.</pre>	796	<pre> * sends a check preview request from the upper tester component.</pre>
798	<pre> * @param</pre>	797	<pre> * @param</pre>
799	<pre> * p_data Reference to icon or text to be displayed to the user, e.g., TV Channel logo</pre>	798	<pre> * p_data Reference to icon or text to be displayed to the user, e.g., TV Channel logo</pre>
800	<pre> * @verdict</pre>	799	<pre> * @verdict</pre>
801	<pre> * pass The request was successful</pre>	800	<pre> * pass The request was successful</pre>
802	<pre> * @verdict</pre>	801	<pre> * @verdict</pre>
803	<pre> * fail A wrong message was received</pre>	802	<pre> * fail A wrong message was received</pre>
804	<pre> * @verdict</pre>	803	<pre> * @verdict</pre>
805	<pre> * inconc A guard timer expires</pre>	804	<pre> * inconc A guard timer expires</pre>
806	<pre> */</pre>	805	<pre> */</pre>
807	<pre>function f_main_ut_CheckPreview(charstring p_data) runs on BCastMainComponent {</pre>	806	<pre>function f_main_ut_CheckPreview(charstring p_data) runs on BCastMainComponent {</pre>
808	<pre> vc_UTC.start(f_ut_CheckPreview(p_data));</pre>	807	<pre> vc_UTC.start(f_ut_CheckPreview(p_data));</pre>
809	<pre> vc_UTC.done;</pre>	808	<pre> vc_UTC.done;</pre>
810	<pre>}</pre>	809	<pre>}</pre>
811		810	
812	<pre>/**</pre>	811	<pre>/**</pre>
813	<pre> *</pre>	812	<pre> *</pre>
814	<pre> * @desc</pre>	813	<pre> * @desc</pre>
815	<pre> * This function creates a test component and starts a function that</pre>	814	<pre> * This function creates a test component and starts a function that</pre>
816	<pre> * sends a select file request from the upper tester component.</pre>	815	<pre> * sends a select file request from the upper tester component.</pre>
817	<pre> * @param</pre>	816	<pre> * @param</pre>
818	<pre> * p_file File name to be selected by the user, e.g., foo.c</pre>	817	<pre> * p_file File name to be selected by the user, e.g., foo.c</pre>
819	<pre> * @verdict</pre>	818	<pre> * @verdict</pre>
820	<pre> * pass The request was successful</pre>	819	<pre> * pass The request was successful</pre>
821	<pre> * @verdict</pre>	820	<pre> * @verdict</pre>
822	<pre> * fail A wrong message was received</pre>	821	<pre> * fail A wrong message was received</pre>
823	<pre> * @verdict</pre>	822	<pre> * @verdict</pre>
824	<pre> * inconc A guard timer expires</pre>	823	<pre> * inconc A guard timer expires</pre>
825	<pre> */</pre>	824	<pre> */</pre>
826	<pre>function f_main_ut_SelectFile(charstring p_file) runs on BCastMainComponent {</pre>	825	<pre>function f_main_ut_SelectFile(charstring p_file) runs on BCastMainComponent {</pre>
827	<pre> vc_UTC.start(f_ut_SelectFile(p_file));</pre>	826	<pre> vc_UTC.start(f_ut_SelectFile(p_file));</pre>
828	<pre> vc_UTC.done;</pre>	827	<pre> vc_UTC.done;</pre>
829	<pre>}</pre>	828	<pre>}</pre>
830		829	
831	<pre>/**</pre>	830	<pre>/**</pre>
832	<pre> *</pre>	831	<pre> *</pre>
833	<pre> * @desc</pre>	832	<pre> * @desc</pre>
834	<pre> * This function creates a test component and starts a function that</pre>	833	<pre> * This function creates a test component and starts a function that</pre>
835	<pre> * sends a check file request from the upper tester component.</pre>	834	<pre> * sends a check file request from the upper tester component.</pre>
836	<pre> * @param</pre>	835	<pre> * @param</pre>
837	<pre> * p_file Reference to file name or description of its content</pre>	836	<pre> * p_file Reference to file name or description of its content</pre>
838	<pre> * to be displayed to the user, e.g., foo.c</pre>	837	<pre> * to be displayed to the user, e.g., foo.c</pre>
839	<pre> * @verdict</pre>	838	<pre> * @verdict</pre>
840	<pre> * pass The request was successful</pre>	839	<pre> * pass The request was successful</pre>
841	<pre> * @verdict</pre>	840	<pre> * @verdict</pre>
842	<pre> * fail A wrong message was received</pre>	841	<pre> * fail A wrong message was received</pre>
843	<pre> * @verdict</pre>	842	<pre> * @verdict</pre>
844	<pre> * inconc A guard timer expires</pre>	843	<pre> * inconc A guard timer expires</pre>
845	<pre> */</pre>	844	<pre> */</pre>
846	<pre>function f_main_ut_CheckFile(charstring p_file) runs on BCastMainComponent {</pre>	845	<pre>function f_main_ut_CheckFile(charstring p_file) runs on BCastMainComponent {</pre>
847	<pre> vc_UTC.start(f_ut_CheckFile(p_file));</pre>	846	<pre> vc_UTC.start(f_ut_CheckFile(p_file));</pre>

Datei: AtsBCast_Main_Functions.ttcn (Fortsetzung)

```
848         vc_UTC.done;
849     }
850
851     /**
852     *
853     * @desc
854     *     This function creates a test component and starts a function that
855     *     sends a check file received request from the upper tester component.
856     * @param
857     *     p_file File name or description of its content that is to be received
858     *     by the user, e.g., foo.c
859     * @verdict
860     *     pass The request was successful
861     * @verdict
862     *     fail A wrong message was received
863     * @verdict
864     *     inconc A guard timer expires
865     */
866     function f_main_ut_CheckFileReceived(charstring p_file) runs on BCastMainComponent {
867         vc_UTC.start(f_ut_CheckFileReceived(p_file));
868         vc_UTC.done;
869     }
870
871     /**
872     *
873     * @desc
874     *     This function creates a test component and starts a function that
875     *     sends a check video request from the upper tester component.
876     * @param
877     *     p_video Reference to video name or description of its content
878     *     to be displayed to the user, e.g., starwars
879     * @verdict
880     *     pass The request was successful
881     * @verdict
882     *     fail A wrong message was received
883     * @verdict
884     *     inconc A guard timer expires
885     */
886     function f_main_ut_CheckVideo(charstring p_video) runs on BCastMainComponent {
887         vc_UTC.start(f_ut_CheckVideo(p_video));
888         vc_UTC.done;
889     }
890
891     /**
892     *
893     * @desc
894     *     This function creates a test component and starts a function that
895     *     sends a check purchase info request from the upper tester component.
896     * @param
897     *     p_info The purchase info content to be displayed to the user
898     * @verdict
899     *     pass The request was successful
900     * @verdict
901     *     fail A wrong message was received
902     * @verdict
903     *     inconc A guard timer expires
904     */
905     function f_main_ut_CheckPurchaseInfo(charstring p_info) runs on BCastMainComponent {
906         vc_UTC.start(f_ut_CheckPurchaseInfo(p_info));
907         vc_UTC.done;
908     }
909
910     /**
911     *
912     * @desc
913     *     This function creates a test component and starts a function that
914     *     sends a purchase service request from the upper tester component.
915     * @param
916     *     p_service The name of the service to be purchased by the user, e.g., TV Channel 1
917     * @verdict
918     *     pass The request was successful
```

```
847         vc_UTC.done;
848     }
849
850     /**
851     *
852     * @desc
853     *     This function creates a test component and starts a function that
854     *     sends a check file received request from the upper tester component.
855     * @param
856     *     p_file File name or description of its content that is to be received
857     *     by the user, e.g., foo.c
858     * @verdict
859     *     pass The request was successful
860     * @verdict
861     *     fail A wrong message was received
862     * @verdict
863     *     inconc A guard timer expires
864     */
865     function f_main_ut_CheckFileReceived(charstring p_file) runs on BCastMainComponent {
866         vc_UTC.start(f_ut_CheckFileReceived(p_file));
867         vc_UTC.done;
868     }
869
870     /**
871     *
872     * @desc
873     *     This function creates a test component and starts a function that
874     *     sends a check video request from the upper tester component.
875     * @param
876     *     p_video Reference to video name or description of its content
877     *     to be displayed to the user, e.g., starwars
878     * @verdict
879     *     pass The request was successful
880     * @verdict
881     *     fail A wrong message was received
882     * @verdict
883     *     inconc A guard timer expires
884     */
885     function f_main_ut_CheckVideo(charstring p_video) runs on BCastMainComponent {
886         vc_UTC.start(f_ut_CheckVideo(p_video));
887         vc_UTC.done;
888     }
889
890     /**
891     *
892     * @desc
893     *     This function creates a test component and starts a function that
894     *     sends a check purchase info request from the upper tester component.
895     * @param
896     *     p_info The purchase info content to be displayed to the user
897     * @verdict
898     *     pass The request was successful
899     * @verdict
900     *     fail A wrong message was received
901     * @verdict
902     *     inconc A guard timer expires
903     */
904     function f_main_ut_CheckPurchaseInfo(charstring p_info) runs on BCastMainComponent {
905         vc_UTC.start(f_ut_CheckPurchaseInfo(p_info));
906         vc_UTC.done;
907     }
908
909     /**
910     *
911     * @desc
912     *     This function creates a test component and starts a function that
913     *     sends a purchase service request from the upper tester component.
914     * @param
915     *     p_service The name of the service to be purchased by the user, e.g., TV Channel 1
916     * @verdict
917     *     pass The request was successful
```

Datei: AtsBCast_Main_Functions.ttcn (Fortsetzung)

919	* @verdict	918	* @verdict
920	* fail A wrong message was received	919	* fail A wrong message was received
921	* @verdict	920	* @verdict
922	* inconc A guard timer expires	921	* inconc A guard timer expires
923	*/	922	*/
924	function f_main_ut_PurchaseService(charstring p_service) runs on BCastMainComponent {	923	function f_main_ut_PurchaseService(charstring p_service) runs on BCastMainComponent {
925	vc_UTC.start(f_ut_CheckPurchaseInfo(p_service));	924	vc_UTC.start(f_ut_CheckPurchaseInfo(p_service));
926	vc_UTC.done;	925	vc_UTC.done;
927	}	926	}
928		927	
929		928	
930		929	
931	}	930	}
932	}	931	}

Datei: AtsBCast_ModuleParameters.ttcn

1	/**	=	1	/**
2	*		2	*
3	* @author		3	* @author
4	* ETSI CTI BCAST CON Project		4	* ETSI CTI BCAST CON Project
5	* @version		5	* @version
6	* \$Id\$		6	* \$Id\$
7	* @desc		7	* @desc
8	* This module specifies ATS specific module parameters and their default values		8	* This module specifies ATS specific module parameters and their default values
9	* which can be still modified in their value as late as just prior to		9	* which can be still modified in their value as late as just prior to
10	* test case execution. The parameters in this file OMA BCAST specific.		10	* test case execution. The parameters in this file OMA BCAST specific.
11	* Module parameters realize so called PIXIT.		11	* Module parameters realize so called PIXIT.
12	*/		12	*/
13	module AtsBCast_ModuleParameters {		13	module AtsBCast_ModuleParameters {
14	import from LibBCast_UpperTesterPrimitives_TypesAndValues all;		14	import from LibBCast_UpperTesterPrimitives_TypesAndValues all;
15	import from LibCommon_TextStrings {		15	import from LibCommon_TextStrings {
16	const c_CRLF		16	const c_CRLF
17	}		17	}
18	// change 1 (WK18): for content protection tests		18	// change 1 (WK18): for content protection tests
19	// hint: for GBS authentication parameter		19	// hint: for GBS authentication parameter
20	import from LibCommon_DataStrings {		20	import from LibCommon_DataStrings {
21	group bitStringSubTypes		21	group bitStringSubTypes
22	}		22	}
23			23	
24	import from LibCommon_DataStrings_Extended {		24	import from LibCommon_DataStrings_Extended {
25	group hexStringSubTypes		25	group hexStringSubTypes
26	}		26	}
27	// change 2 (WK23): GBA authentication/authorization extensions for content protection tests		27	// change 2 (WK23): GBA authentication/authorization extensions for content protection tests
28	import from LibCommon_GBA_BSF_TypesAndValues {		28	import from LibCommon_GBA_BSF_TypesAndValues {
29	group GBA		29	group GBA
30	}		30	}
31			31	
32	group testCaseSelectionSwitches {		32	group testCaseSelectionSwitches {
33	/**		33	/**
34	*		34	*
35	* @desc Specifies to execute all the BCAST test cases		35	* @desc Specifies to execute all the BCAST test cases
36	*/		36	*/
37	modulepar boolean PX_ALL_TCS := true;		37	modulepar boolean PX_ALL_TCS := true;
38			38	
39	/**		39	/**
40	*		40	*
41	* @desc Specifies to execute all the Service Provisioning test cases		41	* @desc Specifies to execute all the Service Provisioning test cases
42	*/		42	*/
43	modulepar boolean PX_ALL_SP_TCS := false;		43	modulepar boolean PX_ALL_SP_TCS := false;
44			44	
45	/**		45	/**
46	*		46	*
47	* @desc Specifies to execute all the Service Guide test cases		47	* @desc Specifies to execute all the Service Guide test cases
48	*/		48	*/
49	modulepar boolean PX_ALL_SG_TCS := false;		49	modulepar boolean PX_ALL_SG_TCS := false;
50			50	
51	/**		51	/**
52	*		52	*
53	* @desc Specifies to execute all the File and Stream Distribution test cases		53	* @desc Specifies to execute all the File and Stream Distribution test cases
54	*/		54	*/
55	modulepar boolean PX_ALL_FD_TCS := false;		55	modulepar boolean PX_ALL_FD_TCS := false;
56			56	
57	/**		57	/**
58	*		58	*
59	* @desc Specifies all Service Interactcion test cases		59	* @desc Specifies all Service Interactcion test cases
60	*/		60	*/
61	modulepar boolean PX_ALL_SI_TCS := false;		61	modulepar boolean PX_ALL_SI_TCS := false;
62			62	
63	/**		63	/**
64	*		64	*
65	* @desc Specifies to execute all the Content Protection test cases		65	* @desc Specifies to execute all the Content Protection test cases
66	*/		66	*/
67	modulepar boolean PX_ALL_CP_TCS := false;		67	modulepar boolean PX_ALL_CP_TCS := false;
68			68	
69	// change 3 (WK34): Service Protection: secure streaming service		69	// change 3 (WK34): Service Protection: secure streaming service
70	/**		70	/**
71	*		71	*

Datei: AtsBCast_ModuleParameters.ttcn (Fortsetzung)

```
72      * @desc Specifies to execute all the Service Protection test cases
73      */
74      modulepar boolean PX_ALL_SProt_TCS := false;
75  } // end group testCaseSelectionSwitches
76
77  group terminalUserApi {
78
79      /**
80      *
81      * @desc Indicates if BCAST application under test needs (manual) operator
82      * interaction in order get service guide content displayed. If set to false
83      * the application is assumed to display the service guide automatically.
84      */
85      modulepar boolean PX_GET_SG_DISPLAY_MANUALLY := false;
86
87      /**
88      *
89      * @desc Indicates if BCAST application under test needs (manual) operator
90      * interaction in order get a service guide update displayed. If set to false
91      * the application is assumed to display the service guide update automatically.
92      */
93      modulepar boolean PX_GET_SG_UPDATE_MANUALLY := false;
94
95  } // end group terminalUserApi
96
97  group ServiceGuideIds {
98
99      group ServiceFragment {
100          /**
101          *
102          * @desc
103          * Specifies the globally unique URI for the Service fragment
104          */
105          modulepar charstring PX_SGDU_SERVICE_ID :=
106              "bcast://bcast.testingtech.com/service/service_id";
107      }
108
109      group ContentFragment {
110          /**
111          *
112          * @desc
113          * Specifies the globally unique URI for the Content fragment.
114          */
115          modulepar charstring PX_SGDU_CONTENT_ID_PROG_1 :=
116              "bcast://bcast.testingtech.com/content/content_id_1";
117
118          /**
119          *
120          * @desc
121          * Specifies the globally unique URI for the Content fragment.
122          */
123          modulepar charstring PX_SGDU_CONTENT_ID_PROG_2 :=
124              "bcast://bcast.testingtech.com/content/content_id_2";
125      }
126
127  }
128
129  group ScheduleFragment {
130      /**
131      *
132      * @desc
133      * Specifies the globally unique URI for the Schedule fragment.
134      */
135      modulepar charstring PX_SGDU_SCHEDULE_ID_PROG_1 :=
136          "bcast://bcast.testingtech.com/schedule/schedule_id_1";
137
138      /**
139      *
140      * @desc
141      * Specifies the globally unique URI for the Schedule fragment.
142      */
```

```
72      * @desc Specifies to execute all the Service Protection test cases
73      */
74      modulepar boolean PX_ALL_SProt_TCS := false;
75  } // end group testCaseSelectionSwitches
76
77  group terminalUserApi {
78
79      /**
80      *
81      * @desc Indicates if BCAST application under test needs (manual) operator
82      * interaction in order get service guide content displayed. If set to false
83      * the application is assumed to display the service guide automatically.
84      */
85      modulepar boolean PX_GET_SG_DISPLAY_MANUALLY := false;
86
87      /**
88      *
89      * @desc Indicates if BCAST application under test needs (manual) operator
90      * interaction in order get a service guide update displayed. If set to false
91      * the application is assumed to display the service guide update automatically.
92      */
93      modulepar boolean PX_GET_SG_UPDATE_MANUALLY := false;
94
95  } // end group terminalUserApi
96
97  group ServiceGuideIds {
98
99      group ServiceFragment {
100          /**
101          *
102          * @desc
103          * Specifies the globally unique URI for the Service fragment
104          */
105          modulepar charstring PX_SGDU_SERVICE_ID :=
106              "bcast://bcast.testingtech.com/service/service_id";
107      }
108
109      group ContentFragment {
110          /**
111          *
112          * @desc
113          * Specifies the globally unique URI for the Content fragment.
114          */
115          modulepar charstring PX_SGDU_CONTENT_ID_PROG_1 :=
116              "bcast://bcast.testingtech.com/content/content_id_1";
117
118          /**
119          *
120          * @desc
121          * Specifies the globally unique URI for the Content fragment.
122          */
123          modulepar charstring PX_SGDU_CONTENT_ID_PROG_2 :=
124              "bcast://bcast.testingtech.com/content/content_id_2";
125      }
126
127  }
128
129  group ScheduleFragment {
130      /**
131      *
132      * @desc
133      * Specifies the globally unique URI for the Schedule fragment.
134      */
135      modulepar charstring PX_SGDU_SCHEDULE_ID_PROG_1 :=
136          "bcast://bcast.testingtech.com/schedule/schedule_id_1";
137
138      /**
139      *
140      * @desc
141      * Specifies the globally unique URI for the Schedule fragment.
142      */
```


Datei: AtsBCast_ModuleParameters.ttcn (Fortsetzung)

143	modulepar charstring PX_SGDU_SCHEDULE_ID_PROG_2 :=	143	modulepar charstring PX_SGDU_SCHEDULE_ID_PROG_2 :=
144	"bcast://bcast.testingtech.com/schedule/schedule_id_2";	144	"bcast://bcast.testingtech.com/schedule/schedule_id_2";
145	}	145	}
146		146	
147	group AccessFragment {	147	group AccessFragment {
148	/**	148	/**
149	*	149	*
150	* @desc	150	* @desc
151	* Specifies the globally unique URI for the Access fragment	151	* Specifies the globally unique URI for the Access fragment
152	*/	152	*/
153	modulepar charstring PX_SGDU_ACCESS_ID_PROG_1 :=	153	modulepar charstring PX_SGDU_ACCESS_ID_PROG_1 :=
154	"bcast://bcast.testingtech.com/access/access_id_1";	154	"bcast://bcast.testingtech.com/access/access_id_1";
155		155	
156	/**	156	/**
157	*	157	*
158	* @desc	158	* @desc
159	* Specifies the globally unique URI for the Access fragment	159	* Specifies the globally unique URI for the Access fragment
160	*/	160	*/
161	modulepar charstring PX_SGDU_ACCESS_ID_PROG_2 :=	161	modulepar charstring PX_SGDU_ACCESS_ID_PROG_2 :=
162	"bcast://bcast.testingtech.com/access/access_id_2";	162	"bcast://bcast.testingtech.com/access/access_id_2";
163	}	163	}
164		164	
165	group PreviewDataFragment {	165	group PreviewDataFragment {
166	/**	166	/**
167	*	167	*
168	* @desc	168	* @desc
169	* Specifies the globally unique URI for the PreviewData fragment	169	* Specifies the globally unique URI for the PreviewData fragment
170	*/	170	*/
171	modulepar charstring PX_SGDU_PREVIEW_DATA_ID :=	171	modulepar charstring PX_SGDU_PREVIEW_DATA_ID :=
172	"bcast://bcast.testingtech.com/previewData/previewData_id_1";	172	"bcast://bcast.testingtech.com/previewData/previewData_id_1";
173	}	173	}
174		174	
175	group PurchaseItemFragment {	175	group PurchaseItemFragment {
176	/**	176	/**
177	*	177	*
178	* @desc	178	* @desc
179	* Specifies the globally unique URI for the PurchaseItem fragment	179	* Specifies the globally unique URI for the PurchaseItem fragment
180	*/	180	*/
181	modulepar charstring PX_SGDU_PURCHASE_ITEM_ID :=	181	modulepar charstring PX_SGDU_PURCHASE_ITEM_ID :=
182	"bcast://bcast.testingtech.com/purchaseItem/purchaseItem_id_1";	182	"bcast://bcast.testingtech.com/purchaseItem/purchaseItem_id_1";
183	}	183	}
184		184	
185	group PurchaseDataFragment {	185	group PurchaseDataFragment {
186	/**	186	/**
187	*	187	*
188	* @desc	188	* @desc
189	* Specifies the globally unique URI for the PurchaseData fragment	189	* Specifies the globally unique URI for the PurchaseData fragment
190	*/	190	*/
191	modulepar charstring PX_SGDU_PURCHASE_DATA_ID :=	191	modulepar charstring PX_SGDU_PURCHASE_DATA_ID :=
192	"bcast://bcast.testingtech.com/purchaseData/purchaseData_id_1";	192	"bcast://bcast.testingtech.com/purchaseData/purchaseData_id_1";
193	}	193	}
194		194	
195	group PurchaseChannelFragment {	195	group PurchaseChannelFragment {
196	/**	196	/**
197	*	197	*
198	* @desc	198	* @desc
199	* Specifies the globally unique URI for the PurchaseChannel fragment	199	* Specifies the globally unique URI for the PurchaseChannel fragment
200	*/	200	*/
201	modulepar charstring PX_SGDU_PURCHASE_CHANNEL_ID :=	201	modulepar charstring PX_SGDU_PURCHASE_CHANNEL_ID :=
202	"bcast://bcast.testingtech.com/purchaseChannel/purchaseChannel_id_1";	202	"bcast://bcast.testingtech.com/purchaseChannel/purchaseChannel_id_1";
203	}	203	}
204		204	
205	group InteractivityDataFragment {	205	group InteractivityDataFragment {
206	/**	206	/**
207	*	207	*
208	* @desc	208	* @desc
209	* Specifies the globally unique URI for the Interactivity fragment	209	* Specifies the globally unique URI for the Interactivity fragment
210	*/	210	*/
211	modulepar charstring PX_SGDU_INTERACTIVITY_DATA_ID :=	211	modulepar charstring PX_SGDU_INTERACTIVITY_DATA_ID :=
212	"bcast://bcast.testingtech.com/interactivityData/interactivityData_id_1";	212	"bcast://bcast.testingtech.com/interactivityData/interactivityData_id_1";
213	}	213	}

Datei: AtsBCast_ModuleParameters.ttcn (Fortsetzung)

214	}	214	}
215		215	
216	group PurchasingData {	216	group PurchasingData {
217	/**	217	/**
218	*	218	*
219	* @desc	219	* @desc
220	* Specifies the PurchaseURL in the PurchaseChannel fragment	220	* Specifies the PurchaseURL in the PurchaseChannel fragment
221	*/	221	*/
222	modulepar charstring PX_PURCHASE_URL :=	222	modulepar charstring PX_PURCHASE_URL :=
223	"bcast://bcast.testingtech.com/purchaseUrl";	223	"bcast://bcast.testingtech.com/purchaseUrl";
224		224	
225	/**	225	/**
226	*	226	*
227	* @desc	227	* @desc
228	* Specifies a value for the PriceInfo element of the PurchaseItem.	228	* Specifies a value for the PriceInfo element of the PurchaseItem.
229	*/	229	*/
230	modulepar float PX_MONETARY_PRICE := 15.00;	230	modulepar float PX_MONETARY_PRICE := 15.00;
231		231	
232	/**	232	/**
233	*	233	*
234	* @desc	234	* @desc
235	* Specifies the currency associated with the PX_MONETARY_PRICE, e.g., eur	235	* Specifies the currency associated with the PX_MONETARY_PRICE, e.g., eur
236	*/	236	*/
237	modulepar charstring PX_CURRENCY := "EUR";	237	modulepar charstring PX_CURRENCY := "EUR";
238		238	
239	}	239	}
240		240	
241	group InteractivityMediaDocument {	241	group InteractivityMediaDocument {
242	/**	242	/**
243	*	243	*
244	* @desc	244	* @desc
245	* Specifies the 'id' of the globally unique URI of the InteractivityMediaDocument	245	* Specifies the 'id' of the globally unique URI of the InteractivityMediaDocument
246	*/	246	*/
247	modulepar charstring PX_MEDIA_DOCUMENT_ID :=	247	modulepar charstring PX_MEDIA_DOCUMENT_ID :=
248	"InteractivityMediaDocumentId";	248	"InteractivityMediaDocumentId";
249		249	
250	/**	250	/**
251	*	251	*
252	* @desc	252	* @desc
253	* TSpecifies the globally unique URI of the Interactivity Media Document Group	253	* TSpecifies the globally unique URI of the Interactivity Media Document Group
254	*/	254	*/
255	modulepar charstring PX_MEDIA_DOCUMENT_GROUP_ID :=	255	modulepar charstring PX_MEDIA_DOCUMENT_GROUP_ID :=
256	"InteractivityMediaDocumentGroupId";	256	"InteractivityMediaDocumentGroupId";
257		257	
258	// change 4 (WK18): according to OMA-TS-BCAST_Services 5.3.6.2	258	// change 4 (WK18): according to OMA-TS-BCAST_Services 5.3.6.2
259	/**	259	/**
260	*	260	*
261	* @desc	261	* @desc
262	* Specifies the globally unique URI of the Interactivity Media Object Group	262	* Specifies the globally unique URI of the Interactivity Media Object Group
263	*/	263	*/
264	modulepar charstring PX_MEDIA_OBJECT_GROUP_ID :=	264	modulepar charstring PX_MEDIA_OBJECT_GROUP_ID :=
265	"InteractivityMediaObjectGroupId";	265	"InteractivityMediaObjectGroupId";
266		266	
267	/**	267	/**
268	*	268	*
269	* @desc	269	* @desc
270	* Specifies the preListenIndicator of the InteractivityData fragment	270	* Specifies the preListenIndicator of the InteractivityData fragment
271	*/	271	*/
272	modulepar boolean PX_PRELISTEN_INDICATOR := false;	272	modulepar boolean PX_PRELISTEN_INDICATOR := false;
273		273	
274	/**	274	/**
275	*	275	*
276	* @desc	276	* @desc
277	* Specifies the ContentLocation of the MediaObjectSet	277	* Specifies the ContentLocation of the MediaObjectSet
278	*/	278	*/
279	modulepar charstring PX_MEDIA_OBJECT_SET_URI :=	279	modulepar charstring PX_MEDIA_OBJECT_SET_URI :=
280	"http://www.testingtech.com/mediaObjectSet.gz";	280	"http://www.testingtech.com/mediaObjectSet.gz";
281	}	281	}
282		282	
283	group SMS {	283	group SMS {
284	/**	284	/**

Datei: AtsBCast_ModuleParameters.ttcn (Fortsetzung)

285	*	285	*
286	* @desc	286	* @desc
287	* Specifies the URI of the SMS	287	* Specifies the URI of the SMS
288	*/	288	*/
289	modulepar charstring PX_SMS_URI := "http://www.openmobilealliance.org/";	289	modulepar charstring PX_SMS_URI := "http://www.openmobilealliance.org/";
290	}	290	}
291		291	
292	group MMS {	292	group MMS {
293	/**	293	/**
294	*	294	*
295	* @desc	295	* @desc
296	* Specifies the file name of a MMS template	296	* Specifies the file name of a MMS template
297	*/	297	*/
298	modulepar charstring PX_MMS_TEMPLATE := "vote.mtd";	298	modulepar charstring PX_MMS_TEMPLATE := "vote.mtd";
299		299	
300	/**	300	/**
301	*	301	*
302	* @desc	302	* @desc
303	* Specifies the file name of a picture	303	* Specifies the file name of a picture
304	*/	304	*/
305	modulepar charstring PX_MMS_VOTE_A_PICTURE := "ChoiceA.png";	305	modulepar charstring PX_MMS_VOTE_A_PICTURE := "ChoiceA.png";
306		306	
307	/**	307	/**
308	*	308	*
309	* @desc	309	* @desc
310	* Specifies the file name of a picture	310	* Specifies the file name of a picture
311	*/	311	*/
312	modulepar charstring PX_MMS_VOTE_PICTURE := "ChoiceB.jpg";	312	modulepar charstring PX_MMS_VOTE_PICTURE := "ChoiceB.jpg";
313		313	
314	/**	314	/**
315	*	315	*
316	* @desc	316	* @desc
317	* Specifies the file name of a text file	317	* Specifies the file name of a text file
318	*/	318	*/
319	modulepar charstring PX_MMS_TEXT_FILE := "Instructions.txt";	319	modulepar charstring PX_MMS_TEXT_FILE := "Instructions.txt";
320	}	320	}
321		321	
322	group XHTML {	322	group XHTML {
323	/**	323	/**
324	*	324	*
325	* @desc	325	* @desc
326	* Specifies the file name of the XHTML file	326	* Specifies the file name of the XHTML file
327	*/	327	*/
328	modulepar charstring PX_XHTML_FILE_ID := "vote-xhtml";	328	modulepar charstring PX_XHTML_FILE_ID := "vote-xhtml";
329		329	
330	/**	330	/**
331	*	331	*
332	* @desc	332	* @desc
333	* Specifies the content type of the XHTML file	333	* Specifies the content type of the XHTML file
334	*/	334	*/
335	modulepar charstring PX_XHTML_CONTENT_TYPE :=	335	modulepar charstring PX_XHTML_CONTENT_TYPE :=
336	"application/vnd.wap.xhtml+xml";	336	"application/vnd.wap.xhtml+xml";
337	}	337	}
338		338	
339	group FileDelivery {	339	group FileDelivery {
340	// change 34 (WK18): rename "TVChannelIcon" SDP session into "Data_Session", rename respect	340	// change 34 (WK18): rename "TVChannelIcon" SDP session into "Data_Session", rename respect
341	/**	341	/**
342	*	342	*
343	* @desc	343	* @desc
344	* Specifies the file name of a picture, e.g., file1.jpg	344	* Specifies the file name of a picture, e.g., file1.jpg
345	*/	345	*/
346	modulepar charstring PX_DATA_SESSION := "TvChannelIcon.jpg";	346	modulepar charstring PX_DATA_SESSION := "TvChannelIcon.jpg";
347		347	
348	/**	348	/**
349	*	349	*
350	* @desc	350	* @desc
351	* Specifies the file name containing video, e.g., video1.mov	351	* Specifies the file name containing video, e.g., video1.mov
352	*/	352	*/
353	modulepar charstring PX_FILE_VIDEO_PROG1 := "video1.mov";	353	modulepar charstring PX_FILE_VIDEO_PROG1 := "video1.mov";
354		354	
355	/**	355	/**

Datei: AtsBCast_ModuleParameters.ttcn (Fortsetzung)

356	*	356	*
357	* @desc	357	* @desc
358	* Specifies the file name containing video, e.g., video2.mov	358	* Specifies the file name containing video, e.g., video2.mov
359	*/	359	*/
360	modulepar charstring PX_FILE_VIDEO_PROG2 := "video2.mov";	360	modulepar charstring PX_FILE_VIDEO_PROG2 := "video2.mov";
361		361	
362	/**	362	/**
363	*	363	*
364	* @desc	364	* @desc
365	* Specifies the file name containing audio, e.g., audio1.mp3	365	* Specifies the file name containing audio, e.g., audio1.mp3
366	*/	366	*/
367	modulepar charstring PX_FILE_AUDIO_PROG_1 := "audio1.mp3";	367	modulepar charstring PX_FILE_AUDIO_PROG_1 := "audio1.mp3";
368		368	
369	/**	369	/**
370	*	370	*
371	* @desc	371	* @desc
372	* Specifies the file name containing audio, e.g., audio2.mp3	372	* Specifies the file name containing audio, e.g., audio2.mp3
373	*/	373	*/
374	modulepar charstring PX_FILE_AUDIO_PROG_2 := "audio2.mp3";	374	modulepar charstring PX_FILE_AUDIO_PROG_2 := "audio2.mp3";
375		375	
376	/**	376	/**
377	*	377	*
378	* @desc	378	* @desc
379	* Specifies the file name containing audio, e.g., audio_eng.mp3	379	* Specifies the file name containing audio, e.g., audio_eng.mp3
380	*/	380	*/
381	modulepar charstring PX_FILE_AUDIO_ENG := "audio_eng.mp3";	381	modulepar charstring PX_FILE_AUDIO_ENG := "audio_eng.mp3";
382		382	
383	/**	383	/**
384	*	384	*
385	* @desc	385	* @desc
386	* Specifies the file name containing audio, e.g., audio_ger.mp3	386	* Specifies the file name containing audio, e.g., audio_ger.mp3
387	*/	387	*/
388	modulepar charstring PX_FILE_AUDIO_GER := "audio_ger.mp3";	388	modulepar charstring PX_FILE_AUDIO_GER := "audio_ger.mp3";
389		389	
390	/**	390	/**
391	*	391	*
392	* @desc	392	* @desc
393	* Specifies the time in seconds for the video to play	393	* Specifies the time in seconds for the video to play
394	*/	394	*/
395	modulepar integer PX_PLAY_TIME := 300;	395	modulepar integer PX_PLAY_TIME := 300;
396		396	
397	// change 5 (WK18): update and documentation of ics/ixit settings	397	// change 5 (WK18): update and documentation of ics/ixit settings
398	/**	398	/**
399	* @desc	399	* @desc
400	* streaming server sends data to this IP address	400	* streaming server sends data to this IP address
401	*/	401	*/
402	modulepar charstring PX_DATA_STREAMING_DESTINATION_IP_1 := "232.0.3.0";	402	modulepar charstring PX_DATA_STREAMING_DESTINATION_IP_1 := "232.0.3.0";
403		403	
404	// change 5 (WK18): update and documentation of ics/ixit settings	404	// change 5 (WK18): update and documentation of ics/ixit settings
405	/**	405	/**
406	* @desc	406	* @desc
407	* streaming server sends data to this IP address	407	* streaming server sends data to this IP address
408	*/	408	*/
409	modulepar charstring PX_DATA_STREAMING_DESTINATION_IP_2 := "232.0.4.0";	409	modulepar charstring PX_DATA_STREAMING_DESTINATION_IP_2 := "232.0.4.0";
410		410	
411	}	411	}
412		412	
413	group FluteSessionParameters {	413	group FluteSessionParameters {
414		414	
415	/**	415	/**
416	*	416	*
417	* @desc	417	* @desc
418	* Specifies the IP address for the DVB-H bootstrapping session	418	* Specifies the IP address for the DVB-H bootstrapping session
419	*/	419	*/
420	modulepar charstring PX_DVB_H_FLUTE_SESSION_IP := "224.00.23.14";	420	modulepar charstring PX_DVB_H_FLUTE_SESSION_IP := "224.00.23.14";
421		421	
422	/**	422	/**
423	* @desc Specifies the port for the DVB-H bootstrapping session	423	* @desc Specifies the port for the DVB-H bootstrapping session
424	*/	424	*/
425	modulepar integer PX_DVB_H_FLUTE_SESSION_PORT := 9214;	425	modulepar integer PX_DVB_H_FLUTE_SESSION_PORT := 9214;
426		426	

Datei: AtsBCast_ModuleParameters.ttcn (Fortsetzung)

427	/**	427	/**
428	* @desc Specifies the IP address for the Flute SGDD session	428	* @desc Specifies the IP address for the Flute SGDD session
429	*/	429	*/
430	modulepar charstring PX_SGDD_FLUTE_SESSION_IP := "232.0.1.0";	430	modulepar charstring PX_SGDD_FLUTE_SESSION_IP := "232.0.1.0";
431		431	
432	/**	432	/**
433	* @desc Specifies the port for the Flute SGDD session	433	* @desc Specifies the port for the Flute SGDD session
434	*/	434	*/
435	modulepar integer PX_SGDD_FLUTE_SESSION_PORT := 9000;	435	modulepar integer PX_SGDD_FLUTE_SESSION_PORT := 9000;
436		436	
437	/**	437	/**
438	* @desc Specifies the IP address for the Flute SGDU session	438	* @desc Specifies the IP address for the Flute SGDU session
439	*/	439	*/
440	modulepar charstring PX_SGDU_FLUTE_SESSION_IP := "232.0.1.0";	440	modulepar charstring PX_SGDU_FLUTE_SESSION_IP := "232.0.1.0";
441		441	
442	/**	442	/**
443	* @desc Specifies the port for the Flute SGDU session	443	* @desc Specifies the port for the Flute SGDU session
444	*/	444	*/
445	modulepar integer PX_SGDU_FLUTE_SESSION_PORT := 9003;	445	modulepar integer PX_SGDU_FLUTE_SESSION_PORT := 9003;
446		446	
447	/**	447	/**
448	* @desc Specifies the IP address for a Flute data session	448	* @desc Specifies the IP address for a Flute data session
449	*/	449	*/
450	modulepar charstring PX_DATA_FLUTE_SESSION_IP := "232.0.7.0";	450	modulepar charstring PX_DATA_FLUTE_SESSION_IP := "232.0.7.0";
451		451	
452	/**	452	/**
453	* @desc Specifies the port for a Flute data session	453	* @desc Specifies the port for a Flute data session
454	*/	454	*/
455	modulepar integer PX_DATA_FLUTE_SESSION_PORT := 10080;	455	modulepar integer PX_DATA_FLUTE_SESSION_PORT := 10080;
456	}	456	}
457		457	
458	group SessionDescription {	458	group SessionDescription {
459	/**	459	/**
460	* @desc Specifies the ID of the SDP	460	* @desc Specifies the ID of the SDP
461	*/	461	*/
462	modulepar charstring PX_SDP_VIDEO_PROG_1_REF_ID := "myFirstSDP";	462	modulepar charstring PX_SDP_VIDEO_PROG_1_REF_ID := "myFirstSDP";
463		463	
464	/**	464	/**
465	* @desc Specifies the ID of the SDP	465	* @desc Specifies the ID of the SDP
466	*/	466	*/
467	modulepar charstring PX_SDP_VIDEO_PROG_2_REF_ID := "mySecondSDP";	467	modulepar charstring PX_SDP_VIDEO_PROG_2_REF_ID := "mySecondSDP";
468		468	
469	/**	469	/**
470	* @desc Specifies the ID of the SDP	470	* @desc Specifies the ID of the SDP
471	*/	471	*/
472	// change 34 (WK18): rename "TVChannelIcon" SDP session into "Data_Session", rename respect	472	// change 34 (WK18): rename "TVChannelIcon" SDP session into "Data_Session", rename respect
473	modulepar charstring PX_SDP_DATA_SESSION_REF_ID := "DataSession";	473	modulepar charstring PX_SDP_DATA_SESSION_REF_ID := "DataSession";
474		474	
475	/**	475	/**
476	*	476	*
477	* @desc	477	* @desc
478	Specifies an SDP with the default settings	478	Specifies an SDP with the default settings
479	*/	479	*/
480	modulepar charstring PX_SDP_VIDEO_PROG_1 :=	480	modulepar charstring PX_SDP_VIDEO_PROG_1 :=
481	"v=0\r\no=- 424 3292855200 IN IP6 FF15:0:0:0:0:81:1BC\r\ns=Unencrypted Mobile TV	481	"v=0\r\no=- 424 3292855200 IN IP6 FF15:0:0:0:0:81:1BC\r\ns=Unencrypted Mobile TV
482		482	
483	/**	483	/**
484	*	484	*
485	* @desc	485	* @desc
486	Specifies an SDP with the default settings	486	Specifies an SDP with the default settings
487	*/	487	*/
488	modulepar charstring PX_SDP_VIDEO_PROG_2 :=	488	modulepar charstring PX_SDP_VIDEO_PROG_2 :=
489	"v=0\r\no=- 424 3292855200 IN IP6 FF15:0:0:0:0:81:1BC\r\ns=Unencrypted Mobile TV	489	"v=0\r\no=- 424 3292855200 IN IP6 FF15:0:0:0:0:81:1BC\r\ns=Unencrypted Mobile TV
490		490	
491	/**	491	/**
492	*	492	*
493	* @desc	493	* @desc
494	Specifies an SDP with a particular language	494	Specifies an SDP with a particular language
495	*/	495	*/
496	modulepar charstring PX_SDP_AUDIO_ENG :=	496	modulepar charstring PX_SDP_AUDIO_ENG :=
497	"v=0\r\no=- 424 3292855200 IN IP6 FF15:0:0:0:0:81:1BC\r\ns=Unencrypted Mobile TV	497	"v=0\r\no=- 424 3292855200 IN IP6 FF15:0:0:0:0:81:1BC\r\ns=Unencrypted Mobile TV

Datei: AtsBCast_ModuleParameters.ttcn (Fortsetzung)

498		498	
499	/**	499	/**
500	*	500	*
501	* @desc	501	* @desc
502	Specifies an SDP with a particular language	502	Specifies an SDP with a particular language
503	*/	503	*/
504	modulepar charstring PX_SDP_AUDIO_GER :=	504	modulepar charstring PX_SDP_AUDIO_GER :=
505	"v=0\r\no=- 424 3292855200 IN IP6 FF15:0:0:0:0:81:1BC\r\ns=Unencrypted Mobile TV	505	"v=0\r\no=- 424 3292855200 IN IP6 FF15:0:0:0:0:81:1BC\r\ns=Unencrypted Mobile TV
506		506	
507	/**	507	/**
508	*	508	*
509	* @desc	509	* @desc
510	Specifies SDP for audio stream of TV Program 1	510	Specifies SDP for audio stream of TV Program 1
511	*/	511	*/
512	modulepar charstring PX_SDP_AUDIO_PROG_1 :=	512	modulepar charstring PX_SDP_AUDIO_PROG_1 :=
513	"v=0\r\no=- 424 3292855200 IN IP6 FF15:0:0:0:0:81:1BC\r\ns=Unencrypted Mobile TV	513	"v=0\r\no=- 424 3292855200 IN IP6 FF15:0:0:0:0:81:1BC\r\ns=Unencrypted Mobile TV
514		514	
515	/**	515	/**
516	*	516	*
517	* @desc	517	* @desc
518	Specifies SDP for audio stream of TV Program 2	518	Specifies SDP for audio stream of TV Program 2
519	*/	519	*/
520	modulepar charstring PX_SDP_AUDIO_PROG_2 :=	520	modulepar charstring PX_SDP_AUDIO_PROG_2 :=
521	"v=0\r\no=- 424 3292855200 IN IP6 FF15:0:0:0:0:81:1BC\r\ns=Unencrypted Mobile TV	521	"v=0\r\no=- 424 3292855200 IN IP6 FF15:0:0:0:0:81:1BC\r\ns=Unencrypted Mobile TV
522		522	
523	// change 34 (WK18): rename "TVChannelIcon" SDP session into "Data_Session", rename respect	523	// change 34 (WK18): rename "TVChannelIcon" SDP session into "Data_Session", rename respect
524	/**	524	/**
525	*	525	*
526	* @desc	526	* @desc
527	Specifies the SDP to be used for file and Interactivity Media Document delivery over	527	Specifies the SDP to be used for file and Interactivity Media Document delivery over
528	*/	528	*/
529	modulepar charstring PX_SDP_DATA_SESSION :=	529	modulepar charstring PX_SDP_DATA_SESSION :=
530	"v=0\r\no=user123 2890844526 2890842807 IN IP6 2201:056D::112E:144A:1E24\r\ns= Exam	530	"v=0\r\no=user123 2890844526 2890842807 IN IP6 2201:056D::112E:144A:1E24\r\ns= Exam
531		531	
532	/**	532	/**
533	*	533	*
534	* @desc	534	* @desc
535	Specifies the SDP to be used for the Content Protection test cases using IPSec, e.g.	535	Specifies the SDP to be used for the Content Protection test cases using IPSec, e.g.
536	*/	536	*/
537	modulepar charstring PX_SDP_SERVICE_PROTECTION_IPSEC :=	537	modulepar charstring PX_SDP_SERVICE_PROTECTION_IPSEC :=
538	"v=0\r\no=- 424 3292855200 IN IP6 FF15:0:0:0:0:81:1BC\r\ns=Unencrypted Mobile TV	538	"v=0\r\no=- 424 3292855200 IN IP6 FF15:0:0:0:0:81:1BC\r\ns=Unencrypted Mobile TV
539		539	
540	/**	540	/**
541	*	541	*
542	* @desc	542	* @desc
543	Specifies the SDP to be used for the Content Protection test cases using SRTP, e.g.,	543	Specifies the SDP to be used for the Content Protection test cases using SRTP, e.g.,
544	*/	544	*/
545	modulepar charstring PX_SDP_SERVICE_PROTECTION_SRTP :=	545	modulepar charstring PX_SDP_SERVICE_PROTECTION_SRTP :=
546	"v=0\r\no=- 424 3292855200 IN IP6 FF15:0:0:0:0:81:1BC\r\ns=Unencrypted Mobile TV	546	"v=0\r\no=- 424 3292855200 IN IP6 FF15:0:0:0:0:81:1BC\r\ns=Unencrypted Mobile TV
547		547	
548	/**	548	/**
549	*	549	*
550	* @desc	550	* @desc
551	Specifies the SDP to be used for the Content Protection test cases using ISMACrypt, e	551	Specifies the SDP to be used for the Content Protection test cases using ISMACrypt, e
552	*/	552	*/
553	modulepar charstring PX_SDP_SERVICE_PROTECTION_ISMA :=	553	modulepar charstring PX_SDP_SERVICE_PROTECTION_ISMA :=
554	"v=0\r\no=- 424 3292855200 IN IP6 FF15:0:0:0:0:81:1BC\r\ns=Unencrypted Mobile TV	554	"v=0\r\no=- 424 3292855200 IN IP6 FF15:0:0:0:0:81:1BC\r\ns=Unencrypted Mobile TV
555	}	555	}
556		556	
557	group Misc {	557	group Misc {
558	/**	558	/**
559	*	559	*
560	* @desc	560	* @desc
561	On true, test case runs in simulate mode.	561	On true, test case runs in simulate mode.
562	*/	562	*/
563	modulepar boolean PX_SIMULATE_TC := false;	563	modulepar boolean PX_SIMULATE_TC := false;
564	}	564	}
565		565	
566	// change 1 (WK18): for content protection tests	566	// change 1 (WK18): for content protection tests
567	group GBA {	567	group GBA {
568	/**	568	/**

Datei: AtsBCast_ModuleParameters.ttcn (Fortsetzung)

```

569      * @desc Specifies the IP Multimedia Private Identity
570    */
571 modulepar charstring PX_IMPI := "001010123456063@ims.mnc001.mcc001.pub.3gppnetwork.org";
572
573 /**
574   *
575   * @desc Specifies the Bootstrap Server Function Fully Qualified Domain Name (BSF FQDN)
576   */
577 modulepar charstring PX_BSF_FQDN := "bsf.rs.de";
578 /**
579   *
580   * @desc Specifies the Authentication / Random challenge (128 bits): min 31, max 127 (TS 34.129)
581   */
582 modulepar Bit128 PX_AuthRAND :=
583     '01010101010101010101010101010101010101010101010101010101010101010101010101010101';
584 /**
585   *
586   * @desc Specifies the Authentication Key (128 bits)
587   */
588 modulepar Bit128 PX_AuthK :=
589     '000000000000000100000010000001100000100000001010000011000000111000010000000100100';
590 /**
591   *
592   * @desc Specifies the Authentication Management Field (16 bits)
593   */
594 modulepar Bit16 PX_AuthAMF := '0000000000000000'B;
595 /**
596   *
597   * @desc Specifies the length of the Authentication Key
598   */
599 modulepar integer PX_AuthN := 127;
600 /**
601   *
602   * @desc Specifies the authentication sequence number for testing
603   */
604 modulepar Bit48 PX_AuthSQN := '0000000000000000000000000000000000000000000000000000000000000000'B;
605 // change 2 (WK23): GBA authentication/authorization extensions for content protection test
606 /**
607   *
608   * @desc Specifies the supported GBA algorithm in the DUT
609   */
610 modulepar GBAType PX_GBA := e_gba_u;
611 // change 3 (WK34): Service Protection: secure streaming service
612 /**
613   *
614   * @desc Specifies the lower bound of the STKM Timestamp
615   */
616 modulepar octetstring PX_TS_LOW := '00'O;
617 /**
618   *
619   * @desc Specifies the upper bound of the STKM Timestamp
620   */
621 modulepar octetstring PX_TS_HIGH := '0f'O;
622 /**
623   *
624   * @desc Specifies the key utilized for encrypting the key included in LTKM KEMAC
625   */
626 modulepar octetstring PX_LTKM := 'ff'O;
627 /**
628   *
629   * @desc Specifies the initial ID of the MBMS Transport Key (MTK)

```

```

569      * @desc Specifies the IP Multimedia Private Identity
570    */
571 modulepar charstring PX_IMPI := "001010123456063@ims.mnc001.mcc001.pub.3gppnetwork.org";
572
573 /**
574   *
575   * @desc Specifies the Bootstrap Server Function Fully Qualified Domain Name (BSF FQDN)
576 */
577 modulepar charstring PX_BSF_FQDN := "bsf.rs.de";
578
579 /**
580   *
581   * @desc Specifies the Authentication / Random challenge (128 bits): min 31, max 127 (TS 3GPP TS 33.102)
582 */
583 modulepar Bit128 PX_AuthRAND :=
584     '01010101010101010101010101010101010101010101010101010101010101010101010101010101';
585
586 /**
587   *
588   * @desc Specifies the Authentication Key (128 bits)
589 */
590 modulepar Bit128 PX_AuthK :=
591     '00000000000000001000000100000001100000100000001010000011000000111000010000000100100';
592
593 /**
594   *
595   * @desc Specifies the Authentication Management Field (16 bits)
596 */
597 modulepar Bit16 PX_AuthAMF := '0000000000000000'B;
598
599 /**
600   *
601   * @desc Specifies the length of the Authentication Key
602 */
603 modulepar integer PX_AuthN := 127;
604
605 /**
606   *
607   * @desc Specifies the authentication sequence number for testing
608 */
609 modulepar Bit48 PX_AuthSQN := '0000000000000000000000000000000000000000000000000000000000000000'B;
610
611 // change 2 (WK23): GBA authentication/authorization extensions for content protection test
612 /**
613   *
614   * @desc Specifies the supported GBA algorithm in the DUT
615 */
616 modulepar GBAType PX_GBA := e_gba_u;
617
618 // change 3 (WK34): Service Protection: secure streaming service
619 /**
620   *
621   * @desc Specifies the lower bound of the STKM Timestamp
622 */
623 modulepar octetstring PX_TS_LOW := '00'O;
624
625 /**
626   *
627   * @desc Specifies the upper bound of the STKM Timestamp
628 */
629 modulepar octetstring PX_TS_HIGH := '0f'O;
630
631 /**
632   *
633   * @desc Specifies the key utilized for encrypting the key included in LTKM KEMAC
634 */
635 modulepar octetstring PX_LTKM := 'ff'O;
636
637 /**
638   *
639   * @desc Specifies the initial ID of the MBMS Transport Key (MTK)
```

Datei: AtsBCast_ModuleParameters.ttcn (Fortsetzung)

640	*/		640	*/
641	modulepar octetstring PX_MTK_ID_START := '0001'O;		641	modulepar octetstring PX_MTK_ID_START := '0001'O;
642			642	
643	/**		643	/**
644	*		644	*
645	* @desc Specifies the initial key of the traffic encryption master key		645	* @desc Specifies the initial key of the traffic encryption master key
646	*/		646	*/
647	modulepar octetstring PX_SRTP_MASTER_KEY_START := '000000000000000000000000000001'O; // 1		647	modulepar octetstring PX_SRTP_MASTER_KEY_START := '000000000000000000000000000001'O; // 1
648			648	
649	/**		649	/**
650	*		650	*
651	* @desc Specifies the lenght of the traffic encryption salt key		651	* @desc Specifies the lenght of the traffic encryption salt key
652	*/		652	*/
653	modulepar integer PX_SALT_KEY_LENGTH := 112; // length in bits		653	modulepar integer PX_SALT_KEY_LENGTH := 112; // length in bits
654	}		654	}
655			655	
656	// change 1 (WK18): for content protection tests		656	// change 1 (WK18): for content protection tests
657	group BSM {		657	group BSM {
658			658	
659	/**		659	/**
660	*		660	*
661	* @desc		661	* @desc
662	* Specifies the BCAST Subscription Management (BSM) Server FQDN		662	* Specifies the BCAST Subscription Management (BSM) Server FQDN
663	*/		663	*/
664	modulepar charstring PX_BSM_FQDN := "bmsc.rs.de";		664	modulepar charstring PX_BSM_FQDN := "bmsc.rs.de";
665			665	
666	/**		666	/**
667	*		667	*
668	* @desc		668	* @desc
669	* Specifies the Mobile Country Code		669	* Specifies the Mobile Country Code
670	*/		670	*/
671	modulepar Hex3 PX_MCC := '001'H; // 1.5 bytes long		671	modulepar Hex3 PX_MCC := '001'H; // 1.5 bytes long
672			672	
673	/**		673	/**
674	*		674	*
675	* @desc		675	* @desc
676	* Specifies the Mobile Network Code		676	* Specifies the Mobile Network Code
677	*/		677	*/
678	modulepar Hex3 PX_MNC := '001'H; // 1.5 bytes long		678	modulepar Hex3 PX_MNC := '001'H; // 1.5 bytes long
679			679	
680	/**		680	/**
681	*		681	*
682	* @desc		682	* @desc
683	* Specifies the a group of SEK/PEKs that are identified by the same Key group part of		683	* Specifies the a group of SEK/PEKs that are identified by the same Key group part of
684	*/		684	*/
685	modulepar Hex4 PX_KEY_GROUP := '0001'H; // 2 bytes long		685	modulepar Hex4 PX_KEY_GROUP := '0001'H; // 2 bytes long
686			686	
687	/**		687	/**
688	*		688	*
689	* @desc		689	* @desc
690	* Specifies within a key group, which SEK/PEK is used		690	* Specifies within a key group, which SEK/PEK is used
691	*/		691	*/
692	modulepar Hex4 PX_KEY_NUMBER := '0000'H; // 2 bytes long		692	modulepar Hex4 PX_KEY_NUMBER := '0000'H; // 2 bytes long
693		+-		
694	// change 3 (WK46/08): Test if MBMS Key Management Support is available			
695	/**			
696	* @desc			
697	* Specifies if DUT supports MBMS key management			
698	*/			
699	modulepar boolean PX_SUPPORT_MBMS_KEY_MANAGEMENT := true;			
700	}	=	693	}
701			694	
702			695	
703	} // end module AtsBCast_ModuleParameters		696	} // end module AtsBCast_ModuleParameters

Datei: AtsBCast_ServiceProtectionTests.ttcn

1 // change 1 (WK18): for content protection tests	=	1 // change 1 (WK18): for content protection tests
2 module AtsBCast_ServiceProtectionTests {		2 module AtsBCast_ServiceProtectionTests {
3 import from AtsBCast_TestConfiguration_Functions all;		3 import from AtsBCast_TestConfiguration_Functions all;
4 import from AtsBCast_TestSystem all;		4 import from AtsBCast_TestSystem all;
5 import from AtsBCast_Main_Functions all;		5 import from AtsBCast_Main_Functions all;
6 import from LibBCast_ServiceGuide_TypesAndValues all;		6 import from LibBCast_ServiceGuide_TypesAndValues all;
7 import from LibCommon_BasicTypesAndValues all;		7 import from LibCommon_BasicTypesAndValues all;
8 import from LibBCast_Common_TypesAndValues all;		8 import from LibBCast_Common_TypesAndValues all;
9 import from AtsBCast_ServiceGuide_Functions all;		9 import from AtsBCast_ServiceGuide_Functions all;
10 import from LibBCast_ServiceGuide_Templates all;		10 import from LibBCast_ServiceGuide_Templates all;
11 import from AtsBCast_ModuleParameters all;		11 import from AtsBCast_ModuleParameters all;
12 import from LibBCast_ModuleParameters all;		12 import from LibBCast_ModuleParameters all;
13 import from LibCommon_GBA_BSF {		13 import from LibCommon_GBA_BSF {
14 group externalFunctions;		14 group externalFunctions;
15 function f_bsf_generateBtid; // change 2 (WK23): GBA authentication/authorization extensions for cc		15 function f_bsf_generateBtid; // change 2 (WK23): GBA authentication/authorization extensions for cc
16 }		16 }
17 // change 1 (WK34): Service Protection: Mikey (definition, sending, receiving ...)		17 // change 1 (WK34): Service Protection: Mikey (definition, sending, receiving ...)
18 import from LibCommon_Mikey_Templates all;		18 import from LibCommon_Mikey_Templates all;
19 import from LibCommon_Mikey_TypesAndValues all;		19 import from LibCommon_Mikey_TypesAndValues all;
20 import from LibCommon_Mikey all;		20 import from LibCommon_Mikey all;
21 // change 3 (WK34): Service Protection: secure streaming service		21 // change 3 (WK34): Service Protection: secure streaming service
22 import from LibBCast_ServicePrimitives_TypesAndValues {		22 import from LibBCast_ServicePrimitives_TypesAndValues {
23 group streamingServerPrimitives;		23 group streamingServerPrimitives;
24 }		24 }
25 import from LibCommon_DataStrings all;		25 import from LibCommon_DataStrings all;
26 import from LibCommon_GBA_BSF_TypesAndValues {	+-	
27 group GBA		
28 }		
29	=	26
30 group bcastConformanceTestCases {		27 group bcastConformanceTestCases {
31 group clientConformanceTestCases {		28 group clientConformanceTestCases {
32		29
33 /**		30 /**
34 *		31 *
35 * @desc		32 * @desc
36 * <p>Registration</p>		33 * <p>Registration</p>
37 * <p>Test Case Description</p>		34 * <p>Test Case Description</p>
38 * <p>When the BCAST Client is started in the terminal that initiates the		35 * <p>When the BCAST Client is started in the terminal that initiates the
39 * MBMS User Service Registration procedure.</p>		36 * MBMS User Service Registration procedure.</p>
40 * <p>Specification Reference</p>		37 * <p>Specification Reference</p>
41 * <p>[BCAST10-Services] Section 5.1.6.7</p>		38 * <p>[BCAST10-Services] Section 5.1.6.7</p>
42 * <p>Preconditions</p>		39 * <p>Preconditions</p>
43 * <p>The service guide cache of the terminal is erased.		40 * <p>The service guide cache of the terminal is erased.
44 * The value for the ServiceID field (valid concatenation of a GlobalPurchaseItemID		41 * The value for the ServiceID field (valid concatenation of a GlobalPurchaseItemID
45 * and a PurchaseDataReference) is known by the Terminal		42 * and a PurchaseDataReference) is known by the Terminal
46 * UICC contains Key management function: GBA_U and (MBMS or BCAST) key management		43 * UICC contains Key management function: GBA_U and (MBMS or BCAST) key management
47 * UICC contains credentials and algorithms according to 34.108 [NEW REF] chapter 8		44 * UICC contains credentials and algorithms according to 34.108 [NEW REF] chapter 8
48 * UICC SQN value is always constant.		45 * UICC SQN value is always constant.
49 * There is a Service Guide available. For the Service Guide instantiation details see 5.		46 * There is a Service Guide available. For the Service Guide instantiation details see 5.
50 * Can be tested at the same time as: 5.10.1.3 - Purchasing information.	<>	47 * Can be tested at the same time as: 5.10.1.3 - Purchasing information.
51 * </p>	=	48 * </p>
52 * <p>Test Procedure</p>		49 * <p>Test Procedure</p>
53 * <p>		50 * <p>
54 * 		51 *
55 * Activate the BCAST application on the terminal.		52 * Activate the BCAST application on the terminal.
56 * Terminal initiates the MBMS User Service Registration procedure with User Service		53 * Terminal initiates the MBMS User Service Registration procedure with User Service
57 * 		54 *
58 * a. The Terminal sends an initial HTTP POST (Registration indication) with		55 * a. The Terminal sends an initial HTTP POST (Registration indication) with
59 * b. Test tool replies with HTTP 401 Unauthorized response with Digest chall		56 * b. Test tool replies with HTTP 401 Unauthorized response with Digest chall
60 * c. Terminal uses MRK and B-TID derived from the valid bootstrapping sessio		57 * c. Terminal uses MRK and B-TID derived from the valid bootstrapping sessio
61 * 		58 *
62 * 		59 *
63 * The test tool replies with HTTP 200 OK.		60 * The test tool replies with HTTP 200 OK.
64 * 		61 *
65 * <p>Note1: The use of test data proposed by the [3GPP TS 35.207-700 (Implementer's Test		62 * <p>Note1: The use of test data proposed by the [3GPP TS 35.207-700 (Implementer's Test
66 * <p>Note2: In case there is no valid bootstrapping context, the terminal runs bootstrapp		63 * <p>Note2: In case there is no valid bootstrapping context, the terminal runs bootstrapp
67 * </p>		64 * </p>
68 * <p>Pass-Criteria</p>		65 * <p>Pass-Criteria</p>
69 * <p>2a. The terminal sends the Registration indication</p>		66 * <p>2a. The terminal sends the Registration indication</p>
70 * <p>2c. The second POST request is properly formatted and contains the authentication head		67 * <p>2c. The second POST request is properly formatted and contains the authentication head
71 * @verdict		68 * @verdict

Datei: AtsBCast_ServiceProtectionTests.ttcn (Fortsetzung)

72	* pass in case the client pass the conformance test.	69	* pass in case the client pass the conformance test.
73	* @verdict	70	* @verdict
74	* fail in case the client does not pass the conformance test.	71	* fail in case the client does not pass the conformance test.
75	* @verdict	72	* @verdict
76	* inconc in case a guard timer expires.	73	* inconc in case a guard timer expires.
77	*/	74	*/
78	testcase TC_BCAST_SERVPROT_101a() runs on BCastMainComponent system BCastPlatformComponent {	75	testcase TC_BCAST_SERVPROT_101a() runs on BCastMainComponent system BCastPlatformComponent {
79	// change 3 (WK46/08): Test if MBMS Key Management Support is available	<>	
80	if (PX_GBA == e_gba_u and PX_SUPPORT_MBMS_KEY_MANAGEMENT) {		
81	// init components and ports	76	// init components and ports
82	f_createComponents();	77	f_createComponents();
83	f_cf_DNS_up();	78	f_cf_DNS_up();
84	f_cf_BSM_up();	79	f_cf_BSM_up();
85	f_cf_UpperTester_up();	80	f_cf_UpperTester_up();
86		81	
87	// preamble	82	// preamble
88	f_cf_enable_DNS();	83	f_cf_enable_DNS();
89	f_startCommonUtPreamble();	84	f_startCommonUtPreamble();
90		85	
91	// change 2 (WK23): GBA authentication/authorization extensions for content protection	86	// change 2 (WK23): GBA authentication/authorization extensions for content protection test
92	f_main_ctrl_InitProtectionKeysAndValues();	87	f_main_ctrl_InitProtectionKeysAndValues();
93	f_main_ctrl_SetBootstrappingTransactionId(f_bsf_generateBtid());	88	f_main_ctrl_SetBootstrappingTransactionId(f_bsf_generateBtid());
94		89	
95	// change 1 (WK23): Service Request extensions for content protection tests	90	// change 1 (WK23): Service Request extensions for content protection tests
96	// Service Protection: simplified Service Request	91	// Service Protection: simplified Service Request
97	// Bcast Subscription Managemant as parallel test component for MBMS User Service Regis	92	// Bcast Subscription Managemant as parallel test component for MBMS User Service Registrat
98	// and Service Request/Response is not allowed	93	// and Service Request/Response is not allowed
99	f_main_ctrl_BSM_Procedures(true, false, false, false);	94	f_main_ctrl_BSM_Procedures(true, false, false, false);
100		95	
101	// postamble	96	// postamble
102	f_main_ut_PowerOff();	97	f_main_ut_PowerOff();
103		98	
104	f_cf_UpperTester_down();	99	f_cf_UpperTester_down();
105	f_cf_BSM_down();	100	f_cf_BSM_down();
106	f_cf_DNS_down();	101	f_cf_DNS_down();
107	f_terminateComponents();	102	f_terminateComponents();
108	} else {		
109	log("Verdict Info: Preconditions are not meet. See PIXIT values 'PX_GBA' and 'PX_SUPPOR		
110	setverdict(inconc);		
111	}		
112	}	=	103
113			104
114	/**		105
115	*		106
116	* @desc		107
117	* <p>GBA-U Bootstrapping USIM</p>		108
118	* <p>Test Case Description</p>		109
119	* <p>When the terminal needs to do authentication and there is no existing		110
120	* bootstrapping context that initiates the bootstrapping flow.</p>		111
121	* <p>Specification Reference</p>		112
122	* <p>[BCAST10-ServContProt] Section 6.5.1</p>		113
123	* <p>Preconditions</p>		114
124	* <p>No bootstrapping context exists between the terminal and the test tool.		115
125	* All existing credentials are marked as invalid.		116
126	* 		117
127	* The value for the ServiceID field (valid concatenation of a GlobalPurchaseItemID		118
128	* UICC contains Key management function: GBA_U and MBMS or BCAST key management.</		119
129	* 		120
130	* UICC contains credentials and algorithms according to 34.108 [NEW REF] chapter 8		121
131	* UICC SQN value is always constant.		122
132	* BSF address is set up in the terminal		123
133	* Can be tested at the same time as: 5.10.1.1 - Registration</p>		124
134	* <p>Test Procedure</p>		125
135	* <p>		126
136	* 		127
137	* Terminal retrieves from the Service Guide the permissionIssuerURI, and extracts f		128
138	* Terminal detects that a bootstrapping procedure is needed (no valid SRK available		129
139	* The Terminal runs the bootstrapping procedure		130
140	* 		131
141	* a. The Terminal sends an initial POST request (HTTP request) containing th		132
142	* b. Test tool replies with HTTP 401 Unauthorized response with Digest chall		133

Datei: AtsBCast_ServiceProtectionTests.ttcn (Fortsetzung)

143	* c. RAND and AUTN are used by USIM to generate RES authentication challenge	134	* c. RAND and AUTN are used by USIM to generate RES authentication challenge
144	* d. Test tool generates B-TID from the IMPI and sends a 200OK message inclu	135	* d. Test tool generates B-TID from the IMPI and sends a 200OK message inclu
145	* e. The terminal stores B-TID and key lifetime in the USIM EFGBABP file	136	* e. The terminal stores B-TID and key lifetime in the USIM EFGBABP file
146	* f. The terminal sends the NAF_ID received in step 1 (FQDN and UA security	137	* f. The terminal sends the NAF_ID received in step 1 (FQDN and UA security
147	* 	138	*
148	* 	139	*
149	* At this time the test tool and the USIM share the bootstrap Key material Ks_int_NA	140	* At this time the test tool and the USIM share the bootstrap Key material Ks_int_NA
150	* 	141	*
151	* </p>	142	* </p>
152	* <p>Pass-Criteria</p>	143	* <p>Pass-Criteria</p>
153	* <p>4a. The terminal sends a POST request with the appropriate IMPI.</p>	144	* <p>4a. The terminal sends a POST request with the appropriate IMPI.</p>
154	* <p>4c. RES corresponds to the XRES in the test tool.</p>	145	* <p>4c. RES corresponds to the XRES in the test tool.</p>
155	* @verdict	146	* @verdict
156	* pass in case the client pass the conformance test.	147	* pass in case the client pass the conformance test.
157	* @verdict	148	* @verdict
158	* fail in case the client does not pass the conformance test.	149	* fail in case the client does not pass the conformance test.
159	* @verdict	150	* @verdict
160	* inconc in case a guard timer expires.	151	* inconc in case a guard timer expires.
161	*/	152	*/
162	testcase TC_BCAST_SERVPROT_101b() runs on BCastMainComponent system BCastPlatformComponent {	153	testcase TC_BCAST_SERVPROT_101b() runs on BCastMainComponent system BCastPlatformComponent {
163	// change 3 (WK46/08): Test if MBMS Key Management Support is available	<>	
164	if (PX_GBA == e_gba_u and PX_SUPPORT_MBMS_KEY_MANAGEMENT) {		
165	// init components and ports	154	// init components and ports
166	f_createComponents();	155	f_createComponents();
167	f_cf_DNS_up();	156	f_cf_DNS_up();
168	f_cf_BSM_up();	157	f_cf_BSM_up();
169	f_cf_BSF_up();	158	f_cf_BSF_up();
170	f_cf_UpperTester_up();	159	f_cf_UpperTester_up();
171		160	
172	// preamble	161	// preamble
173	f_cf_enable_DNS();	162	f_cf_enable_DNS();
174	f_startCommonUtPreamble();	163	f_startCommonUtPreamble();
175		164	
176	// change 2 (WK23): GBA authentication/authorization extensions for content protection	165	// change 2 (WK23): GBA authentication/authorization extensions for content protection test
177	f_main_ctrl_InitProtectionKeysAndValues();	166	f_main_ctrl_InitProtectionKeysAndValues();
178	f_main_ctrl_SetBootstrappingTransactionId(f_bsf_generateBtid());	167	f_main_ctrl_SetBootstrappingTransactionId(f_bsf_generateBtid());
179		168	
180	// change 1 (WK23): Service Request extensions for content protection tests	169	// change 1 (WK23): Service Request extensions for content protection tests
181	// Service Protection: simplified Service Request	170	// Service Protection: simplified Service Request
182	// Bcast Subscription Managemant as parallel test component for MBMS User Service Registrat	171	// Bcast Subscription Managemant as parallel test component for MBMS User Service Registrat
183	// and Service Request/Response is not allowed	172	// and Service Request/Response is not allowed
184	f_main_ctrl_BSM_Procedures(true, true, false, true);	173	f_main_ctrl_BSM_Procedures(true, true, false, true);
185	// GBA/BSF acts as test component	174	// GBA/BSF acts as test component
186	f_main_ctrl_GBA_Bootstrapping(false);	175	f_main_ctrl_GBA_U_Bootstrapping(false);
187		176	
188	// postamble	177	// postamble
189	f_main_ut_PowerOff();	178	f_main_ut_PowerOff();
190		179	
191	f_cf_UpperTester_down();	180	f_cf_UpperTester_down();
192	f_cf_BSF_down();	181	f_cf_BSF_down();
193	f_cf_BSM_down();	182	f_cf_BSM_down();
194	f_cf_DNS_down();	183	f_cf_DNS_down();
195	f_terminateComponents();	184	f_terminateComponents();
196	} else {	=	
197	log("Verdict Info: Preconditions are not meet. See PIXIT values 'PX_GBA' and 'PX_SUPPOR		
198	setverdict(inconc);	185	}
199	}	186	
200	}	187	testcase TC_BCAST_SERVPROT_101c() runs on BCastMainComponent system BCastPlatformComponent {
201			
202	testcase TC_BCAST_SERVPROT_101c() runs on BCastMainComponent system BCastPlatformComponent {		
203	// change 3 (WK46/08): Test if MBMS Key Management Support is available	<>	
204	if (PX_GBA == e_gba_u and PX_SUPPORT_MBMS_KEY_MANAGEMENT) {		
205	// init components and ports	188	// init components and ports
206	f_createComponents();	189	f_createComponents();
207	f_cf_Broadcast_up();	190	f_cf_Broadcast_up();
208	f_cf_UpperTester_up();	191	f_cf_UpperTester_up();
209	f_cf_DNS_up();	192	f_cf_DNS_up();
210	f_cf_BSM_up();	193	f_cf_BSM_up();
211	f_cf_BSF_up();	194	f_cf_BSF_up();
212		195	
213	// preamble	196	// preamble

Datei: AtsBCast_ServiceProtectionTests.ttcn (Fortsetzung)

214	f_cf_enable_DNS();	197	f_cf_enable_DNS();
215	f_startCommonUtPreamble();	198	f_startCommonUtPreamble();
216		199	
217	// change 2 (WK23): GBA authentication/authorization extensions for content protection	200	// change 2 (WK23): GBA authentication/authorization extensions for content protection test
218	var hexstring protectionKeyId := PX_MCC & PX_MNC & PX_KEY_GROUP & PX_KEY_NUMBER;	201	var hexstring protectionKeyId := PX_MCC & PX_MNC & PX_KEY_GROUP & PX_KEY_NUMBER;
219	f_main_ctrl_InitProtectionKeysAndValues();	202	f_main_ctrl_InitProtectionKeysAndValues();
220	f_main_ctrl_SetProtectionKeyId(protectionKeyId);	203	f_main_ctrl_SetProtectionKeyId(protectionKeyId);
221	f_main_ctrl_SetBootstrappingTransactionId(f_bsf_generateBtid());	204	f_main_ctrl_SetBootstrappingTransactionId(f_bsf_generateBtid());
222	// GBA/BSF as parallel test component	205	// GBA/BSF as parallel test component
223	f_main_ctrl_GBA_Bootstrapping(true);	206	f_main_ctrl_GBA_U_Bootstrapping(true);
224		207	
225	// in order to create a purchase items	208	// in order to create a purchase items
226	// ESG with purchase part	209	// ESG with purchase part
227	var ServiceGuideDeliveryUnit v_SGDU := {};	210	var ServiceGuideDeliveryUnit v_SGDU := {};
228	var UInt32 v_Version := 1;	211	var UInt32 v_Version := 1;
229		212	
230	f_main_ctrl_InitStartEndTime(0, c_one_hour);	213	f_main_ctrl_InitStartEndTime(0, c_one_hour);
231		214	
232	var DateTime v_DateTime_StartTime := f_getDateTime(v_startTime);	215	var DateTime v_DateTime_StartTime := f_getDateTime(v_startTime);
233	var DateTime v_DateTime_EndTime := f_getDateTime(v_endTime);	216	var DateTime v_DateTime_EndTime := f_getDateTime(v_endTime);
234		217	
235	// TODO: Session Description fragment containing SDP for "Programme" and the service protection	218	// TODO: Session Description fragment containing SDP for "Programme" and the service protection
236	var SdpFragment v_Sdp := valueof(m_def_SdpFragment(219	var SdpFragment v_Sdp := valueof(m_def_SdpFragment(
237	PX_SDP_VIDEO_PROG_1_REF_ID,	220	PX_SDP_VIDEO_PROG_1_REF_ID,
238	PX_SDP_SERVICE_PROTECTION_SRTP	221	PX_SDP_SERVICE_PROTECTION_SRTP
239));	222));
240		223	
241	var AccessType v_AccessType := valueof(m_def_AccessType_bc_sdp(224	var AccessType v_AccessType := valueof(m_def_AccessType_bc_sdp(
242	f_getAccessType(PX_BROADCAST_NETWORK_BEARER),	225	f_getAccessType(PX_BROADCAST_NETWORK_BEARER),
243	PX_SDP_VIDEO_PROG_1_REF_ID	226	PX_SDP_VIDEO_PROG_1_REF_ID
244));	227));
245		228	
246	var PriceInfoType v_PriceInfo := valueof(m_PriceInfo(229	var PriceInfoType v_PriceInfo := valueof(m_PriceInfo(
247	1,	230	1,
248	PX_MONETARY_PRICE,	231	PX_MONETARY_PRICE,
249	PX_CURRENCY	232	PX_CURRENCY
250));	233));
251		234	
252	var ServiceType v_Service := valueof(m_def_Service(235	var ServiceType v_Service := valueof(m_def_Service(
253	PX_SGDU_SERVICE_ID,	236	PX_SGDU_SERVICE_ID,
254	v_Version,	237	v_Version,
255	"PayTvChannel",	238	"PayTvChannel",
256	c_basicTv_ServiceType	239	c_basicTv_ServiceType
257));	240));
258		241	
259	var ContentType v_Content := valueof(m_def_Content(242	var ContentType v_Content := valueof(m_def_Content(
260	PX_SGDU_CONTENT_ID_PROG_1,	243	PX_SGDU_CONTENT_ID_PROG_1,
261	v_Version,	244	v_Version,
262	"Programme",	245	"Programme",
263	PX_SGDU_SERVICE_ID,	246	PX_SGDU_SERVICE_ID,
264	v_DateTime_StartTime,	247	v_DateTime_StartTime,
265	v_DateTime_EndTime	248	v_DateTime_EndTime
266));	249));
267		250	
268	var ScheduleType v_Schedule := valueof(m_def_Schedule(251	var ScheduleType v_Schedule := valueof(m_def_Schedule(
269	PX_SGDU_SCHEDULE_ID_PROG_1,	252	PX_SGDU_SCHEDULE_ID_PROG_1,
270	v_Version,	253	v_Version,
271	PX_SGDU_SERVICE_ID,	254	PX_SGDU_SERVICE_ID,
272	PX_SGDU_CONTENT_ID_PROG_1,	255	PX_SGDU_CONTENT_ID_PROG_1,
273	v_startTime, // change 01	256	v_startTime, // change 01 (WK 6)
274	v_endTime // change 01	257	v_endTime // change 01 (WK 6)
275));	258));
276		259	
277	var AccessType v_Access := valueof(m_def_Access(260	var AccessType v_Access := valueof(m_def_Access(
278	PX_SGDU_ACCESS_ID_PROG_1,	261	PX_SGDU_ACCESS_ID_PROG_1,
279	v_Version,	262	v_Version,
280	v_AccessType,	263	v_AccessType,
281	PX_SGDU_SERVICE_ID,	264	PX_SGDU_SERVICE_ID,
282	PX_SGDU_SCHEDULE_ID_PROG_1,	265	PX_SGDU_SCHEDULE_ID_PROG_1,
283	c_class_SG	266	c_class_SG
284));	267));

Datei: AtsBCast_ServiceProtectionTests.ttcn (Fortsetzung)

285	v_Access.KeyManagementSystems := {	268	v_Access.KeyManagementSystems := {
286	{	269	{
287	kmsType := 1, // 1 = oma-bcast-gba_u-mbms	270	kmsType := 1, // 1 = oma-bcast-gba_u-mbms
288	protectionType := 0, // Content protection only, as defined by the LTKM	271	protectionType := 0, // Content protection only, as defined by the LTKM
289	PermissionsIssuerURI := {	272	PermissionsIssuerURI := {
290	text_ := "http://" & PX_BSM_FQDN	273	text_ := "http://" & PX_BSM_FQDN
291	},	274	},
292	ProtectionKeyIDs := {	275	ProtectionKeyIDs := {
293	{	276	{
294	type_ := 0, // ProtectionKeyID = Key Domain ID concatenated with SEK/PEK ID	277	type_ := 0, // ProtectionKeyID = Key Domain ID concatenated with SEK/PEK ID
295	// key domain id = MCC MNC	278	// key domain id = MCC MNC
296	// sek id = key group key number	279	// sek id = key group key number
297	text_ := fx_bitstring2Base64(hex2bit(protectionKeyId)) // Base64 encoding	280	text_ := fx_bitstring2Base64(hex2bit(protectionKeyId)) // Base64 encoding
298	}	281	}
299	},	282	},
300	TerminalBindingKeyID := omit	283	TerminalBindingKeyID := omit
301	}	284	}
302	};	285	};
303		286	
304	var PurchaseItemType v_PurchaseItem := valueof(m_def_PurchaseItem(287	var PurchaseItemType v_PurchaseItem := valueof(m_def_PurchaseItem(
305	PX_SGDU_PURCHASE_ITEM_ID,	288	PX_SGDU_PURCHASE_ITEM_ID,
306	v_Version,	289	v_Version,
307	PX_SGDU_PURCHASE_DATA_ID,	290	PX_SGDU_PURCHASE_DATA_ID,
308	PX_SGDU_SERVICE_ID,	291	PX_SGDU_SERVICE_ID,
309	"PurchaseItem1"	292	"PurchaseItem1"
310));	293));
311	v_PurchaseItem.Descriptions := {{lang := omit, text_:"Purchasable SRTPEncrypted service"}}	294	v_PurchaseItem.Descriptions := {{lang := omit, text_:"Purchasable SRTPEncrypted service"}}
312		295	
313	var PurchaseDataType v_PurchaseData := valueof(m_def_PurchaseData(296	var PurchaseDataType v_PurchaseData := valueof(m_def_PurchaseData(
314	PX_SGDU_PURCHASE_DATA_ID,	297	PX_SGDU_PURCHASE_DATA_ID,
315	v_Version,	298	v_Version,
316	{{omit,"Discount price available"}},	299	{{omit,"Discount price available"}},
317	PX_SGDU_PURCHASE_ITEM_ID,	300	PX_SGDU_PURCHASE_ITEM_ID,
318	PX_SGDU_PURCHASE_CHANNEL_ID,	301	PX_SGDU_PURCHASE_CHANNEL_ID,
319	v_PriceInfo	302	v_PriceInfo
320));	303));
321		304	
322	var PurchaseChannelType v_PurchaseChannel := valueof(m_def_PurchaseChannel(305	var PurchaseChannelType v_PurchaseChannel := valueof(m_def_PurchaseChannel(
323	PX_SGDU_PURCHASE_CHANNEL_ID,	306	PX_SGDU_PURCHASE_CHANNEL_ID,
324	v_Version,	307	v_Version,
325	"PurchaseChannel",	308	"PurchaseChannel",
326	PX_PURCHASE_URL //hint: should have this format -> "http://" & PX_BSM_FQDN & "/bsm/purchase"	309	PX_PURCHASE_URL //hint: should have this format -> "http://" & PX_BSM_FQDN & "/bsm/purchase"
327));	310));
328		311	
329	// CR 26/8: generic function for adding XML fragments	312	// CR 26/8: generic function for adding XML fragments
330	f_addXMLFragment(v_SGDU, {Service := v_Service});	313	f_addXMLFragment(v_SGDU, {Service := v_Service});
331	f_addXMLFragment(v_SGDU, {Content := v_Content});	314	f_addXMLFragment(v_SGDU, {Content := v_Content});
332	f_addXMLFragment(v_SGDU, {Schedule := v_Schedule});	315	f_addXMLFragment(v_SGDU, {Schedule := v_Schedule});
333	f_addXMLFragment(v_SGDU, {Access := v_Access});	316	f_addXMLFragment(v_SGDU, {Access := v_Access});
334	f_addXMLFragment(v_SGDU, {PurchaseItem := v_PurchaseItem});	317	f_addXMLFragment(v_SGDU, {PurchaseItem := v_PurchaseItem});
335	f_addXMLFragment(v_SGDU, {PurchaseChannel := v_PurchaseChannel});	318	f_addXMLFragment(v_SGDU, {PurchaseChannel := v_PurchaseChannel});
336	f_addXMLFragment(v_SGDU, {PurchaseData := v_PurchaseData});	319	f_addXMLFragment(v_SGDU, {PurchaseData := v_PurchaseData});
337	f_addSdpFragment(v_SGDU, v_Version, v_Sdp);	320	f_addSdpFragment(v_SGDU, v_Version, v_Sdp);
338		321	
339	f_main_ctrl_broadcastServiceGuide(v_SGDU, omit, vc_FluteUpdateID);	322	f_main_ctrl_broadcastServiceGuide(v_SGDU, omit, vc_FluteUpdateID);
340		323	
341	f_main_ut_RunBCastApplication();	324	f_main_ut_RunBCastApplication();
342		325	
343	f_main_ut_CheckService("PayTvChannel");	326	f_main_ut_CheckService("PayTvChannel");
344	f_main_ut_SelectService("PayTvChannel");	327	f_main_ut_SelectService("PayTvChannel");
345	f_main_ut_CheckContent("Programme with price information", v_DateTime_StartTime, v_DateTime_EndTime);	328	f_main_ut_CheckContent("Programme with price information", v_DateTime_StartTime, v_DateTime_EndTime);
346		329	
347	// change 2 (WK23): GBA authentication/authorization extensions for content protection test	330	// change 2 (WK23): GBA authentication/authorization extensions for content protection test
348	// Bcast Subscription Managemant as parallel test component for MBMS User Service Registration	331	// Bcast Subscription Managemant as parallel test component for MBMS User Service Registration
349	// and Service Request/Response is not allowed	332	// and Service Request/Response is not allowed
350	f_main_ctrl_BSM_Procedures(true, true, false, true);	333	f_main_ctrl_BSM_Procedures(true, true, false, true);
351		334	
352	// postamble	335	// postamble
353	f_main_ut_PowerOff();	336	f_main_ut_PowerOff();
354		337	
355	f_cf_BSF_down();	338	f_cf_BSF_down();

Datei: AtsBCast_ServiceProtectionTests.ttcn (Fortsetzung)

356	f_cf_BSM_down();		339	f_cf_BSM_down();
357	f_cf_DNS_down();		340	f_cf_DNS_down();
358	f_cf_Broadcast_down();		341	f_cf_Broadcast_down();
359	f_cf_UpperTester_down();		342	f_cf_UpperTester_down();
360	f_terminateComponents();		343	f_terminateComponents();
361	} else {			
362	log("Verdict Info: Preconditions are not meet. See PIXIT values 'PX_GBA' and 'PX_SUPPORT_MBMS_KEY_MANAGEMENT');			
363	setverdict(inconc);			
364	}			
365	}	=	344	}
366			345	
367	// change 3 (WK34): Service Protection: secure streaming service		346	// change 3 (WK34): Service Protection: secure streaming service
368	// change 6 (WK23): new service protection testcase		347	// change 6 (WK23): new service protection testcase
369	testcase TC_BCAST_SERVPROT_101d() runs on BCastMainComponent system BCastPlatformComponent {		348	testcase TC_BCAST_SERVPROT_101d() runs on BCastMainComponent system BCastPlatformComponent {
370	// change 3 (WK46/08): Test if MBMS Key Management Support is available	<>		
371	if (PX_GBA == e_gba_u and PX_SUPPORT_MBMS_KEY_MANAGEMENT) {			
372	// init components and ports		349	// init components and ports
373	f_createComponents();		350	f_createComponents();
374	f_cf_Broadcast_up();		351	f_cf_Broadcast_up();
375	f_cf_UpperTester_up();		352	f_cf_UpperTester_up();
376	f_cf_DNS_up();		353	f_cf_DNS_up();
377	f_cf_BSM_up();		354	f_cf_BSM_up();
378	f_cf_BSF_up();		355	f_cf_BSF_up();
379			356	
380	// preamble		357	// preamble
381	f_cf_enable_DNS();		358	f_cf_enable_DNS();
382	f_startCommonUtPreamble();		359	f_startCommonUtPreamble();
383			360	
384	var hexstring protectionKeyId := PX_MCC & PX_MNC & PX_KEY_GROUP & PX_KEY_NUMBER;		361	var hexstring protectionKeyId := PX_MCC & PX_MNC & PX_KEY_GROUP & PX_KEY_NUMBER;
385	f_main_ctrl_InitProtectionKeysAndValues();		362	f_main_ctrl_InitProtectionKeysAndValues();
386	f_main_ctrl_SetProtectionKeyId(protectionKeyId);		363	f_main_ctrl_SetProtectionKeyId(protectionKeyId);
387	f_main_ctrl_SetBootstrappingTransactionId(f_bsf_generateBtid());		364	f_main_ctrl_SetBootstrappingTransactionId(f_bsf_generateBtid());
388	// GBA/BSF as parallel test component		365	// GBA/BSF as parallel test component
389	f_main_ctrl_GBA_Bootstrapping(true);		366	f_main_ctrl_GBA_U_Bootstrapping(true);
390			367	
391	// in order to create a purchase items		368	// in order to create a purchase items
392	// ESG with purchase part		369	// ESG with purchase part
393	var ServiceGuideDeliveryUnit v_SGDU := {};		370	var ServiceGuideDeliveryUnit v_SGDU := {};
394	var UInt32 v_Version := 1;		371	var UInt32 v_Version := 1;
395			372	
396	f_main_ctrl_InitStartEndTime(0, c_one_hour);		373	f_main_ctrl_InitStartEndTime(0, c_one_hour);
397			374	
398	var DateTime v_DateTime_StartTime := f_getDateTime(v_startTime);		375	var DateTime v_DateTime_StartTime := f_getDateTime(v_startTime);
399	var DateTime v_DateTime_EndTime := f_getDateTime(v_endTime);		376	var DateTime v_DateTime_EndTime := f_getDateTime(v_endTime);
400			377	
401	// TODO: Session Description fragment containing SDP for "Programme" and the service protection		378	// TODO: Session Description fragment containing SDP for "Programme" and the service protection
402	var SdpFragment v_Sdp := valueof(m_def_SdpFragment(379	var SdpFragment v_Sdp := valueof(m_def_SdpFragment(
403	PX_SDP_VIDEO_PROG_1_REF_ID,		380	PX_SDP_VIDEO_PROG_1_REF_ID,
404	PX_SDP_SERVICE_PROTECTION_SRTP		381	PX_SDP_SERVICE_PROTECTION_SRTP
405));		382));
406			383	
407	var AccessTypeType v_AccessType := valueof(m_def_AccessType_bc_sdp(384	var AccessTypeType v_AccessType := valueof(m_def_AccessType_bc_sdp(
408	f_getAccessType(PX_BROADCAST_NETWORK_BEARER),		385	f_getAccessType(PX_BROADCAST_NETWORK_BEARER),
409	PX_SDP_VIDEO_PROG_1_REF_ID		386	PX_SDP_VIDEO_PROG_1_REF_ID
410));		387));
411			388	
412	var PriceInfoType v_PriceInfo := valueof(m_PriceInfo(389	var PriceInfoType v_PriceInfo := valueof(m_PriceInfo(
413	1,		390	1,
414	PX_MONETARY_PRICE,		391	PX_MONETARY_PRICE,
415	PX_CURRENCY		392	PX_CURRENCY
416));		393));
417			394	
418	var ServiceType v_Service := valueof(m_def_Service(395	var ServiceType v_Service := valueof(m_def_Service(
419	PX_SGDU_SERVICE_ID,		396	PX_SGDU_SERVICE_ID,
420	v_Version,		397	v_Version,
421	"PayTvChannel",		398	"PayTvChannel",
422	c_basicTv_ServiceType		399	c_basicTv_ServiceType
423));		400));
424			401	
425	var ContentType v_Content := valueof(m_def_Content(402	var ContentType v_Content := valueof(m_def_Content(
426	PX_SGDU_CONTENT_ID_PROG_1,		403	PX_SGDU_CONTENT_ID_PROG_1,

Datei: AtsBCast_ServiceProtectionTests.ttcn (Fortsetzung)

```

427         v_Version,
428         "Programme",
429         PX_SGDU_SERVICE_ID,
430         v_DateTime_StartTime,
431         v_DateTime_EndTime
432     ));
433
434     var ScheduleType v_Schedule := valueof(m_def_Schedule(
435         PX_SGDU_SCHEDULE_ID_PROG_1,
436         v_Version,
437         PX_SGDU_SERVICE_ID,
438         PX_SGDU_CONTENT_ID_PROG_1,
439         v_startTime,
440         v_endTime
441     ));
442
443     var AccessType v_Access := valueof(m_def_Access(
444         PX_SGDU_ACCESS_ID_PROG_1,
445         v_Version,
446         v_AccessType,
447         PX_SGDU_SERVICE_ID,
448         PX_SGDU_SCHEDULE_ID_PROG_1,
449         c_class_SG
450     ));
451     v_Access.KeyManagementSystems := {
452         {
453             kmsType := 1, // 1 = oma-bcast-gba_u-mbms
454             protectionType := 0, // Content protection only, as defined by the LTKM
455             PermissionsIssuerURI := {
456                 text_ := PX_BSM_FQDN
457             },
458             ProtectionKeyIDs := {
459                 {
460                     type_ := 0, // ProtectionKeyID = Key Domain ID concatenated with SEK/PE
461                     // key domain id = MCC || MNC
462                     // sek id = key group || key number
463                     text_ := fx_bitstring2Base64(hex2bit(protectionKeyId)) // Base64 encoded
464                 }
465             },
466             TerminalBindingKeyID := omit
467         }
468     };
469
470     var PurchaseItemType v_PurchaseItem := valueof(m_def_PurchaseItem(
471         PX_SGDU_PURCHASE_ITEM_ID,
472         v_Version,
473         PX_SGDU_PURCHASE_DATA_ID,
474         PX_SGDU_SERVICE_ID,
475         "PurchaseItem1"
476     ));
477     v_PurchaseItem.Descriptions := {{lang := omit, text_ := "Purchasable SRTPEncrypted service"}
478
479     var PurchaseDataType v_PurchaseData := valueof(m_def_PurchaseData(
480         PX_SGDU_PURCHASE_DATA_ID,
481         v_Version,
482         {{omit, "Discount price available"}},
483         PX_SGDU_PURCHASE_ITEM_ID,
484         PX_SGDU_PURCHASE_CHANNEL_ID,
485         v_PriceInfo
486     ));
487
488     var PurchaseChannelType v_PurchaseChannel := valueof(m_def_PurchaseChannel(
489         PX_SGDU_PURCHASE_CHANNEL_ID,
490         v_Version,
491         "PurchaseChannel",
492         "http://" & PX_BSM_FQDN & "/bsm/purchaseURL"
493     ));
494
495     // CR 26/8: generic function for adding XML fragments
496     f_addXMLFragment(v_SGDU, {Service := v_Service});
497     f_addXMLFragment(v_SGDU, {Content := v_Content});

```

```

404         v_Version,
405         "Programme",
406         PX_SGDU_SERVICE_ID,
407         v_DateTime_StartTime,
408         v_DateTime_EndTime
409     ));
410
411     var ScheduleType v_Schedule := valueof(m_def_Schedule(
412         PX_SGDU_SCHEDULE_ID_PROG_1,
413         v_Version,
414         PX_SGDU_SERVICE_ID,
415         PX_SGDU_CONTENT_ID_PROG_1,
416         v_startTime,
417         v_endTime
418     ));
419
420     var AccessType v_Access := valueof(m_def_Access(
421         PX_SGDU_ACCESS_ID_PROG_1,
422         v_Version,
423         v_AccessType,
424         PX_SGDU_SERVICE_ID,
425         PX_SGDU_SCHEDULE_ID_PROG_1,
426         c_class_SG
427     ));
428     v_Access.KeyManagementSystems := {
429         {
430             kmsType := 1, // 1 = oma-bcast-gba_u-mbms
431             protectionType := 0, // Content protection only, as defined by the LTKM
432             PermissionsIssuerURI := {
433                 text_ := PX_BSM_FQDN
434             },
435             ProtectionKeyIDs := {
436                 {
437                     type_ := 0, // ProtectionKeyID = Key Domain ID concatenated with SEK/PEK ID
438                     // key domain id = MCC || MNC
439                     // sek id = key group || key number
440                     text_ := fx_bitstring2Base64(hex2bit(protectionKeyId)) // Base64 encoding
441                 }
442             },
443             TerminalBindingKeyID := omit
444         }
445     };
446
447     var PurchaseItemType v_PurchaseItem := valueof(m_def_PurchaseItem(
448         PX_SGDU_PURCHASE_ITEM_ID,
449         v_Version,
450         PX_SGDU_PURCHASE_DATA_ID,
451         PX_SGDU_SERVICE_ID,
452         "PurchaseItem1"
453     ));
454     v_PurchaseItem.Descriptions := {{lang := omit, text_:"Purchasable SRTPEncrypted service"}};
455
456     var PurchaseDataType v_PurchaseData := valueof(m_def_PurchaseData(
457         PX_SGDU_PURCHASE_DATA_ID,
458         v_Version,
459         {{omit,"Discount price available"}},
460         PX_SGDU_PURCHASE_ITEM_ID,
461         PX_SGDU_PURCHASE_CHANNEL_ID,
462         v_PriceInfo
463     ));
464
465     var PurchaseChannelType v_PurchaseChannel := valueof(m_def_PurchaseChannel(
466         PX_SGDU_PURCHASE_CHANNEL_ID,
467         v_Version,
468         "PurchaseChannel",
469         "http://" & PX_BSM_FQDN & "/bsm/purchaseURL"
470     ));
471
472     // CR 26/8: generic function for adding XML fragments
473     f_addXMLFragment(v_SGDU, {Service := v_Service});
474     f_addXMLFragment(v_SGDU, {Content := v_Content});

```

Datei: AtsBCast_ServiceProtectionTests.ttcn (Fortsetzung)

498	f_addXMLFragment(v_SGDU, {Schedule := v_Schedule});	475	f_addXMLFragment(v_SGDU, {Schedule := v_Schedule});
499	f_addXMLFragment(v_SGDU, {Access := v_Access});	476	f_addXMLFragment(v_SGDU, {Access := v_Access});
500	f_addXMLFragment(v_SGDU, {PurchaseItem := v_PurchaseItem});	477	f_addXMLFragment(v_SGDU, {PurchaseItem := v_PurchaseItem});
501	f_addXMLFragment(v_SGDU, {PurchaseChannel := v_PurchaseChannel});	478	f_addXMLFragment(v_SGDU, {PurchaseChannel := v_PurchaseChannel});
502	f_addXMLFragment(v_SGDU, {PurchaseData := v_PurchaseData});	479	f_addXMLFragment(v_SGDU, {PurchaseData := v_PurchaseData});
503	f_addSdpFragment(v_SGDU, v_Version, v_Sdp);	480	f_addSdpFragment(v_SGDU, v_Version, v_Sdp);
504		481	
505	f_main_ctrl_broadcastServiceGuide(v_SGDU, omit, vc_FluteUpdateID);	482	f_main_ctrl_broadcastServiceGuide(v_SGDU, omit, vc_FluteUpdateID);
506		483	
507	f_main_ut_RunBCastApplication();	484	f_main_ut_RunBCastApplication();
508		485	
509	f_main_ut_CheckService("PayTvChannel");	486	f_main_ut_CheckService("PayTvChannel");
510	f_main_ut_SelectService("PayTvChannel");	487	f_main_ut_SelectService("PayTvChannel");
511	f_main_ut_CheckContent("Programme with price information", v_DateTime_StartTime, v_DateTime_EndTime);	488	f_main_ut_CheckContent("Programme with price information", v_DateTime_StartTime, v_DateTime_EndTime);
512	f_main_ut_PurchaseService("PayTvChannel");	489	f_main_ut_PurchaseService("PayTvChannel");
513		490	
514	// Bcast Subscription Managemant as parallel test component for MBMS User Service Registration	491	// Bcast Subscription Managemant as parallel test component for MBMS User Service Registration
515	// and Service Request and Service Response are test purpose	492	// and Service Request and Service Response are test purpose
516	f_main_ctrl_BSM_Procedures(true, true, true, false);	493	f_main_ctrl_BSM_Procedures(true, true, true, false);
517		494	
518	// postamble	495	// postamble
519	f_main_ut_PowerOff();	496	f_main_ut_PowerOff();
520		497	
521	f_cf_BSF_down();	498	f_cf_BSF_down();
522	f_cf_BSM_down();	499	f_cf_BSM_down();
523	f_cf_DNS_down();	500	f_cf_DNS_down();
524	f_cf_Broadcast_down();	501	f_cf_Broadcast_down();
525	f_cf_UpperTester_down();	502	f_cf_UpperTester_down();
526	f_terminateComponents();	503	f_terminateComponents();
527	} else {		
528	log("Verdict Info: Preconditions are not meet. See PIXIT values 'PX_GBA' and 'PX_SUPPORT_MBMS_KEY_MANAGEMENT');"		
529	setverdict(inconc);		
530	}		
531	}	=	504
532			505
533	// change 3 (WK34): Service Protection: secure streaming service		506
534	// change 6 (WK23): new service protection testcase		507
535	testcase TC_BCAST_SERVPROT_101e() runs on BCastMainComponent system BCastPlatformComponent {		508
536	// change 3 (WK46/08): Test if MBMS Key Management Support is available	<>	
537	if (PX_GBA == e_gba_u and PX_SUPPORT_MBMS_KEY_MANAGEMENT) {		509
538	// init components and ports		510
539	f_createComponents();		511
540	f_cf_Broadcast_up();		512
541	f_cf_UpperTester_up();		513
542	f_cf_DNS_up();		514
543	f_cf_BSM_up();		515
544	f_cf_BSF_up();		516
545	f_cf_Mikey_up();		517
546			518
547	// preamble		519
548	f_cf_enable_DNS();		520
549	f_startCommonUtPreamble();		521
550			522
551	var hexstring v_keyDomainId := PX_MCC & PX_MNC;		523
552	var hexstring v_mskId := PX_KEY_GROUP & PX_KEY_NUMBER;		524
553	var hexstring v_protectionKeyId := v_keyDomainId & v_mskId;		525
554	var octetstring v_ks_NAF := f_main_ctrl_InitProtectionKeysAndValues();		526
555	f_main_ctrl_SetProtectionKeyId(v_protectionKeyId);		527
556	var charstring v_btId := f_bsfc_generateBtid();		528
557	f_main_ctrl_SetBootstrappingTransactionId(v_btId);		
558			
559	// GBA/BSF as parallel test component		
560	f_main_ctrl_GBA_Bootstrapping(true);		
561	// Bcast Subscription Managemant as parallel test component for MBMS User Service Registration	529	// Bcast Subscription Managemant as parallel test component for MBMS User Service Registration
562	// and Service Request/Response	530	// and Service Request/Response
563	f_main_ctrl_BSM_Procedures(true, true, true, true);	531	f_main_ctrl_BSM_Procedures(true, true, true, true);
		532	// GBA/BSF as parallel test component
		533	f_main_ctrl_GBA_U_Bootstrapping(true);
564		534	
565	// in order to create a purchase items	535	// in order to create a purchase items
566	// ESG with purchase part	536	// ESG with purchase part

Datei: AtsBCast_ServiceProtectionTests.ttcn (Fortsetzung)

```

567 var ServiceGuideDeliveryUnit v_SGDU := {};
568 var UInt32 v_Version := 1;
569
570 f_main_ctrl_InitStartEndTime(0, c_one_hour);
571
572 var DateTime v_DateTime_StartTime := f_getDateTime(v_startTime);
573 var DateTime v_DateTime_EndTime := f_getDateTime(v_endTime);
574
575 // TODO: Session Description fragment containing SDP for "Programme" and the service p
576 var SdpFragment v_Sdp := valueof(m_def_SdpFragment(
577     PX_SDP_VIDEO_PROG_1_REF_ID,
578     PX_SDP_SERVICE_PROTECTION_SRTTP
579 ));
580
581 var AccessTypeType v_AccessType := valueof(m_def_AccessType_bc_sdp(
582     f_getAccessType(PX_BROADCAST_NETWORK_BEARER),
583     PX_SDP_VIDEO_PROG_1_REF_ID
584 ));
585
586 var PriceInfoType v_PriceInfo := valueof(m_PriceInfo(
587     1,
588     PX_MONETARY_PRICE,
589     PX_CURRENCY
590 ));
591
592 var ServiceType v_Service := valueof(m_def_Service(
593     PX_SGDU_SERVICE_ID,
594     v_Version,
595     "PayTvChannel",
596     c_basicTv_ServiceType
597 ));
598
599 var ContentType v_Content := valueof(m_def_Content(
600     PX_SGDU_CONTENT_ID_PROG_1,
601     v_Version,
602     "Programme",
603     PX_SGDU_SERVICE_ID,
604     v_DateTime_StartTime,
605     v_DateTime_EndTime
606 ));
607
608 var ScheduleType v_Schedule := valueof(m_def_Schedule(
609     PX_SGDU_SCHEDULE_ID_PROG_1,
610     v_Version,
611     PX_SGDU_SERVICE_ID,
612     PX_SGDU_CONTENT_ID_PROG_1,
613     v_startTime, // change 01 (WK 6): global time definition
614     v_endTime // change 01 (WK 6): global time definition
615 ));
616
617 var AccessType v_Access := valueof(m_def_Access(
618     PX_SGDU_ACCESS_ID_PROG_1,
619     v_Version,
620     v_AccessType,
621     PX_SGDU_SERVICE_ID,
622     PX_SGDU_SCHEDULE_ID_PROG_1,
623     c_class_SG
624 ));
625 v_Access.KeyManagementSystems := {
626     {
627         kmsType := 1, // 1 = oma-bcast-gba_u-mbms
628         protectionType := 0, // Content protection only, as defined by the LTKM
629         PermissionsIssuerURI := {
630             text_ := "http://" & PX_BSM_FQDN
631         },
632         ProtectionKeyIDs := {
633             {
634                 type_ := 0, // ProtectionKeyID = Key Domain ID concatenated with SEK/PH
635                 // key domain id = MCC || MNC
636                 // sek id = key group || key number
637                 text_ := fx bitstring2Base64(hex2bit(v_protectionKeyId)) // Base64 enc

```

```

537 var ServiceGuideDeliveryUnit v_SGDU := {};
538 var UInt32 v_Version := 1;
539
540 f_main_ctrl_InitStartTime(0, c_one_hour);
541
542 var DateTime v_DateTime_StartTime := f_getDateTime(v_startTime);
543 var DateTime v_DateTime_EndTime := f_getDateTime(v_endTime);
544
545 // TODO: Session Description fragment containing SDP for "Programme" and the service protection
546 var SdpFragment v_Sdp := valueof(m_def_SdpFragment(
547     PX_SDP_VIDEO_PROG_1_REF_ID,
548     PX_SDP_SERVICE_PROTECTION_SRTP
549 ));
550
551 var AccessType v_AccessType := valueof(m_def_AccessType_bc_sdp(
552     f_getAccessType(PX_BROADCAST_NETWORK_BEARER),
553     PX_SDP_VIDEO_PROG_1_REF_ID
554 ));
555
556 var PriceInfoType v_PriceInfo := valueof(m_PriceInfo(
557     1,
558     PX_MONETARY_PRICE,
559     PX_CURRENCY
560 ));
561
562 var ServiceType v_Service := valueof(m_def_Service(
563     PX_SGDU_SERVICE_ID,
564     v_Version,
565     "PayTvChannel",
566     c_basicTv_ServiceType
567 ));
568
569 var ContentType v_Content := valueof(m_def_Content(
570     PX_SGDU_CONTENT_ID_PROG_1,
571     v_Version,
572     "Programme",
573     PX_SGDU_SERVICE_ID,
574     v_DateTime_StartTime,
575     v_DateTime_EndTime
576 ));
577
578 var ScheduleType v_Schedule := valueof(m_def_Schedule(
579     PX_SGDU_SCHEDULE_ID_PROG_1,
580     v_Version,
581     PX_SGDU_SERVICE_ID,
582     PX_SGDU_CONTENT_ID_PROG_1,
583     v_startTime, // change 01 (WK 6)
584     v_endTime // change 01 (WK 6)
585 ));
586
587 var AccessType v_Access := valueof(m_def_Access(
588     PX_SGDU_ACCESS_ID_PROG_1,
589     v_Version,
590     v_AccessType,
591     PX_SGDU_SERVICE_ID,
592     PX_SGDU_SCHEDULE_ID_PROG_1,
593     c_class_SG
594 ));
595 v_Access.KeyManagementSystems := {
596     {
597         kmsType := 1, // 1 = oma-bcast-gba_u-mbms
598         protectionType := 0, // Content protection only, as defined by the LTKM
599         PermissionsIssuerURI := {
600             text_ := "http://" & PX_BSM_FQDN
601         },
602         ProtectionKeyIDs := {
603             {
604                 type_ := 0, // ProtectionKeyID = Key Domain ID concatenated with SEK/PEK ID
605                 // key domain id = MCC || MNC
606                 // sek id = key group || key number
607                 text_ := fx_bitstring2Base64(hex2bit(v_protectionKeyId)) // Base64 encoding

```

Datei: AtsBCast_ServiceProtectionTests.ttcn (Fortsetzung)

638	}	608	}
639	},	609	},
640	TerminalBindingKeyID := omit	610	TerminalBindingKeyID := omit
641	}	611	}
642	};	612	};
643		613	
644	var PurchaseItemType v_PurchaseItem := valueof(m_def_PurchaseItem(614	var PurchaseItemType v_PurchaseItem := valueof(m_def_PurchaseItem(
645	PX_SGDU_PURCHASE_ITEM_ID,	615	PX_SGDU_PURCHASE_ITEM_ID,
646	v_Version,	616	v_Version,
647	PX_SGDU_PURCHASE_DATA_ID,	617	PX_SGDU_PURCHASE_DATA_ID,
648	PX_SGDU_SERVICE_ID,	618	PX_SGDU_SERVICE_ID,
649	"PurchaseItem1"	619	"PurchaseItem1"
650));	620));
651	v_PurchaseItem.Descriptions := {{lang := omit, text_:"Purchasable SRTPEncrypted service"}}	621	v_PurchaseItem.Descriptions := {{lang := omit, text_:"Purchasable SRTPEncrypted service"}}
652		622	
653	var PurchaseDataType v_PurchaseData := valueof(m_def_PurchaseData(623	var PurchaseDataType v_PurchaseData := valueof(m_def_PurchaseData(
654	PX_SGDU_PURCHASE_DATA_ID,	624	PX_SGDU_PURCHASE_DATA_ID,
655	v_Version,	625	v_Version,
656	{{omit,"Discount price available"}},	626	{{omit,"Discount price available"}},
657	PX_SGDU_PURCHASE_ITEM_ID,	627	PX_SGDU_PURCHASE_ITEM_ID,
658	PX_SGDU_PURCHASE_CHANNEL_ID,	628	PX_SGDU_PURCHASE_CHANNEL_ID,
659	v_PriceInfo	629	v_PriceInfo
660));	630));
661		631	
662	var PurchaseChannelType v_PurchaseChannel := valueof(m_def_PurchaseChannel(632	var PurchaseChannelType v_PurchaseChannel := valueof(m_def_PurchaseChannel(
663	PX_SGDU_PURCHASE_CHANNEL_ID,	633	PX_SGDU_PURCHASE_CHANNEL_ID,
664	v_Version,	634	v_Version,
665	"PurchaseChannel",	635	"PurchaseChannel",
666	"http://" & PX_BSM_FQDN & "/bsm/purchaseURL"	636	"http://" & PX_BSM_FQDN & "/bsm/purchaseURL"
667));	637));
668		638	
669	f_addXMLFragment(v_SGDU, {Service := v_Service});	639	f_addXMLFragment(v_SGDU, {Service := v_Service});
670	f_addXMLFragment(v_SGDU, {Content := v_Content});	640	f_addXMLFragment(v_SGDU, {Content := v_Content});
671	f_addXMLFragment(v_SGDU, {Schedule := v_Schedule});	641	f_addXMLFragment(v_SGDU, {Schedule := v_Schedule});
672	f_addXMLFragment(v_SGDU, {Access := v_Access});	642	f_addXMLFragment(v_SGDU, {Access := v_Access});
673	f_addXMLFragment(v_SGDU, {PurchaseItem := v_PurchaseItem});	643	f_addXMLFragment(v_SGDU, {PurchaseItem := v_PurchaseItem});
674	f_addXMLFragment(v_SGDU, {PurchaseChannel := v_PurchaseChannel});	644	f_addXMLFragment(v_SGDU, {PurchaseChannel := v_PurchaseChannel});
675	f_addXMLFragment(v_SGDU, {PurchaseData := v_PurchaseData});	645	f_addXMLFragment(v_SGDU, {PurchaseData := v_PurchaseData});
676	f_addSdpFragment(v_SGDU, v_Version, v_Sdp);	646	f_addSdpFragment(v_SGDU, v_Version, v_Sdp);
677		647	
678	f_main_ctrl_broadcastServiceGuide(v_SGDU, omit, vc_FluteUpdateID);	648	f_main_ctrl_broadcastServiceGuide(v_SGDU, omit, vc_FluteUpdateID);
679		649	
680	f_main_ut_RunBCastApplication();	650	f_main_ut_RunBCastApplication();
681		651	
682	f_main_ut_CheckService("PayTvChannel");	652	f_main_ut_CheckService("PayTvChannel");
683	f_main_ut_SelectService("PayTvChannel");	653	f_main_ut_SelectService("PayTvChannel");
684	f_main_ut_CheckContent("Programme with price information", v_DateTime_StartTime, v_DateTime_EndTime);	654	f_main_ut_CheckContent("Programme with price information", v_DateTime_StartTime, v_DateTime_EndTime);
685	f_main_ut_PurchaseService("PayTvChannel");	655	f_main_ut_PurchaseService("PayTvChannel");
686		656	
687	// send via UDP the LTKM	657	// send via UDP the LTKM
688	var UInt8 v_PolicyNumber := 0;	658	var UInt8 v_PolicyNumber := 0;
689	var octetstring v_randValue := '6813b3758c1cfe138d8b2ba886d3f449'O;	659	var octetstring v_randValue := '6813b3758c1cfe138d8b2ba886d3f449'O;
690	var octetstring v_csbId := '00000000'O;	660	var octetstring v_csbId := '00000000'O;
691	var Oct4 v_ssrc := '7BE6AC2E'O;	661	var Oct4 v_ssrc := '7BE6AC2E'O;
692		662	
693	var ListOfSrtpId v_SrtpIds := {}; // empty list	663	var ListOfSrtpId v_SrtpIds := {}; // empty list
694	var SrtpId v_SrtpId := valueof(m_def_SrtpId(v_PolicyNumber, v_ssrc, '00000000'O));	664	var SrtpId v_SrtpId := valueof(m_def_SrtpId(v_PolicyNumber, v_ssrc, '00000000'O));
695	f_addSrtpId(v_SrtpIds, v_SrtpId);	665	f_addSrtpId(v_SrtpIds, v_SrtpId);
696	//v_SrtpId := valueof(m_def_SrtpId(v_PolicyNumber, '00000000'O, '00000000'O));	666	//v_SrtpId := valueof(m_def_SrtpId(v_PolicyNumber, '00000000'O, '00000000'O));
697	//f_addSrtpId(v_SrtpIds, v_SrtpId);	667	//f_addSrtpId(v_SrtpIds, v_SrtpId);
698		668	
699	var MikeyHeaderPayload v_mikeyheader := valueof(m_def_LTKMHeader(GeneralExt, v_csbId, v_SrtpId));	669	var MikeyHeaderPayload v_mikeyheader := valueof(m_def_LTKMHeader(GeneralExt, v_csbId, v_SrtpId));
700		670	
701	var ListOfPayload v_payloads := {}; // empty list	671	var ListOfPayload v_payloads := {}; // empty list
702		672	
703	var KeyId v_keyIdKeyDomianId := valueof(m_def_KeyId_KeyDomainId(hex2oct(v_keyDomainId)));	673	var KeyId v_keyIdKeyDomianId := valueof(m_def_KeyId_KeyDomainId(hex2oct(v_keyDomainId)));
704	var KeyId v_keyIdMskId := valueof(m_def_KeyId_MSKId(hex2oct(v_mskId)));	674	var KeyId v_keyIdMskId := valueof(m_def_KeyId_MSKId(hex2oct(v_mskId)));
705	var GeneralExtension v_ExtMBMS := valueof(m_def_GeneralExtension(KEY_ID, {keyIds := {v_keyIdKeyDomianId, v_keyIdMskId}}));	675	var GeneralExtension v_ExtMBMS := valueof(m_def_GeneralExtension(KEY_ID, {keyIds := {v_keyIdKeyDomianId, v_keyIdMskId}}));
706	f_addMikeyPayload(v_payloads, {generalExtension := v_ExtMBMS});	676	f_addMikeyPayload(v_payloads, {generalExtension := v_ExtMBMS});
707		677	
708	var BcastExtension v_bcastExt := valueof(m_def_BcastExtension_LTKM(m_def_LTKMManagementData, v_PolicyNumber, v_SrtpId, v_ExtMBMS));	678	var BcastExtension v_bcastExt := valueof(m_def_BcastExtension_LTKM(m_def_LTKMManagementData, v_PolicyNumber, v_SrtpId, v_ExtMBMS));

Datei: AtsBCast_ServiceProtectionTests.ttcn (Fortsetzung)

709	var GeneralExtension v_ExtBCAST := valueof(m_def_GeneralExtension(OMA_BCAST, {bcastExtensid	679	var GeneralExtension v_ExtBCAST := valueof(m_def_GeneralExtension(OMA_BCAST, {bcastExtensid
710	f_addMikeyPayload(v_payloads, {generalExtension := v_ExtBCAST});	680	f_addMikeyPayload(v_payloads, {generalExtension := v_ExtBCAST});
711		681	
712	var integer v_ntpTimestamp := f_getNTPTime();	682	var integer v_ntpTimestamp := f_getNTPTime();
713	var Timestamp v_Timestamp := valueof(m_def_TimestampPayloadData(NTP.UTC, v_ntpTimestamp	683	var Timestamp v_Timestamp := valueof(m_def_TimestampPayloadData(NTP.UTC, v_ntpTimestamp));
714	f_addMikeyPayload(v_payloads, {timestamp := v_Timestamp});	684	f_addMikeyPayload(v_payloads, {timestamp := v_Timestamp});
715		685	
716	var Rand v_Rand := valueof(m_def_RandPayloadData(v_randValue));	686	var Rand v_Rand := valueof(m_def_RandPayloadData(v_randValue));
717	f_addMikeyPayload(v_payloads, {rand := v_Rand});	687	f_addMikeyPayload(v_payloads, {rand := v_Rand});
718		688	
719	var Id v_Idi := valueof(m_def_IdPayloadData(URI, PX_BSM_FQDN));	689	var Id v_Idi := valueof(m_def_IdPayloadData(URI, PX_BSM_FQDN));
720	f_addMikeyPayload(v_payloads, {id := v_Idi});	690	f_addMikeyPayload(v_payloads, {id := v_Idi});
721		691	
722	var Id v_Idr := valueof(m_def_IdPayloadData(NAI, v_btId));	692	var Id v_Idr := valueof(m_def_IdPayloadData(NAI, v_btId));
723	f_addMikeyPayload(v_payloads, {id := v_Idr});	693	f_addMikeyPayload(v_payloads, {id := v_Idr});
724		694	
725	// var ListOfSrtpPolicyParam v_SrtpPolicyParams := {}; //empty list	695	// var ListOfSrtpPolicyParam v_SrtpPolicyParams := {}; //empty list
726	// var SrtpPolicyParameter v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTP_	696	// var SrtpPolicyParameter v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParam
727	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);	697	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);
728	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SESSION_ENCR_KEY_LENGTH, 1, {	698	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SESSION_ENCR_KEY_LEN
729	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);	699	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);
730	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTP_AUTH_ALGORITHM, 1, {srtp	700	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTP_AUTH_ALGORITHM,
731	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);	701	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);
732	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SESSION_AUTH_KEY_LENGTH, 1, {	702	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SESSION_AUTH_KEY_LEN
733	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);	703	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);
734	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SESSION_SALT_KEY_LENGTH, 1, {	704	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SESSION_SALT_KEY_LEN
735	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);	705	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);
736	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTP_PRF, 1, {srtpPrf := SRTP	706	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTP_PRF, 1, {srtpPr
737	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);	707	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);
738	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(KEY_DERIVATION_RATE, 1, {keyD	708	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(KEY_DERIVATION_RATE,
739	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);	709	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);
740	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTP_ENCR_OFF_ON, 1, {srtpEnc	710	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTP_ENCR_OFF_ON, 1,
741	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);	711	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);
742	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTCP_ENCR_OFF_ON, 1, {srtcpE	712	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTCP_ENCR_OFF_ON, 1,
743	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);	713	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);
744	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SENDER_FEC_ORDER, 1, {fecOrder	714	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SENDER_FEC_ORDER, 1,
745	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);	715	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);
746	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTP_AUTH_OFF_ON, 1, {srtpAut	716	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTP_AUTH_OFF_ON, 1,
747	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);	717	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);
748	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(AUTH_TAG_LENGTH, 1, {authTagI	718	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(AUTH_TAG_LENGTH, 1,
749	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);	719	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);
750	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTP_PREFIX_LENGTH, 1, {srtpP	720	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTP_PREFIX_LENGTH,
751	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);	721	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);
752	// var SecurityPolicy v_SecurityPolicy := valueof(m_def_SecurityPolicy4Srtp(v_PolicyNum	722	// var SecurityPolicy v_SecurityPolicy := valueof(m_def_SecurityPolicy4Srtp(v_
753	// f_addMikeyPayload(v_payloads, {securityPolicy := v_SecurityPolicy});	723	// f_addMikeyPayload(v_payloads, {securityPolicy := v_SecurityPolicy});
754		724	
755	// preparations for encrypting of LTKM included in KEMAC	725	// preparations for encrypting of LTKM included in KEMAC
756	var Interval v_interval := valueof(m_def_Interval('00'O, '0f'O));	726	var Interval v_interval := valueof(m_def_Interval('00'O, '0f'O));
757	var Key v_keyData := valueof(m_def_KeyData4LTKM(KV_INTERVAL, PX_LTKM, {interval := v_in	727	var Key v_keyData := valueof(m_def_KeyData4LTKM(KV_INTERVAL, PX_LTKM, {interval := v_interv
758	var MikeyPayload v_keyDataPayload := valueof(m_def_Payload(LastPayload, {key := v_keyDa	728	var MikeyPayload v_keyDataPayload := valueof(m_def_Payload(LastPayload, {key := v_keyData}))
759	// PSK is MUK is ks_(ext/int)_NAF (depending of used GBA algorithm and UICC supports MB	729	// PSK is MUK is ks_(ext/int)_NAF (depending of used GBA algorithm and UICC supports MBMS)
760	var octetstring v_encrKeyData := fx_encrypt(v_csbId, v_randValue, v_ntpTimestamp, fx_mi	730	var octetstring v_encrKeyData := fx_encrypt(v_csbId, v_randValue, v_ntpTimestamp, fx_mikeyP
761	// hash calculation is done in the Mikey Codec	731	// hash calculation is done in the Mikey Codec
762	var KeyDataTransport v_kemac := valueof(m_def_KemacPayload(AES_CM_128, v_encrKeyData, H	732	var KeyDataTransport v_kemac := valueof(m_def_KemacPayload(AES_CM_128, v_encrKeyData, HMAC_
763	f_addMikeyPayload(v_payloads, {keyDataTransport := v_kemac});	733	f_addMikeyPayload(v_payloads, {keyDataTransport := v_kemac});
764		734	
765	// PSK needed for hash calculation	735	// PSK needed for hash calculation
766	var MikeyMessage v_sendMikey := valueof(m_def_Mikey(v_mikeyheader, v_payloads, {usedPSK	736	var MikeyMessage v_sendMikey := valueof(m_def_Mikey(v_mikeyheader, v_payloads, {usedPSK :=
767		737	
768	f_main_crtl_sendMikey(v_sendMikey);	738	f_main_crtl_sendMikey(v_sendMikey);
769		739	
770	// postamble	740	// postamble
771	f_main_ut_PowerOff();	741	f_main_ut_PowerOff();
772		742	
773	f_cf_Mikey_down();	743	f_cf_Mikey_down();
774	f_cf_BSF_down();	744	f_cf_BSF_down();
775	f_cf_BSM_down();	745	f_cf_BSM_down();
776	f_cf_DNS_down();	746	f_cf_DNS_down();
777	f_cf_Broadcast_down();	747	f_cf_Broadcast_down();
778	f_cf_UpperTester_down();	748	f_cf_UpperTester_down();
779	f_terminateComponents();	749	f_terminateComponents();

Datei: AtsBCast_ServiceProtectionTests.ttcn (Fortsetzung)

780	} else {			
781	log("Verdict Info: Preconditions are not meet. See PIXIT values 'PX_GBA' and 'PX_SUPPORT_MBMS_KEY_MANAGEMENT'");			
782	setverdict(inconc);			
783	}			
784	}	=	750	}
785			751	
786	// change 3 (WK34): Service Protection: secure streaming service		752	// change 3 (WK34): Service Protection: secure streaming service
787	testcase TC_BCAST_SERVPROT_101f() runs on BCastMainComponent system BCastPlatformComponent {		753	testcase TC_BCAST_SERVPROT_101f() runs on BCastMainComponent system BCastPlatformComponent {
788	// change 3 (WK46/08): Test if MBMS Key Management Support is available	<>		
789	if (PX_GBA == e_gba_u and PX_SUPPORT_MBMS_KEY_MANAGEMENT) {			
790	// init components and ports		754	// init components and ports
791	f_createComponents();		755	f_createComponents();
792	f_cf_Broadcast_up();		756	f_cf_Broadcast_up();
793	f_cf_Secure_Data_up();		757	f_cf_Secure_Data_up();
794	f_cf_Data_up();		758	f_cf_Data_up();
795	f_cf_UpperTester_up();		759	f_cf_UpperTester_up();
796	f_cf_DNS_up();		760	f_cf_DNS_up();
797	f_cf_BSM_up();		761	f_cf_BSM_up();
798	f_cf_BSF_up();		762	f_cf_BSF_up();
799	f_cf_Mikey_up();		763	f_cf_Mikey_up();
800			764	
801	// preamble		765	// preamble
802	f_cf_enable_DNS();		766	f_cf_enable_DNS();
803	f_startCommonUtPreamble();		767	f_startCommonUtPreamble();
804			768	
805	var charstring dstIP := PX_DATA_STREAMING_DESTINATION_IP_1;		769	var charstring dstIP := PX_DATA_STREAMING_DESTINATION_IP_1;
806			770	
807	var hexstring v_keyDomainId := PX_MCC & PX_MNC;		771	var hexstring v_keyDomainId := PX_MCC & PX_MNC;
808	var hexstring v_mskId := PX_KEY_GROUP & PX_KEY_NUMBER;		772	var hexstring v_mskId := PX_KEY_GROUP & PX_KEY_NUMBER;
809	var hexstring v_protectionKeyId := v_keyDomainId & v_mskId;		773	var hexstring v_protectionKeyId := v_keyDomainId & v_mskId;
810	var octetstring v_ks_NAF := f_main_ctrl_InitProtectionKeysAndValues();		774	var octetstring v_ks_NAF := f_main_ctrl_InitProtectionKeysAndValues();
811	f_main_ctrl_SetProtectionKeyId(v_protectionKeyId);		775	f_main_ctrl_SetProtectionKeyId(v_protectionKeyId);
812	var charstring v_btId := f_bsf_generateBtid();		776	var charstring v_btId := f_bsf_generateBtid();
813	f_main_ctrl_SetBootstrappingTransactionId(v_btId);		777	f_main_ctrl_SetBootstrappingTransactionId(v_btId);
814				
815	// GBA/BSF as parallel test component			
816	f_main_ctrl_GBA_Bootstrapping(true);			
817	// Bcast Subscription Managemant as parallel test component for MBMS User Service Registrat		778	// Bcast Subscription Managemant as parallel test component for MBMS User Service Registrat
818	// and Service Request/Response		779	// and Service Request/Response
819	f_main_ctrl_BSM_Procedures(true, true, true, true);		780	f_main_ctrl_BSM_Procedures(true, true, true, true);
			781	// GBA/BSF as parallel test component
			782	f_main_ctrl_GBA_U_Bootstrapping(true);
820			783	
821	// in order to create a purchase items		784	// in order to create a purchase items
822	// ESG with purchase part		785	// ESG with purchase part
823	var ServiceGuideDeliveryUnit v_SGDU := {};		786	var ServiceGuideDeliveryUnit v_SGDU := {};
824	var UInt32 v_Version := 1;		787	var UInt32 v_Version := 1;
825			788	
826	f_main_ctrl_InitStartTime(0, c_one_hour);		789	f_main_ctrl_InitStartTime(0, c_one_hour);
827			790	
828	var DateTime v_DateTime_StartTime := f_getDateTime(v_startTime);		791	var DateTime v_DateTime_StartTime := f_getDateTime(v_startTime);
829	var DateTime v_DateTime_EndTime := f_getDateTime(v_endTime);		792	var DateTime v_DateTime_EndTime := f_getDateTime(v_endTime);
830			793	
831	// TODO: SGDD should be extended with BSM selector > SDP extension for STKM need the		794	// TODO: SGDD should be extended with BSM selector > SDP extension for STKM need the
832	// reference to the BSM selector as declared in the SGDD		795	// reference to the BSM selector as declared in the SGDD
833	// SGDU access fragment need the reference to thd BSM selector as well		796	// SGDU access fragment need the reference to thd BSM selector as well
834			797	
835	var SdpFragment v_Sdp := valueof(m_def_SdpFragment(798	var SdpFragment v_Sdp := valueof(m_def_SdpFragment(
836	PX_SDP_VIDEO_PROG_1_REF_ID,		799	PX_SDP_VIDEO_PROG_1_REF_ID,
837	PX_SDP_SERVICE_PROTECTION_S RTP		800	PX_SDP_SERVICE_PROTECTION_S RTP
838));		801));
839			802	
840	var AccessTypeType v_AccessType := valueof(m_def_AccessType_bc_sdp(803	var AccessTypeType v_AccessType := valueof(m_def_AccessType_bc_sdp(
841	f_getAccessType(PX_BROADCAST_NETWORK_BEARER),		804	f_getAccessType(PX_BROADCAST_NETWORK_BEARER),
842	PX_SDP_VIDEO_PROG_1_REF_ID		805	PX_SDP_VIDEO_PROG_1_REF_ID
843));		806));
844			807	
845	var PriceInfoType v_PriceInfo := valueof(m_PriceInfo(808	var PriceInfoType v_PriceInfo := valueof(m_PriceInfo(
846	1,		809	1,
847	PX_MONETARY_PRICE,		810	PX_MONETARY_PRICE,
848	PX_CURRENCY		811	PX_CURRENCY

Datei: AtsBCast_ServiceProtectionTests.ttcn (Fortsetzung)

```

849     ));
850
851     var ServiceType v_Service := valueof(m_def_Service(
852         PX_SGDU_SERVICE_ID,
853         v_Version,
854         "PayTvChannel",
855         c_basicTv_ServiceType
856     ));
857
858     var ContentType v_Content := valueof(m_def_Content(
859         PX_SGDU_CONTENT_ID_PROG_1,
860         v_Version,
861         "Programme",
862         PX_SGDU_SERVICE_ID,
863         v_DateTime_StartTime,
864         v_DateTime_EndTime
865     ));
866
867     var ScheduleType v_Schedule := valueof(m_def_Schedule(
868         PX_SGDU_SCHEDULE_ID_PROG_1,
869         v_Version,
870         PX_SGDU_SERVICE_ID,
871         PX_SGDU_CONTENT_ID_PROG_1,
872         v_startTime,
873         v_endTime
874     ));
875
876     var AccessType v_Access := valueof(m_def_Access(
877         PX_SGDU_ACCESS_ID_PROG_1,
878         v_Version,
879         v_AccessType,
880         PX_SGDU_SERVICE_ID,
881         PX_SGDU_SCHEDULE_ID_PROG_1,
882         c_class_SG
883     ));
884     v_Access.KeyManagementSystems := {
885     {
886         kmsType := 1, // 1 = oma-bcast-gba_u-mbms
887         protectionType := 0, // Content protection only, as defined by the LTKM
888         PermissionsIssuerURI := {
889             text_ := "http://" & PX_BSM_FQDN
890         },
891         ProtectionKeyIDs := {
892         {
893             type_ := 0, // ProtectionKeyID = Key Domain ID concatenated with SEK/PE
894             // key domain id = MCC || MNC
895             // sek id = key group || key number
896             text_ := fx_bitstring2Base64(hex2bit(v_protectionKeyId)) // Base64 encoded
897         }
898         },
899         TerminalBindingKeyID := omit
900     }
901     };
902     // TODO: set BSM selector like in SGDD
903     // v_Access.ServiceClass.ReferredSGInfo.BSMSelector
904
905     var PurchaseItemType v_PurchaseItem := valueof(m_def_PurchaseItem(
906         PX_SGDU_PURCHASE_ITEM_ID,
907         v_Version,
908         PX_SGDU_PURCHASE_DATA_ID,
909         PX_SGDU_SERVICE_ID,
910         "PurchaseItem1"
911     ));
912     v_PurchaseItem.Descriptions := {{lang := omit, text_ := "Purchasable SRTPEncrypted service"}
913
914     var PurchaseDataType v_PurchaseData := valueof(m_def_PurchaseData(
915         PX_SGDU_PURCHASE_DATA_ID,
916         v_Version,
917         {{omit, "Discount price available"}},
918         PX_SGDU_PURCHASE_ITEM_ID,
919         PX_SGDU_PURCHASE_CHANNEL_ID,

```

```

812   ));
813
814   var ServiceType v_Service := valueof(m_def_Service(
815     PX_SGDU_SERVICE_ID,
816     v_Version,
817     "PayTvChannel",
818     c_basicTv_ServiceType
819   ));
820
821   var ContentType v_Content := valueof(m_def_Content(
822     PX_SGDU_CONTENT_ID_PROG_1,
823     v_Version,
824     "Programme",
825     PX_SGDU_SERVICE_ID,
826     v_DateTime_StartTime,
827     v_DateTime_EndTime
828   ));
829
830   var ScheduleType v_Schedule := valueof(m_def_Schedule(
831     PX_SGDU_SCHEDULE_ID_PROG_1,
832     v_Version,
833     PX_SGDU_SERVICE_ID,
834     PX_SGDU_CONTENT_ID_PROG_1,
835     v_startTime,
836     v_endTime
837   ));
838
839   var AccessType v_Access := valueof(m_def_Access(
840     PX_SGDU_ACCESS_ID_PROG_1,
841     v_Version,
842     v_AccessType,
843     PX_SGDU_SERVICE_ID,
844     PX_SGDU_SCHEDULE_ID_PROG_1,
845     c_class_SG
846   ));
847   v_Access.KeyManagementSystems := {
848     {
849       kmsType := 1, // 1 = oma-bcast-gba_u-mbms
850       protectionType := 0, // Content protection only, as defined by the LTKM
851       PermissionsIssuerURI := {
852         text_ := "http://" & PX_BSM_FQDN
853       },
854       ProtectionKeyIDs := {
855         {
856           type_ := 0, // ProtectionKeyID = Key Domain ID concatenated with SEK/PEK ID
857           // key domain id = MCC || MNC
858           // sek id = key group || key number
859           text_ := fx_bitstring2Base64(hex2bit(v_protectionKeyId)) // Base64 encoding
860         }
861       },
862       TerminalBindingKeyID := omit
863     }
864   };
865   // TODO: set BSM selector like in SGDD
866   // v_Access.ServiceClass.ReferredSGInfo.BSMSSelector
867
868   var PurchaseItemType v_PurchaseItem := valueof(m_def_PurchaseItem(
869     PX_SGDU_PURCHASE_ITEM_ID,
870     v_Version,
871     PX_SGDU_PURCHASE_DATA_ID,
872     PX_SGDU_SERVICE_ID,
873     "PurchaseItem1"
874   ));
875   v_PurchaseItem.Descriptions := {{lang := omit, text_ := "Purchasable SRTPencrypted service"}}
876
877   var PurchaseDataType v_PurchaseData := valueof(m_def_PurchaseData(
878     PX_SGDU_PURCHASE_DATA_ID,
879     v_Version,
880     {{omit, "Discount price available"}}},
881     PX_SGDU_PURCHASE_ITEM_ID,
882     PX_SGDU_PURCHASE_CHANNEL_ID,

```

Datei: AtsBCast_ServiceProtectionTests.ttcn (Fortsetzung)

920	v_PriceInfo	883	v_PriceInfo
921));	884));
922		885	
923	var PurchaseChannelType v_PurchaseChannel := valueof(m_def_PurchaseChannel(886	var PurchaseChannelType v_PurchaseChannel := valueof(m_def_PurchaseChannel(
924	PX_SGDU_PURCHASE_CHANNEL_ID,	887	PX_SGDU_PURCHASE_CHANNEL_ID,
925	v_Version,	888	v_Version,
926	"PurchaseChannel",	889	"PurchaseChannel",
927	"http://" & PX_BSM_FQDN & "/bsm/purchaseURL"	890	"http://" & PX_BSM_FQDN & "/bsm/purchaseURL"
928));	891));
929		892	
930	f_addXMLFragment(v_SGDU, {Service := v_Service});	893	f_addXMLFragment(v_SGDU, {Service := v_Service});
931	f_addXMLFragment(v_SGDU, {Content := v_Content});	894	f_addXMLFragment(v_SGDU, {Content := v_Content});
932	f_addXMLFragment(v_SGDU, {Schedule := v_Schedule});	895	f_addXMLFragment(v_SGDU, {Schedule := v_Schedule});
933	f_addXMLFragment(v_SGDU, {Access := v_Access});	896	f_addXMLFragment(v_SGDU, {Access := v_Access});
934	f_addXMLFragment(v_SGDU, {PurchaseItem := v_PurchaseItem});	897	f_addXMLFragment(v_SGDU, {PurchaseItem := v_PurchaseItem});
935	f_addXMLFragment(v_SGDU, {PurchaseChannel := v_PurchaseChannel});	898	f_addXMLFragment(v_SGDU, {PurchaseChannel := v_PurchaseChannel});
936	f_addXMLFragment(v_SGDU, {PurchaseData := v_PurchaseData});	899	f_addXMLFragment(v_SGDU, {PurchaseData := v_PurchaseData});
937	f_addSdpFragment(v_SGDU, v_Version, v_Sdp);	900	f_addSdpFragment(v_SGDU, v_Version, v_Sdp);
938		901	
939	f_main_ctrl_broadcastServiceGuide(v_SGDU, omit, vc_FluteUpdateID);	902	f_main_ctrl_broadcastServiceGuide(v_SGDU, omit, vc_FluteUpdateID);
940		903	
941	f_main_ut_RunBCastApplication();	904	f_main_ut_RunBCastApplication();
942		905	
943	f_main_ut_CheckService("PayTvChannel");	906	f_main_ut_CheckService("PayTvChannel");
944	f_main_ut_SelectService("PayTvChannel");	907	f_main_ut_SelectService("PayTvChannel");
945	f_main_ut_CheckContent("Programme with price information", v_DateTime_StartTime, v_DateTime_EndTime);	908	f_main_ut_CheckContent("Programme with price information", v_DateTime_StartTime, v_DateTime_EndTime);
946	f_main_ut_PurchaseService("PayTvChannel");	909	f_main_ut_PurchaseService("PayTvChannel");
947		910	
948	// preparation for LTKM	911	// preparation for LTKM
949	var UInt8 v_PolicyNumber := 0;	912	var UInt8 v_PolicyNumber := 0;
950	var octetstring v_randValue := '6813b3758c1cfe138d8b2ba886d3f449'O;	913	var octetstring v_randValue := '6813b3758c1cfe138d8b2ba886d3f449'O;
951	var octetstring v_csbId := '00000000'O;	914	var octetstring v_csbId := '00000000'O;
952	var Oct4 v_ssrc := '7BE6AC2E'O;	915	var Oct4 v_ssrc := '7BE6AC2E'O;
953		916	
954	var ListOfSrtpId v_SrtpIds := {}; // empty list	917	var ListOfSrtpId v_SrtpIds := {}; // empty list
955	var SrtpId v_SrtpId := valueof(m_def_SrtpId(v_PolicyNumber, v_ssrc, '00000000'O));	918	var SrtpId v_SrtpId := valueof(m_def_SrtpId(v_PolicyNumber, v_ssrc, '00000000'O));
956	f_addSrtpId(v_SrtpIds, v_SrtpId);	919	f_addSrtpId(v_SrtpIds, v_SrtpId);
957		920	
958	var MikeyHeaderPayload v_mikeyheader := valueof(m_def_LTKMHeader(GeneralExt, v_csbId, v_SrtpId));	921	var MikeyHeaderPayload v_mikeyheader := valueof(m_def_LTKMHeader(GeneralExt, v_csbId, v_SrtpId));
959		922	
960	var ListOfPayload v_payloads4ltkm := {}; // empty list	923	var ListOfPayload v_payloads4ltkm := {}; // empty list
961		924	
962	var KeyId v_keyIdKeyDomianId := valueof(m_def_KeyId_KeyDomainId(hex2oct(v_keyDomainId)));	925	var KeyId v_keyIdKeyDomianId := valueof(m_def_KeyId_KeyDomainId(hex2oct(v_keyDomainId)));
963	var KeyId v_keyIdMskId := valueof(m_def_KeyId_MSKId(hex2oct(v_mskId)));	926	var KeyId v_keyIdMskId := valueof(m_def_KeyId_MSKId(hex2oct(v_mskId)));
964	var GeneralExtension v_ExtMBMS := valueof(m_def_GeneralExtension(KEY_ID, {keyIds := {v_keyIdKeyDomianId, v_keyIdMskId}}));	927	var GeneralExtension v_ExtMBMS := valueof(m_def_GeneralExtension(KEY_ID, {keyIds := {v_keyIdKeyDomianId, v_keyIdMskId}}));
965	f_addMikeyPayload(v_payloads4ltkm, {generalExtension := v_ExtMBMS});	928	f_addMikeyPayload(v_payloads4ltkm, {generalExtension := v_ExtMBMS});
966		929	
967	var BcastExtension v_bcastExt := valueof(m_def_BcastExtension_LTKM(m_def_LTKMManagementData, v_keyIdKeyDomianId, v_keyIdMskId));	930	var BcastExtension v_bcastExt := valueof(m_def_BcastExtension_LTKM(m_def_LTKMManagementData, v_keyIdKeyDomianId, v_keyIdMskId));
968	var GeneralExtension v_ExtBCAST := valueof(m_def_GeneralExtension(OMA_BCAST, {bcastExtension := v_bcastExt}));	931	var GeneralExtension v_ExtBCAST := valueof(m_def_GeneralExtension(OMA_BCAST, {bcastExtension := v_bcastExt}));
969	f_addMikeyPayload(v_payloads4ltkm, {generalExtension := v_ExtBCAST});	932	f_addMikeyPayload(v_payloads4ltkm, {generalExtension := v_ExtBCAST});
970		933	
971	var integer v_ntpTimestamp := f_getNTPTime();	934	var integer v_ntpTimestamp := f_getNTPTime();
972	var Timestamp v_Timestamp := valueof(m_def_TimestampPayloadData(NTP.UTC, v_ntpTimestamp));	935	var Timestamp v_Timestamp := valueof(m_def_TimestampPayloadData(NTP.UTC, v_ntpTimestamp));
973	f_addMikeyPayload(v_payloads4ltkm, {timestamp := v_Timestamp});	936	f_addMikeyPayload(v_payloads4ltkm, {timestamp := v_Timestamp});
974		937	
975	var Rand v_Rand := valueof(m_def_RandPayloadData(v_randValue));	938	var Rand v_Rand := valueof(m_def_RandPayloadData(v_randValue));
976	f_addMikeyPayload(v_payloads4ltkm, {rand := v_Rand});	939	f_addMikeyPayload(v_payloads4ltkm, {rand := v_Rand});
977		940	
978	var Id v_Idi := valueof(m_def_IdPayloadData(URI, PX_BSM_FQDN));	941	var Id v_Idi := valueof(m_def_IdPayloadData(URI, PX_BSM_FQDN));
979	f_addMikeyPayload(v_payloads4ltkm, {id := v_Idi});	942	f_addMikeyPayload(v_payloads4ltkm, {id := v_Idi});
980		943	
981	var Id v_Idr := valueof(m_def_IdPayloadData(NAI, v_btId));	944	var Id v_Idr := valueof(m_def_IdPayloadData(NAI, v_btId));
982	f_addMikeyPayload(v_payloads4ltkm, {id := v_Idr});	945	f_addMikeyPayload(v_payloads4ltkm, {id := v_Idr});
983		946	
984	// var ListOfSrtpPolicyParam v_SrtpPolicyParams := {}; //empty list	947	// var ListOfSrtpPolicyParam v_SrtpPolicyParams := {}; //empty list
985	// var SrtpPolicyParameter v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTP_AUTH_ALGORITHM, 1, {srtpAuthAlgorithm := v_srtpAuthAlgorithm}));	948	// var SrtpPolicyParameter v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTP_AUTH_ALGORITHM, 1, {srtpAuthAlgorithm := v_srtpAuthAlgorithm}));
986	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);	949	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);
987	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SESSION_ENCR_KEY_LENGTH, 1, {sessionEncrKeyLength := v_sessionEncrKeyLength}));	950	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SESSION_ENCR_KEY_LENGTH, 1, {sessionEncrKeyLength := v_sessionEncrKeyLength}));
988	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);	951	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);
989	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTP_AUTH_ALGORITHM, 1, {srtpAuthAlgorithm := v_srtpAuthAlgorithm}));	952	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTP_AUTH_ALGORITHM, 1, {srtpAuthAlgorithm := v_srtpAuthAlgorithm}));
990	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);	953	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);

Datei: AtsBCast_ServiceProtectionTests.ttcn (Fortsetzung)

991	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SESSION_AUTH_KEY_LENGTH, 1, {	954	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SESSION_AUTH_KEY_LEN
992	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);	955	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);
993	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SESSION_SALT_KEY_LENGTH, 1, {	956	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SESSION_SALT_KEY_LEN
994	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);	957	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);
995	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTP_PRF, 1, {srtpPrf := SRTP	958	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTP_PRF, 1, {srtpPr
996	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);	959	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);
997	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(KEY_DERIVATION_RATE, 1, {keyD	960	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(KEY_DERIVATION_RATE,
998	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);	961	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);
999	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTP_ENCR_OFF_ON, 1, {srtpEncr	962	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTP_ENCR_OFF_ON, 1,
1000	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);	963	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);
1001	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTP_ENCR_OFF_ON, 1, {srtpEn	964	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTP_ENCR_OFF_ON, 1,
1002	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);	965	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);
1003	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SENDER_FEC_ORDER, 1, {fecOrder	966	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SENDER_FEC_ORDER, 1,
1004	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);	967	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);
1005	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTP_AUTH_OFF_ON, 1, {srtpAut	968	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTP_AUTH_OFF_ON, 1,
1006	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);	969	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);
1007	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(AUTH_TAG_LENGTH, 1, {authTagL	970	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(AUTH_TAG_LENGTH, 1,
1008	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);	971	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);
1009	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTP_PREFIX_LENGTH, 1, {srtpP	972	// v_SrtpPolicyParam := valueof(m_def_SrtpPolicyParameter(SRTP_PREFIX_LENGTH,
1010	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);	973	// f_addSrtpPolicyParameter(v_SrtpPolicyParams, v_SrtpPolicyParam);
1011	// var SecurityPolicy v_SecurityPolicy := valueof(m_def_SecurityPolicy4Srtp(v_PolicyNum	974	// var SecurityPolicy v_SecurityPolicy := valueof(m_def_SecurityPolicy4Srtp
1012	// f_addMikeyPayload(v_payloads, {securityPolicy := v_SecurityPolicy});	975	// f_addMikeyPayload(v_payloads, {securityPolicy := v_SecurityPolicy});
1013		976	
1014	// preparations for encrypting of LTKM included in KEMAC	977	// preparations for encrypting of LTKM included in KEMAC
1015	var Interval v_interval := valueof(m_def_Interval(PX_TS_LOW, PX_TS_HIGH));	978	var Interval v_interval := valueof(m_def_Interval(PX_TS_LOW, PX_TS_HIGH));
1016	var Key v_keyData := valueof(m_def_KeyData4LTKM(KV_INTERVAL, PX_LTKM, {interval := v_in	979	var Key v_keyData := valueof(m_def_KeyData4LTKM(KV_INTERVAL, PX_LTKM, {interval := v_interv
1017	var MikeyPayload v_keyDataPayload := valueof(m_def_Payload(LastPayload, {key := v_keyDa	980	var MikeyPayload v_keyDataPayload := valueof(m_def_Payload(LastPayload, {key := v_keyData}}
1018	// PSK is MUK is ks (ext/int) NAF (depending of used GBA algorithm and UICC supports MB	981	// PSK is MUK is ks (ext/int) NAF (depending of used GBA algorithm and UICC supports MBMS)
1019	var octetstring v_encrKeyData := fx_encrypt(v_csbId, v_randValue, v_ntpTimestamp, fx_mi	982	var octetstring v_encrKeyData := fx_encrypt(v_csbId, v_randValue, v_ntpTimestamp, fx_mikeyP
1020	// hash calculation is done in the Mikey Codec	983	// hash calculation is done in the Mikey Codec
1021	var KeyDataTransport v_kemac := valueof(m_def_KemacPayload(AES_CM_128, v_encrKeyData, H	984	var KeyDataTransport v_kemac := valueof(m_def_KemacPayload(AES_CM_128, v_encrKeyData, HMAC
1022	f_addMikeyPayload(v_payloads4ltkm, {keyDataTransport := v_kemac});	985	f_addMikeyPayload(v_payloads4ltkm, {keyDataTransport := v_kemac});
1023		986	
1024	// PSK needed for hash calculation	987	// PSK needed for hash calculation
1025	var MikeyMessage v_sendMikeyLTKM := valueof(m_def_Mikey(v_mikeyheader, v_payloads4ltkm,	988	var MikeyMessage v_sendMikeyLTKM := valueof(m_def_Mikey(v_mikeyheader, v_payloads4ltkm, {us
1026		989	
1027	// preparation for STKMs	990	// preparation for STKMs
1028	v_mikeyheader := valueof(m_def_STKMHeader(GeneralExt, v_csbId));	991	v_mikeyheader := valueof(m_def_STKMHeader(GeneralExt, v_csbId));
1029		992	
1030	var ListOfPayload v_payloads4stkm := {}; // empty list	993	var ListOfPayload v_payloads4stkm := {}; // empty list
1031		994	
1032	var octetstring v_mtkId := PX_MTK_ID_START;	995	var octetstring v_mtkId := PX_MTK_ID_START;
1033	var KeyId v_keyIdMtkId := valueof(m_def_KeyId_MTKId(v_mtkId));	996	var KeyId v_keyIdMtkId := valueof(m_def_KeyId_MTKId(v_mtkId));
1034	v_ExtMBMS := valueof(m_def_GeneralExtension(KEY_ID, {keyIds := {v_keyIdKeyDomianId, v_	997	v_ExtMBMS := valueof(m_def_GeneralExtension(KEY_ID, {keyIds := {v_keyIdKeyDomianId, v_keyId
1035	f_addMikeyPayload(v_payloads4stkm, {generalExtension := v_ExtMBMS});	998	f_addMikeyPayload(v_payloads4stkm, {generalExtension := v_ExtMBMS});
1036		999	
1037	v_bcastExt := valueof(m_def_BcastExtension_STKM(m_def_STKMManagementData()));	1000	v_bcastExt := valueof(m_def_BcastExtension_STKM(m_def_STKMManagementData()));
1038	v_ExtBCAST := valueof(m_def_GeneralExtension(OMA_BCAST, {bcastExtension := v_bcastExt}));	1001	v_ExtBCAST := valueof(m_def_GeneralExtension(OMA_BCAST, {bcastExtension := v_bcastExt}));
1039	f_addMikeyPayload(v_payloads4stkm, {generalExtension := v_ExtBCAST});	1002	f_addMikeyPayload(v_payloads4stkm, {generalExtension := v_ExtBCAST});
1040		1003	
1041	v_Timestamp := valueof(m_def_TimestampPayloadData(COUNTER, oct2int(PX_TS_LOW) + 1));	1004	v_Timestamp := valueof(m_def_TimestampPayloadData(COUNTER, oct2int(PX_TS_LOW) + 1));
1042	f_addMikeyPayload(v_payloads4stkm, {timestamp := v_Timestamp});	1005	f_addMikeyPayload(v_payloads4stkm, {timestamp := v_Timestamp});
1043		1006	
1044	// preparations for encrypting of STKM included in KEMAC	1007	// preparations for encrypting of STKM included in KEMAC
1045	// PSK is key of LTKM	1008	// PSK is key of LTKM
1046	var octetstring v_salt := fx_mikeySaltKey(v_csbId, v_randValue, PX_LTKM, PX_SALT_KEY_LE	1009	var octetstring v_salt := fx_mikeySaltKey(v_csbId, v_randValue, PX_LTKM, PX_SALT_KEY_LENGTH
1047	v_keyData := valueof(m_def_KeyData4STKM(PX_SRTP_MASTER_KEY_START, v_salt));	1010	v_keyData := valueof(m_def_KeyData4STKM(PX_SRTP_MASTER_KEY_START, v_salt));
1048	v_keyDataPayload := valueof(m_def_Payload(LastPayload, {key := v_keyData}));	1011	v_keyDataPayload := valueof(m_def_Payload(LastPayload, {key := v_keyData}));
1049		1012	
1050	v_encrKeyData := fx_encrypt(v_csbId, v_randValue, v_ntpTimestamp, fx_mikeyPayload2Oct(v_	1013	v_encrKeyData := fx_encrypt(v_csbId, v_randValue, v_ntpTimestamp, fx_mikeyPayload2Oct(v_key
1051	// hash calculation is done in the Mikey Codec	1014	// hash calculation is done in the Mikey Codec
1052	v_kemac := valueof(m_def_KemacPayload(AES_CM_128, v_encrKeyData, HMAC_SHA_1_160));	1015	v_kemac := valueof(m_def_KemacPayload(AES_CM_128, v_encrKeyData, HMAC_SHA_1_160));
1053	f_addMikeyPayload(v_payloads4stkm, {keyDataTransport := v_kemac});	1016	f_addMikeyPayload(v_payloads4stkm, {keyDataTransport := v_kemac});
1054		1017	
1055	// PSK and RAND from LTKM needed for hash calculation	1018	// PSK and RAND from LTKM needed for hash calculation
1056	// RAND payload is not send again in STKM	1019	// RAND payload is not send again in STKM
1057	var MikeyMessage v_sendMikeySTKM := valueof(m_def_Mikey(v_mikeyheader, v_payloads4stkm,	1020	var MikeyMessage v_sendMikeySTKM := valueof(m_def_Mikey(v_mikeyheader, v_payloads4stkm, {us
1058		1021	
1059	// generates all keys delivered in STKM	1022	// generates all keys delivered in STKM
1060	var ListOfSTKMKeys v_stkmKeys := {};	1023	var ListOfSTKMKeys v_stkmKeys := {};
1061	var integer numOfKeys := 15;	1024	var integer numOfKeys := 15;

Datei: AtsBCast_ServiceProtectionTests.ttcn (Fortsetzung)

1062	for (var integer i := 0; i < numOfKeys; i := i + 1) {		1025	for (var integer i := 0; i < numOfKeys; i := i + 1) {	
1063	v_stkmKeys[i] := {int2oct((oct2int(PX_SRTP_MASTER_KEY_START) + i), 16), v_salt}		1026	v_stkmKeys[i] := {int2oct((oct2int(PX_SRTP_MASTER_KEY_START) + i), 16), v_salt}	
1064	}		1027	}	
1065			1028		
1066	// generates remaining STKMs (updates the timestamp in every STKM; replaces the master		1029	// generates remaining STKMs (updates the timestamp in every STKM; replaces the master key	
1067	var ListOfMikeyMessage v_stkms := {v_sendMikeySTKM};		1030	var ListOfMikeyMessage v_stkms := {v_sendMikeySTKM};	
1068	var integer renewalOfKeys := 10; // the master key changes every 10th STKM		1031	var integer renewalOfKeys := 10; // the master key changes every 10th STKM	
1069	for (var integer i := 1; i < numOfKeys * renewalOfKeys; i := i + 1) {		1032	for (var integer i := 1; i < numOfKeys * renewalOfKeys; i := i + 1) {	
1070	v_stkms[i] := f_updateTSinSTKM(v_stkms[i - 1]);		1033	v_stkms[i] := f_updateTSinSTKM(v_stkms[i - 1]);	
1071	if (i mod renewalOfKeys == 0) {		1034	if (i mod renewalOfKeys == 0) {	
1072	v_stkms[i] := f_updateKEYinSTKM(v_csbId, v_ntpTimestamp, v_stkmKeys[i / renewalOfKeys]);		1035	v_stkms[i] := f_updateKEYinSTKM(v_csbId, v_ntpTimestamp, v_stkmKeys[i / renewalOfKeys]);	
1073	}				
1074	}	=	1036	}	
		<>	1037	}	
1075			1038		
1076	// sends LTKM		1039	// sends LTKM	
1077	f_main_ctrl_sendMikey(v_sendMikeyLTKM);		1040	f_main_ctrl_sendMikey(v_sendMikeyLTKM);	
1078			1041		
1079	f_main_data_secure_setKeys(v_stkmKeys);		1042	f_main_data_secure_setKeys(v_stkmKeys);	
1080			1043		
1081	f_main_data_secure_startStreamingFile(0, PX_FILE_VIDEO_PROG1, PX_SDP_SERVICE_PROTECTION_SRTP);		1044	f_main_data_secure_startStreamingFile(0, PX_FILE_VIDEO_PROG1, PX_SDP_SERVICE_PROTECTION_SRTP);	
1082			1045		
1083	// sends STKMs		1046	// sends STKMs	
1084	timer v_nextSTKMTimer := 1.0;		1047	timer v_nextSTKMTimer := 1.0;	
1085	var integer v_currentKeyNumber := 0;		1048	var integer v_currentKeyNumber := 0;	
1086	for (var integer i := 0; i < numOfKeys * renewalOfKeys; i := i + 1) {		1049	for (var integer i := 0; i < numOfKeys * renewalOfKeys; i := i + 1) {	
1087	// after sending 10 STKMs the next key must be used		1050	// after sending 10 STKMs the next key must be used	
1088	if (i mod 10 == 0) {		1051	if (i mod 10 == 0) {	
1089	f_main_data_secure_activateKey(v_stkmKeys[v_currentKeyNumber], v_currentKeyNumber);		1052	f_main_data_secure_activateKey(v_stkmKeys[v_currentKeyNumber], v_currentKeyNumber);	
1090	v_currentKeyNumber := v_currentKeyNumber + 1;		1053	v_currentKeyNumber := v_currentKeyNumber + 1;	
1091	}		1054	}	
1092	v_nextSTKMTimer.start;		1055	v_nextSTKMTimer.start;	
1093	f_main_ctrl_broadcastMikey(v_stkms[i]);		1056	f_main_ctrl_broadcastMikey(v_stkms[i]);	
1094	v_nextSTKMTimer.timeout;		1057	v_nextSTKMTimer.timeout;	
1095	}		1058	}	
1096			1059		
1097	f_main_data_secure_stopStreaming(0);		1060	f_main_data_secure_stopStreaming(0);	
1098			1061		
1099	// postamble		1062	// postamble	
1100	f_main_ut_PowerOff();		1063	f_main_ut_PowerOff();	
1101			1064		
1102	f_cf_Mikey_down();		1065	f_cf_Mikey_down();	
1103	f_cf_BSF_down();		1066	f_cf_BSF_down();	
1104	f_cf_BSM_down();		1067	f_cf_BSM_down();	
1105	f_cf_DNS_down();		1068	f_cf_DNS_down();	
1106	f_cf_Data_down();		1069	f_cf_Data_down();	
1107	f_cf_Secure_Data_down();		1070	f_cf_Secure_Data_down();	
1108	f_cf_Broadcast_down();		1071	f_cf_Broadcast_down();	
1109	f_cf_UpperTester_down();		1072	f_cf_UpperTester_down();	
1110	f_terminateComponents();		1073	f_terminateComponents();	
1111	} else {				
1112	log("Verdict Info: Preconditions are not meet. See PIXIT values 'PX_GBA' and 'PX_SUPPORTS_SECURE_STREAMING'");				
1113	setverdict(inconc);				
1114	}				
1115	}	=	1074	}	
1116	}		1075	}	
1117			1076		
1118	// change 3 (WK34): Service Protection: secure streaming service		1077	// change 3 (WK34): Service Protection: secure streaming service	
1119	group Function {		1078	group Function {	
1120	/**		1079	/**	
1121	*		1080	*	
1122	* @desc increases TS value by 1 in STKM		1081	* @desc increases TS value by 1 in STKM	
1123	* @param p_currentKey		1082	* @param p_currentKey	
1124	* @param p_stkm		1083	* @param p_stkm	
1125	* @return		1084	* @return	
1126	* @verdict		1085	* @verdict	
1127	*/		1086	*/	
1128	function f_updateTSinSTKM(in MikeyMessage p_stkm) return MikeyMessage {		1087	function f_updateTSinSTKM(in MikeyMessage p_stkm) return MikeyMessage {	
1129	var integer v_size := sizeof(p_stkm.payloads);		1088	var integer v_size := sizeof(p_stkm.payloads);	
1130	var MikeyMessage v_mikey := p_stkm;		1089	var MikeyMessage v_mikey := p_stkm;	
1131		<>	1090		

Datei: AtsBCast_ServiceProtectionTests.ttcn (Fortsetzung)

1132	// updates the timestamp	=	1091	// updates the timestamp
1133	for (var integer i := 0; i < v_size; i := i + 1) {		1092	for (var integer i := 0; i < v_size; i := i + 1) {
1134	if (ischosen(v_mikey.payloads[i].payloadData.timestamp)) {		1093	if (ischosen(v_mikey.payloads[i].payloadData.timestamp)) {
1135	v_mikey.payloads[i].payloadData.timestamp.tsValue := v_mikey.payloads[i].payloadData.timestamp.tsValue		1094	v_mikey.payloads[i].payloadData.timestamp.tsValue := v_mikey.payloads[i].payloadData.timestamp.tsValue
1136	i := v_size + 1; // stops the loop		1095	i := v_size + 1; // stops the loop
1137	}		1096	}
1138	}		1097	}
1139		<>	1098	
1140	return v_mikey;	=	1099	return v_mikey;
1141	}		1100	}
1142			1101	
1143	/**		1102	/**
1144	*		1103	*
1145	* @desc replaces the Key Data in STKM		1104	* @desc replaces the Key Data in STKM
1146	* @param p_currentKey		1105	* @param p_currentKey
1147	* @param p_stkm		1106	* @param p_stkm
1148	* @return		1107	* @return
1149	* @verdict		1108	* @verdict
1150	*/		1109	*/
1151	function f_updateKEYinSTKM(in octetstring p_csbId, in integer p_timestamp, in SecureStreamingKey p_key, in integer p_size, in integer p_index) return SecureStreamingKey		1110	function f_updateKEYinSTKM(in octetstring p_csbId, in integer p_timestamp, in SecureStreamingKey p_key, in integer p_size, in integer p_index) return SecureStreamingKey
1152	// the key to be replaced is stored in the encrypted key data of the KEMAC payload		1111	// the key to be replaced is stored in the encrypted key data of the KEMAC payload
1153	// KEMAC payload is always the last payload in an STKM		1112	// KEMAC payload is always the last payload in an STKM
1154	p_stkm.payloads[sizeof(p_stkm.payloads) - 1].payloadData.keyDataTransport.encrData := fx_encrypt(p_key, p_stkm.payloads[sizeof(p_stkm.payloads) - 1].payloadData.keyDataTransport.plaintextData, p_timestamp)		1113	p_stkm.payloads[sizeof(p_stkm.payloads) - 1].payloadData.keyDataTransport.encrData := fx_encrypt(p_key, p_stkm.payloads[sizeof(p_stkm.payloads) - 1].payloadData.keyDataTransport.plaintextData, p_timestamp)
1155	return p_stkm;		1114	return p_stkm;
1156	}		1115	}
1157	}		1116	}
1158			1117	
1159	// change 3 (WK34): Service Protection: secure streaming service		1118	// change 3 (WK34): Service Protection: secure streaming service
1160	group externalFunctions {		1119	group externalFunctions {
1161	external function fx_mikeyPayload2Oct(in MikeyPayload p_payload) return octetstring;		1120	external function fx_mikeyPayload2Oct(in MikeyPayload p_payload) return octetstring;
1162	external function fx_encrypt(in octetstring p_csbId, in octetstring p_rand, in integer p_timestamp, in octetstring p_data) return octetstring;		1121	external function fx_encrypt(in octetstring p_csbId, in octetstring p_rand, in integer p_timestamp, in octetstring p_data) return octetstring;
1163	external function fx_decrypt(in octetstring p_csbId, in octetstring p_rand, in integer p_timestamp, in octetstring p_data) return octetstring;		1122	external function fx_decrypt(in octetstring p_csbId, in octetstring p_rand, in integer p_timestamp, in octetstring p_data) return octetstring;
1164	external function fx_mikeySaltKey(in octetstring p_csbid, in octetstring p_rand, in octetstring p_data) return octetstring;		1123	external function fx_mikeySaltKey(in octetstring p_csbid, in octetstring p_rand, in octetstring p_data) return octetstring;
1165	external function fx_oct2mikeyPayload(in octetstring p_mikeyPayload) return MikeyPayload;		1124	external function fx_oct2mikeyPayload(in octetstring p_mikeyPayload) return MikeyPayload;
1166	}		1125	}
1167	}		1126	}
1168	}		1127	}

Datei: LibCommon_BSM.ttcn

<pre>1 //change 1 (WK18): for content protection tests 2 module LibCommon_BSM { 3 import from LibCommon_HTTP_TypesAndValues all; 4 import from LibCommon_DataStrings all; 5 import from AtsBCast_ModuleParameters { 6 group GBA; 7 group BSM; 8 } 9 import from LibCommon_GBA_BSF all; 10 import from LibCommon_BSM_TypesAndValues all; 11 import from LibCommon_Time all; 12 // change 1 (WK23): Service Request extensions for content protection tests 13 import from LibCommon_GBA_BSF_TypesAndValues all; 14 // change 1 (WK23): Service Request extensions for content protection tests 15 import from LibCommon_BasicTypesAndValues all; 16 17 group BSM_Registration { 18 group BSM_Templates { 19 group BSM_HTTP_Templates { 20 group RequestTemplates { 21 template HttpRequest mw_registerAllServicesHttpRequest modifies mw 22 UriParameters := { 23 {Parameter := "requesttype", Value := "register"}, 24 * 25 }, 26 ContentType := { 27 // see TS 26.346 28 Name := "Content-Type", ListOfValue := {{Va 29 }, 30 Authorization := omit, 31 Body := { 32 //OMA Services 5.1.6.7 Registration Procedure 'oma 33 mbmsSecurityRegister := mw_simpleMbmsSecurityRegist 34 } 35 } 36 37 // change 1 (WK23): Service Request extensions for content protecti 38 // MBMS User Service Registration extensions 39 template HttpRequest mw_registerAllServicesHttpRequestWithAuthoriza 40 Authorization := { 41 Value := "Digest", 42 ParameterList := { 43 {Name := "response", Value := ?}, 44 {Name := "username", Value := p_username}, 45 *</pre>	=	<pre>1 //change 1 (WK18): for content protection tests 2 module LibCommon_BSM { 3 import from LibCommon_HTTP_TypesAndValues all; 4 import from LibCommon_DataStrings all; 5 import from AtsBCast_ModuleParameters { 6 group GBA; 7 group BSM; 8 } 9 import from LibCommon_GBA_BSF all; 10 import from LibCommon_BSM_TypesAndValues all; 11 import from LibCommon_Time all; 12 // change 1 (WK23): Service Request extensions for content protection tests 13 import from LibCommon_GBA_BSF_TypesAndValues all; 14 // change 1 (WK23): Service Request extensions for content protection tests 15 import from LibCommon_BasicTypesAndValues all; 16 17 group BSM_Registration { 18 group BSM_Templates { 19 group BSM_HTTP_Templates { 20 group RequestTemplates { 21 template HttpRequest mw_registerAllServicesHttpRequest modifies mw 22 UriParameters := { 23 {Parameter := "requesttype", Value := "register"}, 24 * 25 }, 26 ContentType := { 27 // see TS 26.346 28 Name := "Content-Type", ListOfValue := {{Va 29 }, 30 Authorization := omit, 31 Body := { 32 //OMA Services 5.1.6.7 Registration Procedure 'oma 33 mbmsSecurityRegister := mw_simpleMbmsSecurityRegist 34 } 35 } 36 37 // change 1 (WK23): Service Request extensions for content protecti 38 // MBMS User Service Registration extensions 39 template HttpRequest mw_registerAllServicesHttpRequestWithAuthoriza 40 Authorization := { 41 Value := "Digest", 42 ParameterList := { 43 {Name := "response", Value := ?}, 44 {Name := "username", Value := "" & p_user 45 *</pre>
<pre>46 } 47 } 48 } 49 50 template HttpRequest mw_deRegisterAllServicesHttpRequest modifies m 51 UriParameters := { 52 {Parameter := "requesttype", Value := "deregister"} 53 }, 54 ContentType := { 55 // see TS 26.346 56 Name := "Content-Type", ListOfValue := {{Va 57 }, 58 Authorization := omit, 59 Body := { 60 mbmsSecurityRegister := ? 61 } 62 } 63 64 // change 1 (WK23): Service Request extensions for content protecti 65 template HttpRequest mw_smartcartServiceRequestHttpRequest(UInt p_p 66 ContentType := { 67 // OMA Services 5.1.2.1 68 Name := "Content-Type", ListOfValue := {{Va 69 }, 70 Authorization := omit,</pre>	=	<pre>46 } 47 } 48 } 49 50 template HttpRequest mw_deRegisterAllServicesHttpRequest modifies m 51 UriParameters := { 52 {Parameter := "requesttype", Value := "deregister"} 53 }, 54 ContentType := { 55 // see TS 26.346 56 Name := "Content-Type", ListOfValue := {{Va 57 }, 58 Authorization := omit, 59 Body := { 60 mbmsSecurityRegister := ? 61 } 62 } 63 64 // change 1 (WK23): Service Request extensions for content protecti 65 template HttpRequest mw_smartcartServiceRequestHttpRequest(UInt p_p 66 ContentType := { 67 // OMA Services 5.1.2.1 68 Name := "Content-Type", ListOfValue := {{Va 69 }, 70 Authorization := omit,</pre>

Datei: LibCommon_BSM.ttcn (Fortsetzung)

72	Body := {		72	Body := {	
73	//OMA Services 5.1.5.2.1 Service Request		73	//OMA Services 5.1.5.2.1 Service Request	
74	ServiceRequest := mw_smartcartServiceRequest(p_prot		74	ServiceRequest := mw_smartcartServiceRequest(p_prot	
75	}		75	}	
76	}		76	}	
77			77		
78	// change 1 (WK23): Service Request extensions for content protecti		78	// change 1 (WK23): Service Request extensions for content protecti	
79	template HttpRequest mw_anyServiceRequestHttpRequest modifies mw_ar		79	template HttpRequest mw_anyServiceRequestHttpRequest modifies mw_ar	
80	ContentType := {		80	ContentType := {	
81	// OMA Services 5.1.2.1		81	// OMA Services 5.1.2.1	
82	Name := "Content-Type", ListOfValue := {{Va		82	Name := "Content-Type", ListOfValue := {{Va	
83	},		83	},	
84	Body := {		84	Body := {	
85	//OMA Services 5.1.5.2.1 Service Request		85	//OMA Services 5.1.5.2.1 Service Request	
86	ServiceRequest := mw_anyServiceRequest		86	ServiceRequest := mw_anyServiceRequest	
87	}		87	}	
88	}		88	}	
89			89		
90	// change 1 (WK23): Service Request extensions for content protecti		90	// change 1 (WK23): Service Request extensions for content protecti	
91	template HttpRequest mw_smartcartServiceRequestHttpRequestWithAuthc		91	template HttpRequest mw_smartcartServiceRequestHttpRequestWithAuthc	
92	Authorization := {		92	Authorization := {	
93	Value := "Digest",		93	Value := "Digest",	
94	ParameterList := {		94	ParameterList := {	
95	{Name := "response", Value := ?},		95	{Name := "response", Value := ?},	
96	{Name := "username", Value := p_username},	<>	96	{Name := "username", Value := "" & p_user	
97	*	=	97	*	
98	}		98	}	
99	}		99	}	
100	}		100	}	
101	}		101	}	
102			102		
103	group ResponseTemplates {		103	group ResponseTemplates {	
104	template HttpResponse m_digestBsmResponse(charstring p_nonce) modif		104	template HttpResponse m_digestBsmResponse(charstring p_nonce) modif	
105	anotherHeaders := {		105	anotherHeaders := {	
106	{Name := "Server", ListOfValue := {m_defaultBsmServ		106	{Name := "Server", ListOfValue := {m_defaultBsmServ	
107	{Name := "WWW-Authenticate", ListOfValue := {m_bsmW		107	{Name := "WWW-Authenticate", ListOfValue := {m_bsmW	
108	}		108	}	
109	}		109	}	
110			110		
111	template HttpResponse m_successBsmResponse(integer p_statusCode, ch		111	template HttpResponse m_successBsmResponse(integer p_statusCode, ch	
112	ContentType := {		112	ContentType := {	
113	// see TS 26.346		113	// see TS 26.346	
114	Name := "Content-Type", ListOfValue := {{Value := "		114	Name := "Content-Type", ListOfValue := {{Value := "	
115	},		115	},	
116	anotherHeaders := {		116	anotherHeaders := {	
117	{Name := "Server", ListOfValue := {m_defaultBsmServ		117	{Name := "Server", ListOfValue := {m_defaultBsmServ	
118	{Name := "Authentication-Info", ListOfValue := {m_d		118	{Name := "Authentication-Info", ListOfValue := {m_d	
119	},		119	},	
120	Body := {		120	Body := {	
121	mbmsSecurityRegisterResponse := valueof(p_mbmsSecur		121	mbmsSecurityRegisterResponse := valueof(p_mbmsSecur	
122	}		122	}	
123	}		123	}	
124	// change 1 (WK23): Service Request extensions for content protecti		124	// change 1 (WK23): Service Request extensions for content protecti	
125	template HttpResponse m_successBsmResponseCommon(charstring p_conte		125	template HttpResponse m_successBsmResponseCommon(charstring p_conte	
126	ContentType := {		126	ContentType := {	
127	// see TS 26.346		127	// see TS 26.346	
128	Name := "Content-Type", ListOfValue := {{Value := g		128	Name := "Content-Type", ListOfValue := {{Value := g	
129	},		129	},	
130	anotherHeaders := {		130	anotherHeaders := {	
131	{Name := "Server", ListOfValue := {m_defaultBsmServ		131	{Name := "Server", ListOfValue := {m_defaultBsmServ	
132	{Name := "Authentication-Info", ListOfValue := {m_d		132	{Name := "Authentication-Info", ListOfValue := {m_d	
133	},		133	},	
134	Body := p_httpBody		134	Body := p_httpBody	
135	}		135	}	
136	}		136	}	
137			137		
138	group HttpHeaderTemplates {		138	group HttpHeaderTemplates {	
139	template HttpHeaderValue m_defaultBsmServerHeader := {		139	template HttpHeaderValue m_defaultBsmServerHeader := {	
140	Value := "RS ATE BSM Server/0.01",		140	Value := "RS ATE BSM Server/0.01",	
141	ParameterList := omit		141	ParameterList := omit	
142	}		142	}	

Datei: LibCommon_BSM.ttcn (Fortsetzung)

143			143	
144	template HttpHeaderValue m_bsmWwwAuthenticate(charstring p_nonce) :		144	template HttpHeaderValue m_bsmWwwAuthenticate(charstring p_nonce) :
145	Value := "Digest",		145	Value := "Digest",
146	ParameterList := {		146	ParameterList := {
147	// see OMA BCAST SvcCntProtection 6.6		147	// see OMA BCAST SvcCntProtection 6.6
148	{Name := "realm", Value := ""3GPP-bootstrapping-ui<>		148	{Name := "realm", Value := "3GPP-bootstrapping-ui<
149	{Name := "nonce", Value := "" & p_nonce & ""}, =		149	{Name := "nonce", Value := "" & p_nonce & ""},
150	{Name := "opaque", Value := "" & "1234567890abcde		150	{Name := "opaque", Value := "" & "1234567890abcde
151	{Name := "algorithm", Value := "MD5"},		151	{Name := "algorithm", Value := "MD5"},
152	{Name := "qop", Value := "" & "auth-int" & ""},		152	{Name := "qop", Value := "" & "auth-int" & ""},
153	{Name := "stale", Value := "true"} // change 2 (WK2		153	{Name := "stale", Value := "true"} // change 2 (WK2
154	}		154	}
155	}		155	}
156	}		156	}
157	}		157	}
158			158	
159	group BSM_XML_Templates {		159	group BSM_XML_Templates {
160	group RegisterTemplates {		160	group RegisterTemplates {
161	template mbmsSecurityRegisterType mw_simpleMbmsSecurityRegister(char		161	template mbmsSecurityRegisterType mw_simpleMbmsSecurityRegister(char
162	serviceID := {p_serviceId},		162	serviceID := {p_serviceId},
163	registrationRequestExtension := *		163	registrationRequestExtension := *
164	}		164	}
165			165	
166	template mbmsSecurityRegisterResponseType m_simpleMbmsSecurityRegis		166	template mbmsSecurityRegisterResponseType m_simpleMbmsSecurityRegis
167	ResponseTypes := {		167	ResponseTypes := {
168	{		168	{
169	serviceID := {p_serviceId},		169	serviceID := {p_serviceId},
170	ResponseCode := {"200 OK"}		170	ResponseCode := {"200 OK"}
171	}		171	}
172	}		172	}
173	}		173	}
174			174	
175	template mbmsSecurityRegisterResponseType m_testMbmsSecurityRegiste		175	template mbmsSecurityRegisterResponseType m_testMbmsSecurityRegiste
176	ResponseTypes := {		176	ResponseTypes := {
177	{		177	{
178	serviceID := {p_serviceId},		178	serviceID := {p_serviceId},
179	ResponseCode := {"200 OK"}		179	ResponseCode := {"200 OK"}
180	}		180	}
181	},		181	},
182	RegistrationResponseExtension := {		182	RegistrationResponseExtension := {
183	version := 0, // BCAST 1.0		183	version := 0, // BCAST 1.0
184	LTKMDelivery := {Types := {{0}}} // 0 = UDP		184	LTKMDelivery := {Types := {{0}}} // 0 = UDP
185	}		185	}
186	}		186	}
187	}		187	}
188			188	
189	group DeRegisterTemplates {		189	group DeRegisterTemplates {
190	// change 3 (WK47/08): SC deregistration: Clerical Change	<>	190	
191	template mbmsSecurityDeregisterType mw_simpleMbmsSecurityDeRegister		190	template mbmsSecurityDeregisterType mw_simpleMbmsSecurityDeRegister
192	serviceID := {	=	191	serviceID := {
193	p_serviceId		192	p_serviceId
194	}		193	}
195	}		194	}
196	}		195	}
197			196	
198	// change 1 (WK23): Service Request extensions for content protection tests		197	// change 1 (WK23): Service Request extensions for content protection tests
199	group ServiceTemplates {		198	group ServiceTemplates {
200	template ServiceRequestType mw_smartcartServiceRequest(UInt p_prote		199	template ServiceRequestType mw_smartcartServiceRequest(UInt p_prote
201	SmartcardProfileSpecificPart := {		200	SmartcardProfileSpecificPart := {
202	ProtectionKeyIDs := {		201	ProtectionKeyIDs := {
203	{p_protectionKeyId},		202	{p_protectionKeyId},
204	*		203	*
205	}		204	}
206	}		205	}
207	}		206	}
208			207	
209	// change 1 (WK23): Service Request extensions for content protecti		208	// change 1 (WK23): Service Request extensions for content protecti
210	template ServiceRequestType mw_anyServiceRequest := {		209	template ServiceRequestType mw_anyServiceRequest := {
211	requestID := *,		210	requestID := *,
212	UserID := *,		211	UserID := *,
213	DeviceID := *,		212	DeviceID := *,

Datei: LibCommon_BSM.ttcn (Fortsetzung)

214	PurchaseItems := {*,		213	PurchaseItems := {*,	
215	DrmProfileSpecificPart := *,		214	DrmProfileSpecificPart := *,	
216	SmartcardProfileSpecificPart := *		215	SmartcardProfileSpecificPart := *	
217			216		
218	}		217	}	
219			218		
220	// change 1 (WK23): Service Request extensions for content protection tests		219	// change 1 (WK23): Service Request extensions for content protection tests	
221	template ServiceResponseType m_testServiceResponse(UInt p_requestId		220	template ServiceResponseType m_testServiceResponse(UInt p_requestId	
222	requestID := p_requestId		221	requestID := p_requestId	
223	}		222	}	
224			223		
225	// change 1 (WK23): Service Request extensions for content protection tests		224	// change 1 (WK23): Service Request extensions for content protection tests	
226	template ServiceResponseType m_testServiceResponseWithoutRequestID		225	template ServiceResponseType m_testServiceResponseWithoutRequestID	
227	requestID := omit,		226	requestID := omit,	
228	globalStatusCode := 0,		227	globalStatusCode := 0,	
229	adaptationMode := omit,		228	adaptationMode := omit,	
230	PurchaseItems := p_purchaseItemList		229	PurchaseItems := p_purchaseItemList	
231	}		230	}	
232	}		231	}	
233	}		232	}	
234	}		233	}	
235			234		
236	group BSM_Functions {		235	group BSM_Functions {	
237	// change 1 (WK46/08): Clerical naming changes	+-			
238	// change 1 (WK23): Service Request extensions for content protection tests	=	236	// change 1 (WK23): Service Request extensions for content protection tests	
239	function BSM_ServerSimulation(boolean p_ActivateUserServiceRegistration, boolean p_	<>	237	function BSM_ServerSimulation(boolean p_activateUserServiceRegistration, boolean p_	
240	boolean p_ActivateServiceRequest, boolean p_SrActAsParallelComponent {		238	boolean p_activateServiceRequest, boolean p_activateServiceRequest, boolean p_activateS	
241) runs on BSMServerComponent {		239) runs on BSMServerComponent {	
242	// variables for HTTP communication	=	240	// variables for HTTP communication	
243	var HttpResponse v_response;		241	var HttpResponse v_response;	
244	var HttpRequest v_request;		242	var HttpRequest v_request;	
245			243		
246	var octetstring v_rspAuth, v_mrk;		244	var octetstring v_rspAuth, v_mrk;	
247	var charstring v_nonce := fx_bitstring2Base64('00000000'B);		245	var charstring v_nonce := fx_bitstring2Base64('00000000'B);	
248			246		
249	// variables used for MBMS Security Register Requests/Responses		247	// variables used for MBMS Security Register Requests/Responses	
250	var mbmsSecurityRegisterResponseType v_mbmsXml;		248	var mbmsSecurityRegisterResponseType v_mbmsXml;	
251	// variables used for Service Requests/Responses		249	// variables used for Service Requests/Responses	
252	var ServiceResponseType v_serviceResponseXml;		250	var ServiceResponseType v_serviceResponseXml;	
253			251		
254	var UInt v_protectionKeyId;		252	var UInt v_protectionKeyId;	
255			253		
256	if (p_ActivateServiceRequest) {	<>	254	if (p_activateServiceRequest) {	
257	v_protectionKeyId := hex2int(protectionKeyId)	=	255	v_protectionKeyId := hex2int(protectionKeyId)	
258	}		256	}	
259			257		
260		<>			
261	// change 6 (WK46/08): MBMS Registration Key = ks_ext_NAF				
262	// if DUT supports MBMS key management and GBA_U				
263	v_mrk := ks_ext_NAF;				
264	// MRK = KDF(Ks_ext_NAF, "mbms-mrk")	=	258	// MRK = KDF(Ks_ext_NAF, "mbms-mrk")	
265	// v_mrk := fx_calculatingKDF4MRK(ks_ext_NAF, fx_char2octet("mbms-mrk"));	<>	259	v_mrk := fx_calculatingKDF4MRK(ksNAF, fx_char2octet("mbms-mrk"));	
266		=	260		
267	log("BSM ready...");		261	log("BSM ready...");	
268			262		
269	if ((not p_UsrActAsParallelComponent) or (not p_SrActAsParallelComponent))		263	if ((not p_UsrActAsParallelComponent) or (not p_SrActAsParallelComponent))	
270	t_wait.start(PX_TWAIT);		264	t_wait.start(PX_TWAIT);	
271	}		265	}	
272	alt {		266	alt {	
273	// change 1 (WK46/08): Clerical naming changes	<>			
274	[p_ActivateUserServiceRegistration] bsm.receive(mw_registerAllServicesHttpRequ		267	[p_activateUserServiceRegistration] bsm.receive(mw_registerAllServicesHttpRequ	
275	log("Verdict Info: 2a. The terminal sends the Registration		268	log("Verdict Info: 2a. The terminal sends the Registration	
276	setverdict(pass);	=	269	setverdict(pass);	
277	bsm.send(m_digestBsmResponse(v_nonce));		270	bsm.send(m_digestBsmResponse(v_nonce));	
278	alt {		271	alt {	
279	[] bsm.receive(mw_registerAllServicesHttpRequestWithProtectionKey		272	[] bsm.receive(mw_registerAllServicesHttpRequestWithProtectionKey	
280	// change 4 (WK46/08): SC Registration, modification	<>			
281	if (match(str2oct(getHeaderParameterValue("Service-Registration-Key", "mbms-mrk"))		273	if (match(str2oct(unq(getHeaderParameterValue("Service-Registration-Key", "mbms-mrk"))	
282	log("Verdict Info: 2c. The second registration attempt failed");	=	274	log("Verdict Info: 2c. The second registration attempt failed");	
283	setverdict(pass);		275	setverdict(pass);	
284	v_mbmsXml := valueof(m_testMbmsSecurityRegisterResponse);		276	v_mbmsXml := valueof(m_testMbmsSecurityRegisterResponse);	

Datei: LibCommon_BSM.ttcn (Fortsetzung)

285	v_rspAuth := ts_calculateRspauth(v	277	v_rspAuth := ts_calculateRspauth(v
286	v_response := m_successBsmResponseC	<> 278	v_response := m_successBsmResponseC
287	bsm.send(v_response);	= 279	bsm.send(v_response);
288	}	280	}
289	else {	281	else {
290	log("Verdict Info: RES doesn't corr	282	log("Verdict Info: RES doesn't corr
291	setverdict(fail);	283	setverdict(fail);
292	bsm.send(m_basicResponse(400));	284	bsm.send(m_basicResponse(400));
293	}	285	}
294	}	286	}
295	[] bsm.receive(mw_anyHttpRequest) {	287	[] bsm.receive(mw_anyHttpRequest) {
296	log("Verdict Info: The terminal doesn't ser	288	log("Verdict Info: The terminal doesn't ser
297	setverdict(fail);	289	setverdict(fail);
298	bsm.send(m_basicResponse(400));	290	bsm.send(m_basicResponse(400));
299	}	291	}
300	[not p_UsrActAsParallelComponent] t_wait.timeout {	292	[not p_UsrActAsParallelComponent] t_wait.timeout {
301	log("Verdict Info: The terminal doesn't ser	293	log("Verdict Info: The terminal doesn't ser
302	setverdict(inconc);	294	setverdict(inconc);
303	}	295	}
304	} // inner alt	296	} // inner alt
305		297	
306	if (p_UsrActAsParallelComponent) {	298	if (p_UsrActAsParallelComponent) {
307	repeat;	299	repeat;
308	}	300	}
309	}	301	}
310	// change 1 (WK46/08): Clerical naming changes	<>	
311	[p_ActivateServiceRequest] bsm.receive(mw_anyServiceRequestHttpRequest) ->	302	[p_activateServiceRequest] bsm.receive(mw_anyServiceRequestHttpRequest) ->
312	log("Verdict Info: 2a. The terminal sends the Service Request for t	= 303	log("Verdict Info: 2a. The terminal sends the Service Request for t
313	setverdict(pass);	304	setverdict(pass);
314	bsm.send(m_digestBsmResponse(v_nonce));	305	bsm.send(m_digestBsmResponse(v_nonce));
315	alt {	306	alt {
316	[] bsm.receive(mw_anyServiceRequestHttpRequest) ->	307	[] bsm.receive(mw_anyServiceRequestHttpRequest) ->
317	// change 4 (WK46/08): SC Registration, mod	<>	
318	if (match(str2oct(getHeaderParameterValue("	308	if (match(str2oct(unq(getHeaderParameterValue("
319	log("Verdict Info: 2c. The second E	= 309	log("Verdict Info: 2c. The second E
320	setverdict(pass);	310	setverdict(pass);
321	var PurchaseItemList requestPurchase	311	var PurchaseItemList requestPurchase
322	var PurchaseItemResponseList respon	312	var PurchaseItemResponseList respon
323	for (var integer i := 0; i < sizeof	313	for (var integer i := 0; i < sizeof
324	responsePurchaseItems[i] :=	314	responsePurchaseItems[i] :=
325	}	315	}
326		316	
327	if (ispresent(v_request.Body.Servic	317	if (ispresent(v_request.Body.Servic
328	v_serviceResponseXml := val	318	v_serviceResponseXml := val
329	}	319	}
330	else {	320	else {
331	v_serviceResponseXml := val	321	v_serviceResponseXml := val
332	}	322	}
333		323	
334	v_rspAuth := ts_calculateRspauth(v	324	v_rspAuth := ts_calculateRspauth(v
335	v_response := m_successBsmResponseC	325	v_response := m_successBsmResponseC
336	bsm.send(v_response);	326	bsm.send(v_response);
337	}	327	}
338	else {	328	else {
339	log("Verdict Info: RES doesn't corr	329	log("Verdict Info: RES doesn't corr
340	setverdict(fail);	330	setverdict(fail);
341	bsm.send(m_basicResponse(400));	331	bsm.send(m_basicResponse(400));
342	}	332	}
343	}	333	}
344	[] bsm.receive(mw_anyHttpRequest) {	334	[] bsm.receive(mw_anyHttpRequest) {
345	log("Verdict Info: The terminal doesn't ser	335	log("Verdict Info: The terminal doesn't ser
346	setverdict(fail);	336	setverdict(fail);
347	bsm.send(m_basicResponse(400));	337	bsm.send(m_basicResponse(400));
348	}	338	}
349	[not p_SrActAsParallelComponent] t_wait.timeout {	339	[not p_SrActAsParallelComponent] t_wait.timeout {
350	log("Verdict Info: The terminal doesn't ser	340	log("Verdict Info: The terminal doesn't ser
351	setverdict(inconc);	341	setverdict(inconc);
352	}	342	}
353	} // inner alt	343	} // inner alt
354		344	
355	if (p_SrActAsParallelComponent) {	345	if (p_SrActAsParallelComponent) {

Datei: LibCommon_BSM.ttcn (Fortsetzung)

356	repeat;		346	repeat;
357	}		347	}
358	}		348	}
359	[] bsm.receive(mw_anyHttpRequest) {		349	[] bsm.receive(mw_anyHttpRequest) {
360	log("Verdict Info: The terminal doesn't send a HTTP request		350	log("Verdict Info: The terminal doesn't send a HTTP request
361	setverdict(fail);		351	setverdict(fail);
362	bsm.send(m_basicResponse(400));		352	bsm.send(m_basicResponse(400));
363			353	
364	if (p_UsrActAsParallelComponent or p_SrActAsParallelComponen		354	if (p_UsrActAsParallelComponent or p_SrActAsParallelComponen
365	repeat;		355	repeat;
366	}		356	}
367	}		357	}
368	[(not p_UsrActAsParallelComponent) or (not p_SrActAsParallelComponen		358	[(not p_UsrActAsParallelComponent) or (not p_SrActAsParallelComponen
369	log("Verdict Info: The terminal doesn't send a HTTP request		359	log("Verdict Info: The terminal doesn't send a HTTP request
370	setverdict(fail);		360	setverdict(fail);
371	}		361	}
372	} // alt		362	} // alt
373			363	
374	return;		364	return;
375	}		365	}
376			366	
377	// change 4 (WK46/08): SC Registration, modified HTTP templates and functions	+-		
378	// change 2 (WK23): GBA authentication extensions for content protection test	=	367	// change 2 (WK23): GBA authentication extensions for content protection test
379	function ts_calculateAuthResponse(in HttpRequest p_httpRequest, in octetstring p_pa		368	function ts_calculateAuthResponse(in HttpRequest p_httpRequest, in octetstring p_pa
380	var ListOfParameter paramList := p_httpRequest.Authorization.ParameterList;		369	var ListOfParameter paramList := p_httpRequest.Authorization.ParameterList;
381	var charstring v_Username := getHeaderParameterValue("username", paramList);		370	var charstring v_Username := getHeaderParameterValue("username", paramList);
382	var octetstring v_oUsername := fx_char2octet(v_Username);	<>	371	var octetstring v_oUsername := fx_char2octet(unq(v_Username));
383	var charstring v_Realm := getHeaderParameterValue("realm", paramList);	=	372	var charstring v_Realm := getHeaderParameterValue("realm", paramList);
384	var octetstring v_oRealm := fx_char2octet(v_Realm);	<>	373	var octetstring v_oRealm := fx_char2octet(unq(v_Realm));
385	var charstring v_Method := p_httpRequest.Method;	=	374	var charstring v_Method := p_httpRequest.Method;
386	var octetstring v_oMETHOD := fx_char2octet(v_Method);		375	var octetstring v_oMETHOD := fx_char2octet(v_Method);
387	var charstring v_DigestUri := getHeaderParameterValue("uri", paramList);		376	var charstring v_DigestUri := getHeaderParameterValue("uri", paramList);
388	var octetstring v_oDigestUri := fx_char2octet(v_DigestUri);		377	var octetstring v_oDigestUri := fx_char2octet(v_DigestUri);
389	var charstring v_Nonce := getHeaderParameterValue("nonce", paramList);		378	var charstring v_Nonce := getHeaderParameterValue("nonce", paramList);
390	var octetstring v_oNonce := fx_char2octet(v_Nonce);	<>	379	var octetstring v_oNonce := fx_char2octet(unq(v_Nonce));
391	var charstring v_NC := getHeaderParameterValue("nc", paramList);	=	380	var charstring v_NC := getHeaderParameterValue("nc", paramList);
392	var octetstring v_oNC := fx_char2octet(v_NC);		381	var octetstring v_oNC := fx_char2octet(v_NC);
393	var charstring v_CNonce := getHeaderParameterValue("cnonce", paramList);		382	var charstring v_CNonce := getHeaderParameterValue("cnonce", paramList);
394	var octetstring v_oCNonce := fx_char2octet(v_CNonce);	<>	383	var octetstring v_oCNonce := fx_char2octet(unq(v_CNonce));
395	var charstring v_Qop := getHeaderParameterValue("qop", paramList);	=	384	var charstring v_Qop := getHeaderParameterValue("qop", paramList);
396	var octetstring v_oQop := fx_char2octet(v_Qop);		385	var octetstring v_oQop := fx_char2octet(v_Qop);
397			386	
398	// RFC 2617 3.2.2.2		387	// RFC 2617 3.2.2.2
399	// A1 = unq(username-value) ":" unq(realm-value) ":" passwd		388	// A1 = unq(username-value) ":" unq(realm-value) ":" passwd
400	var octetstring A1 := v_oUsername & COLON_HEX & v_oRealm & COLON_HEX & p_pa		389	var octetstring A1 := v_oUsername & COLON_HEX & v_oRealm & COLON_HEX & p_pa
401	// H(A1)		390	// H(A1)
402	var octetstring HA1 := fx_md5_hex(A1);		391	var octetstring HA1 := fx_md5_hex(A1);
403	var charstring cHA1 := oct2str(HA1);		392	var charstring cHA1 := oct2str(HA1);
404	var octetstring v_RequestDigest, A2, secret_data, v_Secret, v_HA2, v_Data;		393	var octetstring v_RequestDigest, A2, secret_data, v_Secret, v_HA2, v_Data;
405			394	
406	v_Secret := fx_charLow2octet(cHA1);		395	v_Secret := fx_charLow2octet(cHA1);
407			396	
408	// RFC 2617 3.2.2.3		397	// RFC 2617 3.2.2.3
409	// A2 = Method ":" digest-uri-value		398	// A2 = Method ":" digest-uri-value
410	A2 := v_oMETHOD & COLON_HEX & v_oDigestUri;		399	A2 := v_oMETHOD & COLON_HEX & v_oDigestUri;
411			400	
412	v_HA2 := fx_charLow2octet(oct2str(fx_md5_hex(A2)));		401	v_HA2 := fx_charLow2octet(oct2str(fx_md5_hex(A2)));
413			402	
414	// RFC 2617 3.2.2.1		403	// RFC 2617 3.2.2.1
415	// unq(nonce-value) ":" nc-value ":" unq(cnonce-value) ":" unq(qop-value) "		404	// unq(nonce-value) ":" nc-value ":" unq(cnonce-value) ":" unq(qop-value) "
416	v_Data := (v_oNonce & COLON_HEX & v_oNC & COLON_HEX & v_oCNonce & COLON_HEX		405	v_Data := (v_oNonce & COLON_HEX & v_oNC & COLON_HEX & v_oCNonce & COLON_HEX
417			406	
418	// now apply KD function described in RFC 2617 section 3.2.1		407	// now apply KD function described in RFC 2617 section 3.2.1
419	// KD(secret, data) = H(concat(secret, ":", data))		408	// KD(secret, data) = H(concat(secret, ":", data))
420	secret_data := v_Secret & COLON_HEX & v_Data;		409	secret_data := v_Secret & COLON_HEX & v_Data;
421			410	
422	v_RequestDigest := fx_md5_hex(secret_data);		411	v_RequestDigest := fx_md5_hex(secret_data);
423			412	
424	log("ts_calculateAuthResponse(): METHOD '" & v_Method & "'");		413	log("ts_calculateAuthResponse(): METHOD '" & v_Method & "'");
425	log("ts_calculateAuthResponse(): client username '" & v_Username & "'");		414	log("ts_calculateAuthResponse(): client username '" & v_Username & "'");
426	log("ts_calculateAuthResponse(): client realm '" & v_Realm & "'");		415	log("ts_calculateAuthResponse(): client realm '" & v_Realm & "'");

Datei: LibCommon_BSM.ttcn (Fortsetzung)

427	log("ts_calculateAuthResponse(): client URI '" & v_DigestUri & "'");		416	log("ts_calculateAuthResponse(): client URI '" & v_DigestUri & "'");
428	log("ts_calculateAuthResponse(): client nonce '" & v_Nonce & "'");		417	log("ts_calculateAuthResponse(): client nonce '" & v_Nonce & "'");
429	log("ts_calculateAuthResponse(): client nc '" & v_NC & "'");		418	log("ts_calculateAuthResponse(): client nc '" & v_NC & "'");
430	log("ts_calculateAuthResponse(): client cnonce '" & v_CNonce & "'");		419	log("ts_calculateAuthResponse(): client cnonce '" & v_CNonce & "'");
431	log("ts_calculateAuthResponse(): client QoP '" & v_Qop & "'");		420	log("ts_calculateAuthResponse(): client QoP '" & v_Qop & "'");
432			421	
433	return v_RequestDigest;		422	return v_RequestDigest;
434	}		423	}
435			424	
436			425	
437	// change 4 (WK46/08): SC Registration, modified HTTP templates and functions	+-		
438	// change 2 (WK23): GBA authentication extensions for content protection test	=	426	// change 2 (WK23): GBA authentication extensions for content protection test
439	function ts_calculateRspauth(in HttpRequest p_request, in octetstring p_password, in		427	function ts_calculateRspauth(in HttpRequest p_request, in octetstring p_password, in
440	// RFC 2617 and 3310		428	// RFC 2617 and 3310
441	var ListOfParameter p_list := p_request.Authorization.ParameterList;		429	var ListOfParameter p_list := p_request.Authorization.ParameterList;
442	var charstring v_NC := getHeaderParameterValue("nc", p_list);		430	var charstring v_NC := getHeaderParameterValue("nc", p_list);
443	var charstring v_CNonce := getHeaderParameterValue("cnonce", p_list);		431	var charstring v_CNonce := getHeaderParameterValue("cnonce", p_list);
444	var charstring v_DigestUri := getHeaderParameterValue("uri", p_list);		432	var charstring v_DigestUri := getHeaderParameterValue("uri", p_list);
445			433	
446	// username is BTID		434	// username is BTID
447	var octetstring v_oUsername := fx_char2octet(btId);		435	var octetstring v_oUsername := fx_char2octet(btId);
448	// complete URI (e.g. "/bmsc.home1.net/keymanagement?requesttype=register")		436	// complete URI (e.g. "/bmsc.home1.net/keymanagement?requesttype=register")
449	var octetstring v_oDigestUri := fx_char2octet(v_DigestUri);		437	var octetstring v_oDigestUri := fx_char2octet(v_DigestUri);
450	var octetstring v_oNonce := fx_char2octet(p_nonce);		438	var octetstring v_oNonce := fx_char2octet(p_nonce);
451	var octetstring v_oNC := fx_char2octet(v_NC);		439	var octetstring v_oNC := fx_char2octet(v_NC);
452	var octetstring v_oCNonce := fx_char2octet(v_CNonce);	<>	440	var octetstring v_oCNonce := fx_char2octet(unq(v_CNonce));
453	var octetstring v_oQop := fx_char2octet("auth-int");	=	441	var octetstring v_oQop := fx_char2octet("auth-int");
454			442	
455	var octetstring A1, v_oRealm, v_Secret, v_HA2, v_Data, v_RequestDigest, h_e		443	var octetstring A1, v_oRealm, v_Secret, v_HA2, v_Data, v_RequestDigest, h_e
456			444	
457	// see OMA BCAST SvcCntProtection 6.6		445	// see OMA BCAST SvcCntProtection 6.6
458	if (PX_GBA == e_gba_me) {		446	if (PX_GBA == e_gba_me) {
459	v_oRealm := fx_char2octet("3GPP-bootstrapping@" & PX_BSM_FQDN);		447	v_oRealm := fx_char2octet("3GPP-bootstrapping@" & PX_BSM_FQDN);
460	}		448	}
461	else {		449	else {
462	v_oRealm := fx_char2octet("3GPP-bootstrapping-uicc@" & PX_BSM_FQDN)		450	v_oRealm := fx_char2octet("3GPP-bootstrapping-uicc@" & PX_BSM_FQDN)
463	}		451	}
464			452	
465	// RFC 2617 3.2.2.2		453	// RFC 2617 3.2.2.2
466	// A1 = unq(username-value) ":" unq(realm-value) ":" passwd		454	// A1 = unq(username-value) ":" unq(realm-value) ":" passwd
467	A1 := v_oUsername & COLON_HEX & v_oRealm & COLON_HEX & p_password;		455	A1 := v_oUsername & COLON_HEX & v_oRealm & COLON_HEX & p_password;
468			456	
469	// H(A1)		457	// H(A1)
470	v_Secret := fx_charLow2octet(oct2str(fx_md5_hex(A1)));		458	v_Secret := fx_charLow2octet(oct2str(fx_md5_hex(A1)));
471			459	
472	// RFC 2617 3.2.3		460	// RFC 2617 3.2.3
473	// A2 = ":" digest-uri-value ":" H(entity-body)		461	// A2 = ":" digest-uri-value ":" H(entity-body)
474	h_entityBody := fx_charLow2octet(oct2str(fx_md5_hex(p_entityBody)));		462	h_entityBody := fx_charLow2octet(oct2str(fx_md5_hex(p_entityBody)));
475	A2 := COLON_HEX & v_oDigestUri & COLON_HEX & h_entityBody;		463	A2 := COLON_HEX & v_oDigestUri & COLON_HEX & h_entityBody;
476			464	
477	v_HA2 := fx_charLow2octet(oct2str(fx_md5_hex(A2)));		465	v_HA2 := fx_charLow2octet(oct2str(fx_md5_hex(A2)));
478			466	
479	// RFC 2617 3.2.2.1		467	// RFC 2617 3.2.2.1
480	// unq(nonce-value) ":" nc-value ":" unq(cnonce-value) ":" unq(qop-value)		468	// unq(nonce-value) ":" nc-value ":" unq(cnonce-value) ":" unq(qop-value)
481	v_Data := (v_oNonce & COLON_HEX & v_oNC & COLON_HEX & v_oCNonce & COLON_HEX		469	v_Data := (v_oNonce & COLON_HEX & v_oNC & COLON_HEX & v_oCNonce & COLON_HEX
482			470	
483	//now apply KD function described in RFC 2617 section 3.2.1		471	//now apply KD function described in RFC 2617 section 3.2.1
484	//KD(secret, data) = H(concat(secret, ":", data))		472	//KD(secret, data) = H(concat(secret, ":", data))
485	secret_data := v_Secret & COLON_HEX & v_Data;		473	secret_data := v_Secret & COLON_HEX & v_Data;
486			474	
487	log("ts_calculateRspauth(): v_NC -> " & v_NC);		475	log("ts_calculateRspauth(): v_NC -> " & v_NC);
488	log("ts_calculateRspauth(): v_CNonce -> " & v_CNonce);		476	log("ts_calculateRspauth(): v_CNonce -> " & v_CNonce);
489	log("ts_calculateRspauth(): v_oUsername -> " & oct2str(v_oUsername));		477	log("ts_calculateRspauth(): v_oUsername -> " & oct2str(v_oUsername));
490	log("ts_calculateRspauth(): v_oRealm -> " & oct2str(v_oRealm));		478	log("ts_calculateRspauth(): v_oRealm -> " & oct2str(v_oRealm));
491	log("ts_calculateRspauth(): v_DigestUri -> " & v_DigestUri);		479	log("ts_calculateRspauth(): v_DigestUri -> " & v_DigestUri);
492	log("ts_calculateRspauth(): v_oNonce/p_nonce -> " & p_nonce);		480	log("ts_calculateRspauth(): v_oNonce/p_nonce -> " & p_nonce);
493	log("ts_calculateRspauth(): v_oNC/v_NC -> " & v_NC);		481	log("ts_calculateRspauth(): v_oNC/v_NC -> " & v_NC);
494	log("ts_calculateRspauth(): v_oCNonce -> " & v_CNonce);		482	log("ts_calculateRspauth(): v_oCNonce -> " & v_CNonce);
495	log("ts_calculateRspauth(): v_oQop -> " & oct2str(v_oQop));		483	log("ts_calculateRspauth(): v_oQop -> " & oct2str(v_oQop));
496			484	
497	return fx_md5_hex(secret_data);		485	return fx_md5_hex(secret_data);

Datei: LibCommon_BSM.ttcn (Fortsetzung)

498	}		486	}
499			487	
500	// change 2 (WK23): GBA authentication/authorization extensions for content protect		488	// change 2 (WK23): GBA authentication/authorization extensions for content protect
501	function f_bsm_setProtectionKeyId(hexstring p_protectionKeyId) runs on BSMServerCon		489	function f_bsm_setProtectionKeyId(hexstring p_protectionKeyId) runs on BSMServerCon
502	protectionKeyId := p_protectionKeyId;		490	protectionKeyId := p_protectionKeyId;
503	}		491	}
504			492	
505	// change 2 (WK23): GBA authentication/authorization extensions for content protect		493	// change 2 (WK23): GBA authentication/authorization extensions for content protect
506	function f_bsm_setBtid(charstring p_btid) runs on BSMServerComponent {		494	function f_bsm_setBtid(charstring p_btid) runs on BSMServerComponent {
507	btid := p_btid;		495	btid := p_btid;
508	}		496	}
509			497	
510	// change 2 (WK23): GBA authentication/authorization extensions for content protect		498	// change 2 (WK23): GBA authentication/authorization extensions for content protect
511	function f_bsm_setProtectionKeysAndValues(octetstring p_ksExtNAF) runs on BSMServer	<>	499	function f_bsm_setProtectionKeysAndValues(octetstring p_ksNAF) runs on BSMServerCon
512	ks_ext_NAF := p_ksExtNAF; // change 1 (WK46/08): Clerical naming changes		500	ksNAF := p_ksNAF;
513	}	=	501	}
514	}		502	}
515			503	
516	group BSM_Configuration_and_Components {		504	group BSM_Configuration_and_Components {
517	type component BSMServerComponent {		505	type component BSMServerComponent {
518	port BSMPort bsm;		506	port BSMPort bsm;
519			507	
520	timer t_wait;		508	timer t_wait;
521			509	
522	var hexstring protectionKeyId; // change 2 (WK23): GBA authentication/autho		510	var hexstring protectionKeyId; // change 2 (WK23): GBA authentication/autho
523	var charstring btid; // change 2 (WK23): GBA authentication/authorization e		511	var charstring btid; // change 2 (WK23): GBA authentication/authorization e
524	// change 1 (WK46/08): Clerical naming changes	<>		
525	var octetstring ks_ext_NAF // change 2 (WK23): GBA authentication/authoriza		512	var octetstring ksNAF // change 2 (WK23): GBA authentication/authorization
526	}	=	513	}
527			514	
528	type port BSMPort message {		515	type port BSMPort message {
529	in HttpRequest;		516	in HttpRequest;
530	out HttpResponse		517	out HttpResponse
531	}		518	}
532			519	
533	}		520	}
534			521	
535	group externalFunctions {		522	group externalFunctions {
536	external function fx_mbmsXml2Oct(in mbmsSecurityRegisterResponseType p_mbmsXml) ret		523	external function fx_mbmsXml2Oct(in mbmsSecurityRegisterResponseType p_mbmsXml) ret
537	// change 1 (WK23): Service Request extensions for content protection tests		524	// change 1 (WK23): Service Request extensions for content protection tests
538	external function fx_serviceXml2Oct(in ServiceResponseType p_serviceXml) return oct		525	external function fx_serviceXml2Oct(in ServiceResponseType p_serviceXml) return oct
539	// change 1 (WK23): Service Request extensions for content protection tests		526	// change 1 (WK23): Service Request extensions for content protection tests
540	external function fx_calculatingKDF4MRK(in octetstring p_keys, in octetstring p_kdf		527	external function fx_calculatingKDF4MRK(in octetstring p_keys, in octetstring p_kdf
541	}		528	}
542	}		529	}
543	}		530	}

Datei: LibCommon_GBA_BSF.ttcn

1	//change 1 (WK18): for content protection tests	=	1	//change 1 (WK18): for content protection tests
2	module LibCommon_GBA_BSF {		2	module LibCommon_GBA_BSF {
3	import from LibCommon_HTTP_TypesAndValues all;		3	import from LibCommon_HTTP_TypesAndValues all;
4	import from LibCommon_DataStrings all;		4	import from LibCommon_DataStrings all;
5	import from AtsBCast_ModuleParameters {		5	import from AtsBCast_ModuleParameters {
6	group GBA;		6	group GBA;
7	group BSM; // change 2 (WK23): GBA authentication/authorization extensions for content prot		7	group BSM; // change 2 (WK23): GBA authentication/authorization extensions for content prot
8	}		8	}
9	import from LibCommon_GBA_BSF_TypesAndValues all;		9	import from LibCommon_GBA_BSF_TypesAndValues all;
10	import from LibCommon_Time all;		10	import from LibCommon_Time all;
11			11	
12	group BSF {		12	group BSF {
13	group BSF_Templates {		13	group BSF_Templates {
14	group HTTPTemplates {		14	group HTTPTemplates {
15	group RequestTemplates {		15	group RequestTemplates {
16	template HttpRequest mw_anyHttpRequest := {		16	template HttpRequest mw_anyHttpRequest := {
17	Method := ?,		17	Method := ?,
18	Uri := ?,		18	Uri := ?,
19	Version := ?,		19	Version := ?,
20	UriParameters := *,		20	UriParameters := *,
21	Host := ?,		21	Host := ?,
22	ContentType := *,		22	ContentType := *,
23	Authorization := *,		23	Authorization := *,
24	anotherHeaders := *,		24	anotherHeaders := *,
25	Body := *		25	Body := *
26	}		26	}
27			27	
28	template HttpRequest mw_initialHttpRequest modifies mw_anyHttpReque		28	template HttpRequest mw_initialHttpRequest modifies mw_anyHttpReque
29	Authorization := {		29	Authorization := {
30	Value := "Digest",		30	Value := "Digest",
31	ParameterList := {		31	ParameterList := {
32	{Name := "response", Value := ?},		32	{Name := "response", Value := ?},
33	{Name := "username", Value := PX_IMPI}, // <>		33	{Name := "username", Value := "" & PX_IMPI
34	*	=	34	*
35	}		35	}
36	}		36	}
37	}		37	}
38	}		38	}
39			39	
40	group ResponseTemplates {		40	group ResponseTemplates {
41	template HttpResponse m_basicResponse(integer p_statusCode) := {		41	template HttpResponse m_basicResponse(integer p_statusCode) := {
42	Version := "HTTP/1.1",		42	Version := "HTTP/1.1",
43	Code := p_statusCode,		43	Code := p_statusCode,
44	ContentType := omit,		44	ContentType := omit,
45	anotherHeaders := {		45	anotherHeaders := {
46	{Name := "Server", ListOfValue := {m_defaultBsfServ		46	{Name := "Server", ListOfValue := {m_defaultBsfServ
47	}		47	}
48	,		48	,
49	Body := omit		49	Body := omit
50	}		50	}
51			51	
52	template HttpResponse m_digestBsfResponse(charstring p_nonce) modif		52	template HttpResponse m_digestBsfResponse(charstring p_nonce) modif
53	anotherHeaders := {		53	anotherHeaders := {
54	{Name := "Server", ListOfValue := {m_defaultBsfServ		54	{Name := "Server", ListOfValue := {m_defaultBsfServ
55	{Name := "WWW-Authenticate", ListOfValue := {m_defa		55	{Name := "WWW-Authenticate", ListOfValue := {m_defa
56	}		56	}
57	}		57	}
58			58	
59	template HttpResponse m_successBsfResponse(charstring p_response, c		59	template HttpResponse m_successBsfResponse(charstring p_response, c
60	ContentType := {		60	ContentType := {
61	Name := "Content-Type", ListOfValue := {{Value := "		61	Name := "Content-Type", ListOfValue := {{Value := "
62	},		62	},
63	anotherHeaders := {		63	anotherHeaders := {
64	{Name := "Server", ListOfValue := {m_defaultBsfServ		64	{Name := "Server", ListOfValue := {m_defaultBsfServ
65	{Name := "Expires", ListOfValue := {{Value := g		65	{Name := "Expires", ListOfValue := {{Value := g
66	{Name := "Authentication-Info", ListOfValue := {m_d		66	{Name := "Authentication-Info", ListOfValue := {m_d
67	},		67	},
68	Body := {		68	Body := {
69	BootstrappingInfo := p_bsfxml		69	BootstrappingInfo := p_bsfxml
70	}		70	}
71	}		71	}

Datei: LibCommon_GBA_BSF.ttcn (Fortsetzung)

72	}	72	}
73		73	
74	group HttpHeaderTemplates {	74	group HttpHeaderTemplates {
75	template HttpHeaderValue m_defaultBsfServerHeader := {	75	template HttpHeaderValue m_defaultBsfServerHeader := {
76	Value := "RS ATE BSF Server/0.02",	76	Value := "RS ATE BSF Server/0.02",
77	ParameterList := omit	77	ParameterList := omit
78	}	78	}
79		79	
80	template HttpHeaderValue m_defaultWwwAuthenticate(charstring p_nonce	80	template HttpHeaderValue m_defaultWwwAuthenticate(charstring p_nonce
81	Value := "Digest",	81	Value := "Digest",
82	ParameterList := {	82	ParameterList := {
83	{Name := "realm", Value := "" & PX_BSF_FQDN & ""	83	{Name := "realm", Value := "" & PX_BSF_FQDN & ""
84	{Name := "nonce", Value := "" & p_nonce & ""},	84	{Name := "nonce", Value := "" & p_nonce & ""},
85	{Name := "algorithm", Value := "AKAv1-MD5"},	85	{Name := "algorithm", Value := "AKAv1-MD5"},
86	{Name := "qop", Value := "" & "auth-int" & ""}	86	{Name := "qop", Value := "" & "auth-int" & ""}
87	}	87	}
88	}	88	}
89		89	
90	template HttpHeaderValue m_defaultAuthInfoHeaderRspauth(charstring	90	template HttpHeaderValue m_defaultAuthInfoHeaderRspauth(charstring
91	Value := "rspauth",	91	Value := "rspauth",
92	ParameterList := {{ Name := p_response, Value := on	92	ParameterList := {{ Name := p_response, Value := on
93	}	93	}
94	}	94	}
95		95	
96	template HttpHeaderValue m_defaultAuthInfoHeaderQop := {	96	template HttpHeaderValue m_defaultAuthInfoHeaderQop := {
97	Value := "qop",	97	Value := "qop",
98	ParameterList := {{ Name := "auth-int", Value := on	98	ParameterList := {{ Name := "auth-int", Value := on
99	}	99	}
100	}	100	}
101	}	101	}
102	}	102	}
103		103	
104	group BSFXMLTemplate {	104	group BSFXMLTemplate {
105	template BootstrappingInfoType m_bsfXml(charstring p_btid, charstring p_li	105	template BootstrappingInfoType m_bsfXml(charstring p_btid, charstring p_li
106	btid := { text_ := p_btid },	106	btid := { text_ := p_btid },
107	lifeTime := { text_ := p_lifeTime }	107	lifeTime := { text_ := p_lifeTime }
108	}	108	}
109	}	109	}
110	}	110	}
111		111	
112	group BSF_Functions {	112	group BSF_Functions {
113	// change 2 (WK23): GBA authentication/authorization extensions for content protect	113	// change 2 (WK23): GBA authentication/authorization extensions for content protect
114	/**	114	/**
115	*	115	*
116	* @desc Initialisation of BSF depending values.	116	* @desc Initialisation of BSF depending values.
117	* @param p_ks_ext_NAF Returns Ks_ext_NAF value. Used from BSM server.	117	* @param p_ks_ext_NAF Returns Ks_ext_NAF value. Used from BSM server.
118	*/	118	*/
119	function f_bsf_init(out octetstring p_ks_ext_NAF, out octetstring p_ks_int_NAF) run	119	function f_bsf_init(out octetstring p_ks_ext_NAF, out octetstring p_ks_int_NAF) run
120	// BSF generates authentication vector and calculate expected response from	120	// BSF generates authentication vector and calculate expected response from
121	var bitstring v_IK := '0'B, v_CK := '0'B;	121	var bitstring v_IK := '0'B, v_CK := '0'B;
122	var Bit128 v_AUTN;	122	var Bit128 v_AUTN;
123		123	
124	// BSF generate key material -> TS 33.220 4.5.2	124	// BSF generate key material -> TS 33.220 4.5.2
125	var octetstring v_UaSecProtocolId := '0100000001'O;	125	var octetstring v_UaSecProtocolId := '0100000001'O;
126	var octetstring v_NAF_Id := fx_char2octet(PX_BSM_FQDN) & v_UaSecProtocolId;	126	var octetstring v_NAF_Id := fx_char2octet(PX_BSM_FQDN) & v_UaSecProtocolId;
127		127	
128	if (PX_GBA == e_gba_me) {	128	if (PX_GBA == e_gba_me) {
129	v_AUTN := ts_AuthenticationInit(v_IK, v_CK, xres);	129	v_AUTN := ts_AuthenticationInit(v_IK, v_CK, xres);
130	}	130	}
131	else {	131	else {
132	v_AUTN := ts_AuthenticationInitForGbaU(v_IK, v_CK, xres);	132	v_AUTN := ts_AuthenticationInitForGbaU(v_IK, v_CK, xres);
133	}	133	}
134		134	
135	// nonce for HTTP Digest	135	// nonce for HTTP Digest
136	nonce := fx_bitstring2Base64(PX_AuthRAND & v_AUTN);	136	nonce := fx_bitstring2Base64(PX_AuthRAND & v_AUTN);
137	//p_nonce := nonce;	137	//p_nonce := nonce;
138		138	
139	//now we can generate Ks_(ext/int)_NAF = KDF(Ks, "gba-me/u", RAND, IMPI, NA	139	//now we can generate Ks_(ext/int)_NAF = KDF(Ks, "gba-me/u", RAND, IMPI, NA
140	ks_ext_NAF := fx_calculatingKDF4NAF(bit2oct(v_CK & v_IK), fx_char2octet("gk	140	ks_ext_NAF := fx_calculatingKDF4NAF(bit2oct(v_CK & v_IK), fx_char2octet("gk
141	p_ks_ext_NAF := ks_ext_NAF;	141	p_ks_ext_NAF := ks_ext_NAF;
142	ks_int_NAF := fx_calculatingKDF4NAF(bit2oct(v_CK & v_IK), fx_char2octet("gk	142	ks_int_NAF := fx_calculatingKDF4NAF(bit2oct(v_CK & v_IK), fx_char2octet("gk

Datei: LibCommon_GBA_BSF.ttcn (Fortsetzung)

143	p_ks_int_NAF := ks_int_NAF;		143	p_ks_int_NAF := ks_int_NAF;	
144	}		144	}	
145			145		
146	// change 2 (WK23): GBA authentication/authorization extensions for content protect		146	// change 2 (WK23): GBA authentication/authorization extensions for content protect	
147	function BSF_ServerSimulation(boolean p_isParallelComponent) runs on BSFServerComp		147	function BSF_ServerSimulation(boolean p_isParallelComponent) runs on BSFServerComp	
148	var HttpResponse v_response;		148	var HttpResponse v_response;	
149	var HttpRequest v_request;		149	var HttpRequest v_request;	
150			150		
151	var BootstrappingInfoType v_bsfxml;		151	var BootstrappingInfoType v_bsfxml;	
152	var octetstring v_rspauth;		152	var octetstring v_rspauth;	
153			153		
154	log("BSF ready...");		154	log("BSF ready...");	
155	if (not p_isParallelComponent) {		155	if (not p_isParallelComponent) {	
156	t_wait.start(PX_TWAIT);		156	t_wait.start(PX_TWAIT);	
157	}		157	}	
158	alt {		158	alt {	
159	[] bsf.receive(mw_initialHttpRequest) -> value v_request {		159	[] bsf.receive(mw_initialHttpRequest) -> value v_request {	
160	log("Verdict Info: 4a. The terminal sends a POST request wi		160	log("Verdict Info: 4a. The terminal sends a POST request wi	
161	setverdict(pass);		161	setverdict(pass);	
162	bsf.send(m_digestBsfResponse(nonce));		162	bsf.send(m_digestBsfResponse(nonce));	
163	alt {		163	alt {	
164	[] bsf.receive(mw_initialHttpRequest) -> value v_re		164	[] bsf.receive(mw_initialHttpRequest) -> value v_re	
165	// change 4 (WK46/08): SC Registration, mod	<>	165	if (match(str2oct(unq(getHeaderParameterVal	
166	if (match(str2oct(getHeaderParameterValue("		166	if (match(str2oct(unq(getHeaderParameterVal	
167	log("Verdict Info: 4c. RES correns	=	166	log("Verdict Info: 4c. RES correns	
168	setverdict(pass);		167	setverdict(pass);	
169	v_bsfxml := valueof(m_bsfxml(btid,		168	v_bsfxml := valueof(m_bsfxml(btid,	
170	v_rspauth := ts_calculateRspauth(v		169	v_rspauth := ts_calculateRspauth(v	
171	v_response := m_successBsfResponse		170	v_response := m_successBsfResponse	
172	bsf.send(v_response);		171	bsf.send(v_response);	
173	}		172	}	
174	else {		173	else {	
175	log("Verdict Info: RES doesn't corr		174	log("Verdict Info: RES doesn't corr	
176	setverdict(fail);		175	setverdict(fail);	
177	bsf.send(m_basicResponse(400));		176	bsf.send(m_basicResponse(400));	
178	}		177	}	
179	}		178	}	
180	}		179	}	
181	if (p_isParallelComponent) {		180	if (p_isParallelComponent) {	
182	repeat;		181	repeat;	
183	}		182	}	
184	}		183	}	
185	[] bsf.receive {		184	[] bsf.receive {	
186	log("Verdict Info: The terminal doesn't send a POST request		185	log("Verdict Info: The terminal doesn't send a POST request	
187	setverdict(fail);		186	setverdict(fail);	
188	bsf.send(m_basicResponse(400));		187	bsf.send(m_basicResponse(400));	
189	if (p_isParallelComponent) {		188	if (p_isParallelComponent) {	
190	repeat;		189	repeat;	
191	}		190	}	
192	}		191	}	
193	[not p_isParallelComponent] t_wait.timeout {		192	[not p_isParallelComponent] t_wait.timeout {	
194	log("Verdict Info: The terminal doesn't send a HTTP request		193	log("Verdict Info: The terminal doesn't send a HTTP request	
195	setverdict(inconc);		194	setverdict(inconc);	
196	}		195	}	
197	} // alt		196	} // alt	
198	return;		197	return;	
199	}		198	}	
200			199		
201	// change 4 (WK46/08): SC Registration, modified HTTP templates and functions	+-			
202	function ts_AuthResponseCheckAKAv1MD5(in HttpRequest p_httpRequest, in bitstring p_ =		200	function ts_AuthResponseCheckAKAv1MD5(in HttpRequest p_httpRequest, in bitstring p_	
203	// input is the comma separated parameter list: digestResponse from the Aut		201	// input is the comma separated parameter list: digestResponse from the Aut	
204	// and the value of XRES calculated in ts_AuthenticationInit		202	// and the value of XRES calculated in ts_AuthenticationInit	
205	// This function calculates the digest response (v_RequestDigest) for p_XRE		203	// This function calculates the digest response (v_RequestDigest) for p_XRE	
206	// RFC 2617 and compares it with the response received from the UE (v_Dresp		204	// RFC 2617 and compares it with the response received from the UE (v_Dresp	
207			205		
208	var charstring v_Method := p_httpRequest.Method;		206	var charstring v_Method := p_httpRequest.Method;	
209	var ListOfParameter paramList := p_httpRequest.Authorization.ParameterList;		207	var ListOfParameter paramList := p_httpRequest.Authorization.ParameterList;	
210		<>			
211	var charstring v_Username := getHeaderParameterValue("username", paramList)		208	var charstring v_Username := getHeaderParameterValue("username", paramList)	
212	var charstring v_Realm := getHeaderParameterValue("realm", paramList);		209	var charstring v_Realm := getHeaderParameterValue("realm", paramList);	
213	var charstring v_DigestUri := getHeaderParameterValue("uri", paramList);	=	210	var charstring v_DigestUri := getHeaderParameterValue("uri", paramList);	

Datei: LibCommon_GBA_BSF.ttcn (Fortsetzung)

214	var charstring v_Nonce :=	getHeaderParameterValue("nonce", paramList);	<>	211	var charstring v_Nonce :=	getHeaderParameterValue("nonce", paramList);
215	var charstring v_NC :=	getHeaderParameterValue("nc", paramList);		212	var charstring v_NC :=	getHeaderParameterValue("nc", paramList);
216	var charstring v_CNonce :=	getHeaderParameterValue("cnonce", paramList);		213	var charstring v_CNonce :=	getHeaderParameterValue("cnonce", paramList);
217	var charstring v_Qop :=	getHeaderParameterValue("qop", paramList);		214	var charstring v_Qop :=	getHeaderParameterValue("qop", paramList);
218	var charstring v_Dresponse :=	getHeaderParameterValue("response", paramList)	=	215	var charstring v_Dresponse :=	getHeaderParameterValue("response", paramList)
219				216		
220	var octetstring v_oMETHOD :=	fx_char2octet(v_Method);	<>	217	var octetstring v_oMETHOD :=	fx_char2octet(v_Method);
221	var octetstring v_oUsername :=	fx_char2octet(v_Username);		218	var octetstring v_oUsername :=	fx_char2octet(unq(v_Username));
222	var octetstring v_opasswd :=	bit2oct(p_XRES);		219	var octetstring v_opasswd :=	bit2oct(p_XRES);
223			=	220		
224	var octetstring v_oRealm :=	fx_char2octet(v_Realm);	<>	221	var octetstring v_oRealm :=	fx_char2octet(unq(v_Realm));
225	var octetstring v_oDigestUri :=	fx_char2octet(v_DigestUri);		222	var octetstring v_oDigestUri :=	fx_char2octet(v_DigestUri);
226	var octetstring v_oNonce :=	fx_char2octet(v_Nonce);		223	var octetstring v_oNonce :=	fx_char2octet(unq(v_Nonce));
227	var octetstring v_oNC :=	fx_char2octet(v_NC);		224	var octetstring v_oNC :=	fx_char2octet(v_NC);
228	var octetstring v_oCNonce :=	fx_char2octet(v_CNonce);		225	var octetstring v_oCNonce :=	fx_char2octet(unq(v_CNonce));
229	var octetstring v_oQop :=	fx_char2octet(v_Qop);		226	var octetstring v_oQop :=	fx_char2octet(v_Qop);
230	var octetstring v_oDresponse :=	str2oct(v_Dresponse);		227	var octetstring v_oDresponse :=	str2oct(unq(v_Dresponse));
231			=	228		
232	var octetstring A1 :=	'O;	+-			
233	var octetstring A2 :=	'O;				
234	var octetstring v_Secret :=	'O;	=	229	var octetstring v_Secret :=	'O;
235	var octetstring v_HA2 :=	'O;		230	var octetstring v_HA2 :=	'O;
236	var octetstring v_Data :=	'O;		231	var octetstring v_Data :=	'O;
237	var octetstring v_RequestDigest :=	'O;		232	var octetstring v_RequestDigest :=	'O;
238	var octetstring secret_data :=	'O;	+-			
239	var octetstring v_HBody :=	'O;				
240			=	233		
241	log("ts_AuthResponseCheckAKAv1MD5(): METHOD: "	& v_Method & "' ");	<>			
242	log("ts_AuthResponseCheckAKAv1MD5(): client username: "	& v_Username & "' ");				
243	log("ts_AuthResponseCheckAKAv1MD5(): client realm: "	& v_Realm & "' ");				
244	log("ts_AuthResponseCheckAKAv1MD5(): client URI: "	& v_DigestUri & "' ");				
245	log("ts_AuthResponseCheckAKAv1MD5(): client nonce: "	& v_Nonce & "' ");				
246	log("ts_AuthResponseCheckAKAv1MD5(): client nc: "	& v_NC & "' ");				
247	log("ts_AuthResponseCheckAKAv1MD5(): client CNonce: "	& v_CNonce & "' ");				
248	log("ts_AuthResponseCheckAKAv1MD5(): client QoP: "	& v_Qop & "' ");				
249	log("ts_AuthResponseCheckAKAv1MD5(): client response value: "	& v_Dresponse & "' ");				
250						
251	log("ts_AuthResponseCheckAKAv1MD5(): server XRES: "	& oct2str(v_opasswd) & "' ");				
252						
253	// RFC 2617 3.2.2.2		=	234	// RFC 2617 3.2.2.2	
254	// A1 = unq(username-value) ":" unq(realm-value) ":" passwd			235	// A1 = unq(username-value) ":" unq(realm-value) ":" passwd	
255	A1 := v_oUsername & COLON_HEX & v_oRealm & COLON_HEX & v_opasswd;		<>	236	var octetstring A1 := v_oUsername & COLON_HEX & v_oRealm & COLON_HEX & v_opasswd;	
				237		
				238	var octetstring A2, secret_data;	
256			=	239		
257	// change 5 (wk46/08): correction in AKA-MD5 calculation		+-			
258	// H(A1)		=	240	// H(A1)	
259	v_Secret := fx_charLow2octet(oct2str(fx_md5_hex(A1)));			241	v_Secret := fx_charLow2octet(oct2str(fx_md5_hex(A1)));	
260				242		
261	// RFC 2617 3.2.2.3			243	// RFC 2617 3.2.2.3	
262	if (v_Qop == "auth-int")		<>			
263	{					
264	// assuming empty body					
265	v_HBody := fx_charLow2octet(oct2str(fx_md5_hex('O')));					
266						
267	// A2 = Method ":" digest-uri-value ":" H(entity-body)					
268	A2 := v_oMETHOD & COLON_HEX & v_oDigestUri & COLON_HEX & v_HBody;					
269	}					
270	else					
271	{					
272	// A2 = Method ":" digest-uri-value			244	// A2 = Method ":" digest-uri-value	
273	A2 := v_oMETHOD & COLON_HEX & v_oDigestUri;			245	A2 := v_oMETHOD & COLON_HEX & v_oDigestUri;	
274	}					
275				246		
276	v_HA2 := fx_charLow2octet(oct2str(fx_md5_hex(A2)));		=	247	v_HA2 := fx_charLow2octet(oct2str(fx_md5_hex(A2)));	
277				248		
278	// RFC 2617 3.2.2.1			249	// RFC 2617 3.2.2.1	
279	// unq(nonce-value)			250	// unq(nonce-value)	
280	// ":" nc-value			251	// ":" nc-value	
281	// ":" unq(cnonce-value)			252	// ":" unq(cnonce-value)	
282	// ":" unq(qop-value)			253	// ":" unq(qop-value)	

Datei: LibCommon_GBA_BSF.ttcn (Fortsetzung)

283	// ":" H(A2)		254	// ":" H(A2)	
284	v_Data := (v_oNonce & COLON_HEX & v_oNC & COLON_HEX & v_oCNonce		255	v_Data := (v_oNonce & COLON_HEX & v_oNC & COLON_HEX & v_oCNonce	
285			256		
286	//now apply KD function described in RFC 2617 section 3.2.1		257	//now apply KD function described in RFC 2617 section 3.2.1	
287	//KD(secret, data) = H(concat(secret, ":", data))		258	//KD(secret, data) = H(concat(secret, ":", data))	
288	secret_data := v_Secret & COLON_HEX & v_Data;		259	secret_data := v_Secret & COLON_HEX & v_Data;	
289	v_RequestDigest := fx_md5_hex(secret_data);		260	v_RequestDigest := fx_md5_hex(secret_data);	
290			261		
		<>	262	log("ts_AuthResponseCheckAKAv1MD5(): METHOD '" & v_Method & "'");	
			263	log("ts_AuthResponseCheckAKAv1MD5(): client username '" & v_Username & "'");	
			264	log("ts_AuthResponseCheckAKAv1MD5(): client realm '" & v_Realm & "'");	
			265	log("ts_AuthResponseCheckAKAv1MD5(): client URI '" & v_DigestUri & "'");	
			266	log("ts_AuthResponseCheckAKAv1MD5(): client nonce '" & v_Nonce & "'");	
			267	log("ts_AuthResponseCheckAKAv1MD5(): client nc '" & v_NC & "'");	
			268	log("ts_AuthResponseCheckAKAv1MD5(): client CNonce '" & v_CNonce & "'");	
			269	log("ts_AuthResponseCheckAKAv1MD5(): client QoP '" & v_Qop & "'");	
			270	log("ts_AuthResponseCheckAKAv1MD5(): client response value '" & v_Dresponse	
			271		
			272	log("ts_AuthResponseCheckAKAv1MD5(): server XRES: " & oct2str(v_opasswd));	
			273		
291	log("ts_AuthResponseCheckAKAv1MD5(): calculated response value '" & oct2str		274	log("ts_AuthResponseCheckAKAv1MD5(): calculated response value "" & oct2str	
292		=	275		
		+-	276	//	
			277	if (v_RequestDigest == v_oDresponse) {	
			278	return (true)	
			279	} else {	
			280	return (false)	
			281	}	
293	return v_RequestDigest;	=	281	return v_RequestDigest;	
294	}		282	}	
295			283		
296	// change 4 (WK46/08): SC Registration, modified HTTP templates and functions	+-			
297	function ts_calculateRspauth(in ListOfParameter p_list, in bitstring p_XRES, in cha	=	284	function ts_calculateRspauth(in ListOfParameter p_list, in bitstring p_XRES, in cha	
298	// RFC 2617 and 3310		285	// RFC 2617 and 3310	
299	var charstring v_NC := getHeaderParameterValue("nc", p_list);		286	var charstring v_NC := getHeaderParameterValue("nc", p_list);	
300	var charstring v_CNonce := getHeaderParameterValue("cnonce", p_list);		287	var charstring v_CNonce := getHeaderParameterValue("cnonce", p_list);	
301	var charstring v_DigestUri := getHeaderParameterValue("uri", p_list);		288	var charstring v_DigestUri := getHeaderParameterValue("uri", p_list);	
302			289		
303	var octetstring v_oUsername := fx_char2octet(PX_IMPI);		290	var octetstring v_oUsername := fx_char2octet(PX_IMPI);	
304	var octetstring v_opasswd := bit2oct(p_XRES);		291	var octetstring v_opasswd := bit2oct(p_XRES);	
305	var octetstring v_oRealm := fx_char2octet(PX_BSF_FQDN);		292	var octetstring v_oRealm := fx_char2octet(PX_BSF_FQDN);	
306	var octetstring v_oDigestUri := fx_char2octet("/");		293	var octetstring v_oDigestUri := fx_char2octet("/");	
307	var octetstring v_oNonce := fx_char2octet(p_nonce);		294	var octetstring v_oNonce := fx_char2octet(p_nonce);	
308	var octetstring v_oNC := fx_char2octet(v_NC);		295	var octetstring v_oNC := fx_char2octet(v_NC);	
309	var octetstring v_oCNonce := fx_char2octet(v_CNonce);	<>	296	var octetstring v_oCNonce := fx_char2octet(unq(v_CNonce));	
310	var octetstring v_oQop := fx_char2octet("auth-int");	=	297	var octetstring v_oQop := fx_char2octet("auth-int");	
311			298		
312	var octetstring v_Secret := '0;		299	var octetstring v_Secret := '0;	
313	var octetstring v_HA2 := '0;		300	var octetstring v_HA2 := '0;	
314	var octetstring v_Data := '0;		301	var octetstring v_Data := '0;	
315	var octetstring v_RequestDigest := '0;		302	var octetstring v_RequestDigest := '0;	
316			303		
317	// RFC 2617 3.2.2.2		304	// RFC 2617 3.2.2.2	
318	// A1 = unq(username-value) ":" unq(realm-value) ":" passwd		305	// A1 = unq(username-value) ":" unq(realm-value) ":" passwd	
319	var octetstring A1 := v_oUsername & COLON_HEX & v_oRealm & COLON_HEX & v_o		306	var octetstring A1 := v_oUsername & COLON_HEX & v_oRealm & COLON_HEX & v_o	
320			307		
321	var octetstring h_entityBody, A2, secret_data;		308	var octetstring h_entityBody, A2, secret_data;	
322			309		
323	// H(A1)		310	// H(A1)	
324	v_Secret := fx_charLow2octet(oct2str(fx_md5_hex(A1)));		311	v_Secret := fx_charLow2octet(oct2str(fx_md5_hex(A1)));	
325			312		
326	// RFC 2617 3.2.3		313	// RFC 2617 3.2.3	
327	// A2 = ":" digest-uri-value ":" H(entity-body)		314	// A2 = ":" digest-uri-value ":" H(entity-body)	
328	h_entityBody := fx_charLow2octet(oct2str(fx_md5_hex(p_entityBody)));		315	h_entityBody := fx_charLow2octet(oct2str(fx_md5_hex(p_entityBody)));	
329	A2 := COLON_HEX & v_oDigestUri & COLON_HEX & h_entityBody;		316	A2 := COLON_HEX & v_oDigestUri & COLON_HEX & h_entityBody;	
330			317		
331	v_HA2 := fx_charLow2octet(oct2str(fx_md5_hex(A2)));		318	v_HA2 := fx_charLow2octet(oct2str(fx_md5_hex(A2)));	
332			319		
333	// RFC 2617 3.2.2.1		320	// RFC 2617 3.2.2.1	
334	// unq(nonce-value) ":" nc-value ":" unq(cnonce-value) ":" unq(qop-value) "		321	// unq(nonce-value) ":" nc-value ":" unq(cnonce-value) ":" unq(qop-value) "	
335	v_Data := (v_oNonce & COLON_HEX & v_oNC & COLON_HEX & v_oCNonce & COLON_HEX		322	v_Data := (v_oNonce & COLON_HEX & v_oNC & COLON_HEX & v_oCNonce & COLON_HEX	
336			323		

Datei: LibCommon_GBA_BSF.ttcn (Fortsetzung)

337	//now apply KD function described in RFC 2617 section 3.2.1		324	//now apply KD function described in RFC 2617 section 3.2.1
338	//KD(secret, data) = H(concat(secret, ":", data))		325	//KD(secret, data) = H(concat(secret, ":", data))
339	secret_data := v_Secret & COLON_HEX & v_Data;		326	secret_data := v_Secret & COLON_HEX & v_Data;
340			327	
341	log("ts_calculateRspauth(): v_NC -> " & v_NC);		328	log("ts_calculateRspauth(): v_NC -> " & v_NC);
342	log("ts_calculateRspauth(): v_CNonce -> " & v_CNonce);		329	log("ts_calculateRspauth(): v_CNonce -> " & v_CNonce);
343	log("ts_calculateRspauth(): v_oUsername/ixit_IMPI_USIM -> " & PX_IMPI);		330	log("ts_calculateRspauth(): v_oUsername/ixit_IMPI_USIM -> " & PX_IMPI);
344	log("ts_calculateRspauth(): v_opasswd -> " & oct2str(v_opasswd));		331	log("ts_calculateRspauth(): v_opasswd -> " & oct2str(v_opasswd));
345	log("ts_calculateRspauth(): v_oRealm/ixit_BSF_FQDN -> " & PX_BSF_FQDN);		332	log("ts_calculateRspauth(): v_oRealm/ixit_BSF_FQDN -> " & PX_BSF_FQDN);
346	log("ts_calculateRspauth(): v_DigestUri -> " & v_DigestUri);		333	log("ts_calculateRspauth(): v_DigestUri -> " & v_DigestUri);
347	log("ts_calculateRspauth(): v_oNonce/p_nonce -> " & p_nonce);		334	log("ts_calculateRspauth(): v_oNonce/p_nonce -> " & p_nonce);
348	log("ts_calculateRspauth(): v_oNC/v_NC -> " & v_NC);		335	log("ts_calculateRspauth(): v_oNC/v_NC -> " & v_NC);
349	log("ts_calculateRspauth(): v_oCNonce -> " & v_CNonce);		336	log("ts_calculateRspauth(): v_oCNonce -> " & v_CNonce);
350	log("ts_calculateRspauth(): v_oQop -> " & oct2str(v_oQop));		337	log("ts_calculateRspauth(): v_oQop -> " & oct2str(v_oQop));
351			338	
352	return fx_md5_hex(secret_data);		339	return fx_md5_hex(secret_data);
		<>	340	}
			341	
			342	function unq(in charstring p_string) return charstring {
			343	var charstring retVal;
			344	log("unq(): in -> " & p_string);
			345	
			346	if ((substr(p_string, 0, 1) == "\"") and (substr(p_string, lengthof(p_string), 1) == "\""))
			347	retVal := substr(p_string, 1, lengthof(p_string) - 2);
			348	}
			349	else {
			350	retVal := p_string;
			351	}
			352	
			353	log("unq(): out -> " & retVal);
			354	return retVal;
353	}	=	355	}
354			356	
355	function getHeaderParameterValue(in charstring p_headerParameterValue, in ListOfParameters p_list)		357	function getHeaderParameterValue(in charstring p_headerParameterValue, in ListOfParameters p_list)
356	var integer i;		358	var integer i;
357			359	
358	for (i := 0; i < sizeof(p_list); i:= i+1){		360	for (i := 0; i < sizeof(p_list); i:= i+1){
359	if (p_list[i].Name == p_headerParameterValue) {		361	if (p_list[i].Name == p_headerParameterValue) {
360	return (p_list[i].Value);		362	return (p_list[i].Value);
361	}		363	}
362	}		364	}
363			365	
364	return "";		366	return "";
365	}		367	}
366			368	
367	function ts_AuthenticationInit(out bitstring v_IKey, out bitstring v_CKey, out bitstring v_AK, out bitstring v_MAC, out bitstring v_AUTN_1, out bitstring v_AUTN_2)		369	function ts_AuthenticationInit(out bitstring v_IKey, out bitstring v_CKey, out bitstring v_AK, out bitstring v_MAC, out bitstring v_AUTN_1, out bitstring v_AUTN_2)
368	// Calculation is done according to 34.108, clause 8.1.2 and 33.102, clause 8.1.2		370	// Calculation is done according to 34.108, clause 8.1.2 and 33.102, clause 8.1.2
369	var Bit128 v_XDOut, v_AUTN;		371	var Bit128 v_XDOut, v_AUTN;
370	var bitstring v_AUTN_2;		372	var bitstring v_AUTN_2;
371	var Bit64 v_CDOut, v_XDOut_Half, v_MAC;		373	var Bit64 v_CDOut, v_XDOut_Half, v_MAC;
372	var Bit48 v_AK;		374	var Bit48 v_AK;
373	var bitstring v_AUTN_1;		375	var bitstring v_AUTN_1;
374			376	
375	v_XDOut := PX_AuthRAND xor4b PX_AuthK;		377	v_XDOut := PX_AuthRAND xor4b PX_AuthK;
376			378	
377	v_CDOut := PX_AuthSQN & PX_AuthAMF;		379	v_CDOut := PX_AuthSQN & PX_AuthAMF;
378			380	
379	v_XDOut_Half := substr (v_XDOut, 0, 64);		381	v_XDOut_Half := substr (v_XDOut, 0, 64);
380			382	
381	v_AK := substr (v_XDOut, 24, 48);		383	v_AK := substr (v_XDOut, 24, 48);
382			384	
383	v_AUTN_1 := PX_AuthSQN xor4b v_AK;		385	v_AUTN_1 := PX_AuthSQN xor4b v_AK;
384			386	
385	v_MAC := v_XDOut_Half xor4b v_CDOut;		387	v_MAC := v_XDOut_Half xor4b v_CDOut;
386			388	
387	v_AUTN_2 := PX_AuthAMF & v_MAC;		389	v_AUTN_2 := PX_AuthAMF & v_MAC;
388			390	
389	v_AUTN := v_AUTN_1 & v_AUTN_2;		391	v_AUTN := v_AUTN_1 & v_AUTN_2;
390			392	
391	// v_IKey := 128 bits of v_XDOut, wrapped, starting from offset 16		393	// v_IKey := 128 bits of v_XDOut, wrapped, starting from offset 16
392	v_IKey :=		394	v_IKey :=

Datei: LibCommon_GBA_BSF.ttcn (Fortsetzung)

393	substr (v_XDOut, 16, (128 - 16)) & substr (v_XDOut, 0, 16);	395	substr (v_XDOut, 16, (128 - 16)) & substr (v_XDOut, 0, 16);
394		396	
395	// v_CKey := 128 bits of v_XDOut, wrapped, starting from offset 8	397	// v_CKey := 128 bits of v_XDOut, wrapped, starting from offset 8
396	v_CKey :=	398	v_CKey :=
397	substr (v_XDOut, 8, (128 - 8)) & substr (v_XDOut, 0, 8);	399	substr (v_XDOut, 8, (128 - 8)) & substr (v_XDOut, 0, 8);
398		400	
399	v_XRES := substr (v_XDOut, 0, (PX_AuthN + 1));	401	v_XRES := substr (v_XDOut, 0, (PX_AuthN + 1));
400		402	
401	return (v_AUTN);	403	return (v_AUTN);
402	}	404	}
403		405	
404	// change 2 (WK23): GBA authentication/authorization extensions for content protection	406	// change 2 (WK23): GBA authentication/authorization extensions for content protection
405	function ts_AuthenticationInitForGbaU(out bitstring v_IKey, out bitstring v_CKey)	407	function ts_AuthenticationInitForGbaU(out bitstring v_IKey, out bitstring v_CKey)
406	// Calculation is done according to 34.108, clause 8.1.2 and 33.102, clause 8.1.2	408	// Calculation is done according to 34.108, clause 8.1.2 and 33.102, clause 8.1.2
407	var Bit128 v_XDOut, v_AUTN;	409	var Bit128 v_XDOut, v_AUTN;
408	var Bit64 v_CDOut, v_XDOut_Half, v_MAC, v_MAC_;	410	var Bit64 v_CDOut, v_XDOut_Half, v_MAC, v_MAC_;
409	var Bit48 v_AK;	411	var Bit48 v_AK;
410		412	
411	v_XDOut := PX_AuthRAND xor4b PX_AuthK;	413	v_XDOut := PX_AuthRAND xor4b PX_AuthK;
412		414	
413	v_CDOut := PX_AuthSQN & PX_AuthAMF;	415	v_CDOut := PX_AuthSQN & PX_AuthAMF;
414		416	
415	v_XDOut_Half := substr (v_XDOut, 0, 64);	417	v_XDOut_Half := substr (v_XDOut, 0, 64);
416		418	
417	v_AK := substr (v_XDOut, 24, 48);	419	v_AK := substr (v_XDOut, 24, 48);
418		420	
419	v_MAC := v_XDOut_Half xor4b v_CDOut;	421	v_MAC := v_XDOut_Half xor4b v_CDOut;
420		422	
421	// v_IKey := 128 bits of v_XDOut, wrapped, starting from offset 16	423	// v_IKey := 128 bits of v_XDOut, wrapped, starting from offset 16
422	v_IKey :=	424	v_IKey :=
423	substr (v_XDOut, 16, (128 - 16)) & substr (v_XDOut, 0, 16);	425	substr (v_XDOut, 16, (128 - 16)) & substr (v_XDOut, 0, 16);
424		426	
425	// v_CKey := 128 bits of v_XDOut, wrapped, starting from offset 8	427	// v_CKey := 128 bits of v_XDOut, wrapped, starting from offset 8
426	v_CKey :=	428	v_CKey :=
427	substr (v_XDOut, 8, (128 - 8)) & substr (v_XDOut, 0, 8);	429	substr (v_XDOut, 8, (128 - 8)) & substr (v_XDOut, 0, 8);
428		430	
429	v_MAC_ := v_MAC xor4b substr(oct2bit((fx_sha_1_hex(bit2oct(v_IKey)))), 0, 64);	431	v_MAC_ := v_MAC xor4b substr(oct2bit((fx_sha_1_hex(bit2oct(v_IKey)))), 0, 64);
430		432	
431	v_AUTN_ := (PX_AuthSQN xor4b v_AK) & PX_AuthAMF & v_MAC_;	433	v_AUTN_ := (PX_AuthSQN xor4b v_AK) & PX_AuthAMF & v_MAC_;
432		434	
433	// length of v_XRES is depending on PX_AuthN	435	// length of v_XRES is depending on PX_AuthN
434	v_XRES := substr (v_XDOut, 0, (PX_AuthN + 1));	436	v_XRES := substr (v_XDOut, 0, (PX_AuthN + 1));
435	// flipping the last significant bit	437	// flipping the last significant bit
436	v_XRES := v_XRES xor4b int2bit(1, PX_AuthN + 1);	438	v_XRES := v_XRES xor4b int2bit(1, PX_AuthN + 1);
437		439	
438	return (v_AUTN_);	440	return (v_AUTN_);
439	}	441	}
440		442	
441		443	
442	/**	444	/**
443	*	445	*
444	* @desc BSF generate B-TID (Bootstrapping Transaction Identifier) -> TS 33.220 4.5.1	446	* @desc BSF generate B-TID (Bootstrapping Transaction Identifier) -> TS 33.220 4.5.1
445	* @return B-TID as charstring	447	* @return B-TID as charstring
446	*/	448	*/
447	function f_bsf_generateBtid() return charstring {	449	function f_bsf_generateBtid() return charstring {
448	return fx_bitstring2Base64(PX_AuthRAND) & "@" & PX_BSF_FQDN;	450	return fx_bitstring2Base64(PX_AuthRAND) & "@" & PX_BSF_FQDN;
449	}	451	}
450	// change 2 (WK23): GBA authentication/authorization extensions for content protection	452	// change 2 (WK23): GBA authentication/authorization extensions for content protection
451	/**	453	/**
452	*	454	*
453	* @desc Sets a new B-TID for BSF.	455	* @desc Sets a new B-TID for BSF.
454	* @param p_btid	456	* @param p_btid
455	* @verdict	457	* @verdict
456	*/	458	*/
457	function f_bsf_setBtid(charstring p_btid) runs on BSFServerComponent {	459	function f_bsf_setBtid(charstring p_btid) runs on BSFServerComponent {
458	btid := p_btid;	460	btid := p_btid;
459	}	461	}
460		462	
461	const octetstring COLON_HEX := '3A'O;	463	const octetstring COLON_HEX := '3A'O;
462		464	
463	}	465	}

Datei: LibCommon_GBA_BSF.ttcn (Fortsetzung)

464	group BSF_Configuration_and_Components {	466	group BSF_Configuration_and_Components {
465	type component BSFServerComponent {	467	type component BSFServerComponent {
466	port BSFPort bsf;	468	port BSFPort bsf;
467		469	
468	timer t_wait;	470	timer t_wait;
469		471	
470	// change 2 (WK23): GBA authentication/authorization extensions for content protection	472	// change 2 (WK23): GBA authentication/authorization extensions for content protection
471	var charstring btid;	473	var charstring btid;
472	var charstring nonce;	474	var charstring nonce;
473	var octetstring ks_ext_NAF, ks_int_NAF;	475	var octetstring ks_ext_NAF, ks_int_NAF;
474	var bitstring xres;	476	var bitstring xres;
475	}	477	}
476		478	
477	type port BSFPort message {	479	type port BSFPort message {
478	in HttpRequest;	480	in HttpRequest;
479	out HttpResponse	481	out HttpResponse
480	}	482	}
481	}	483	}
482	}	484	}
483		485	
484	group externalFunctions {	486	group externalFunctions {
485	external function fx_char2octet(in charstring p_string) return octetstring;	487	external function fx_char2octet(in charstring p_string) return octetstring;
486	external function fx_charLow2octet(charstring p_string) return octetstring; //transformes c	488	external function fx_charLow2octet(charstring p_string) return octetstring; //transformes c
487	//function to calculate the MD5 Message-Digest Algorithm according to RFC 1321	489	//function to calculate the MD5 Message-Digest Algorithm according to RFC 1321
488	external function fx_md5_hex(octetstring p_data) return octetstring;	490	external function fx_md5_hex(octetstring p_data) return octetstring;
489	external function fx_bitstring2Base64(in bitstring p_bitstring) return charstring;	491	external function fx_bitstring2Base64(in bitstring p_bitstring) return charstring;
490	/**	492	/**
491	* @desc Converts the system time in the RFC 1123 format (e.g. 'Thu, 08 Jan 2004 10:23:17 C	493	* @desc Converts the system time in the RFC 1123 format (e.g. 'Thu, 08 Jan 2004 10:23:17 C
492	* @param p_additionalDays system time + 'p_additionalDays' => return value	494	* @param p_additionalDays system time + 'p_additionalDays' => return value
493	* @return date in RFC 1123 format	495	* @return date in RFC 1123 format
494	* @verdict	496	* @verdict
495	*/	497	*/
496	external function fx_date2RFC1123(in integer p_additionalDays) return charstring;	498	external function fx_date2RFC1123(in integer p_additionalDays) return charstring;
497	/**	499	/**
498	* @desc Converts the system time in the UTC format (e.g. '2007-07-24T13.20:00Z').	500	* @desc Converts the system time in the UTC format (e.g. '2007-07-24T13.20:00Z').
499	* @param p_additionalDays system time + 'p_additionalDays' => return value	501	* @param p_additionalDays system time + 'p_additionalDays' => return value
500	* @return date in UTC format	502	* @return date in UTC format
501	* @verdict	503	* @verdict
502	*/	504	*/
503	external function fx_date2UTC(in integer p_additionalDays) return charstring;	505	external function fx_date2UTC(in integer p_additionalDays) return charstring;
504	external function fx_bsfxml2Oct(in BootstrappingInfoType p_bsfxml) return octetstring;	506	external function fx_bsfxml2Oct(in BootstrappingInfoType p_bsfxml) return octetstring;
505	// change 2 (WK23): GBA authentication/authorization extensions for content protection test	507	// change 2 (WK23): GBA authentication/authorization extensions for content protection test
506	external function fx_sha_1_hex(octetstring p_data) return octetstring;	508	external function fx_sha_1_hex(octetstring p_data) return octetstring;
507	// change 2 (WK23): GBA authentication/authorization extensions for content protection test	509	// change 2 (WK23): GBA authentication/authorization extensions for content protection test
508	external function fx_calculatingKDF4NAF(in octetstring p_keys, in octetstring p_kdfType,	510	external function fx_calculatingKDF4NAF(in octetstring p_keys, in octetstring p_kdfType,
509	in	511	in
510	in	512	in
511	}	513	}
512	}	514	}