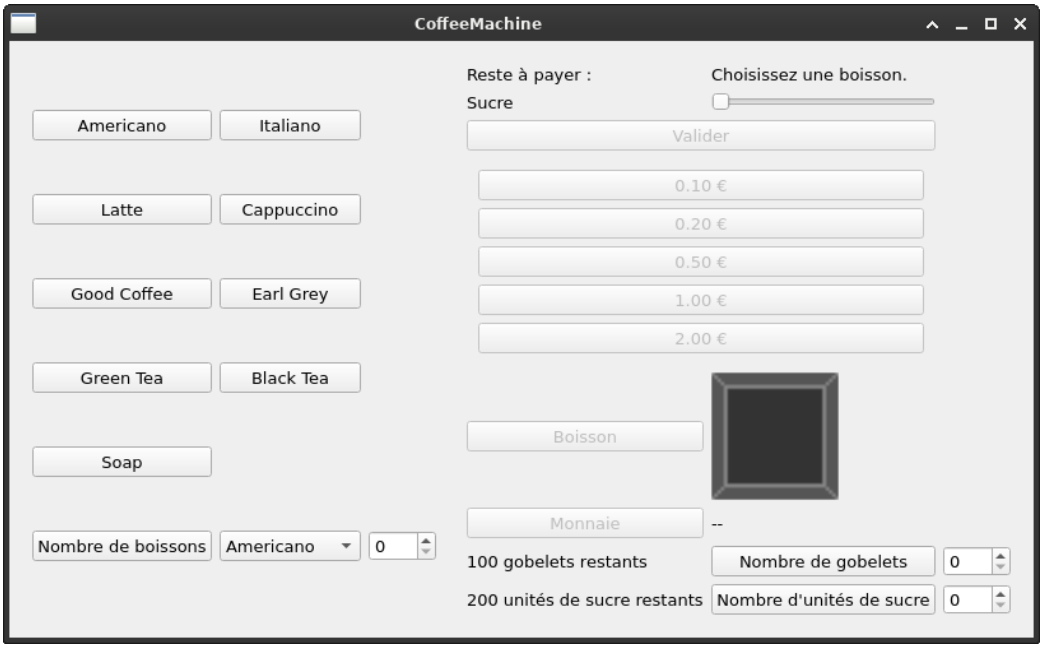
Table des matières

1 Introduction



Le projet consiste à réaliser un outil de test permettant de vérifier le bon fonctionnement d'une machine à café. Avant de commencer à lire la suite, notez qu'il ne faut pas lire les annexes tant qu'il n'est pas indiqué qu'il est possible de le faire.

1.1 Spécification de la machine à café

La machine à café est composée de deux parties : `CoffeeMachineDrinks` à gauche et `CoffeeMachineInput` à droite (cf. Figure 1). `CoffeeMachineDrinks` permet de sélectionner une boisson. La sélection d'une boisson modifie la partie de droite qui affiche le prix de la boisson sélectionnée (cf. Figure 2). Une

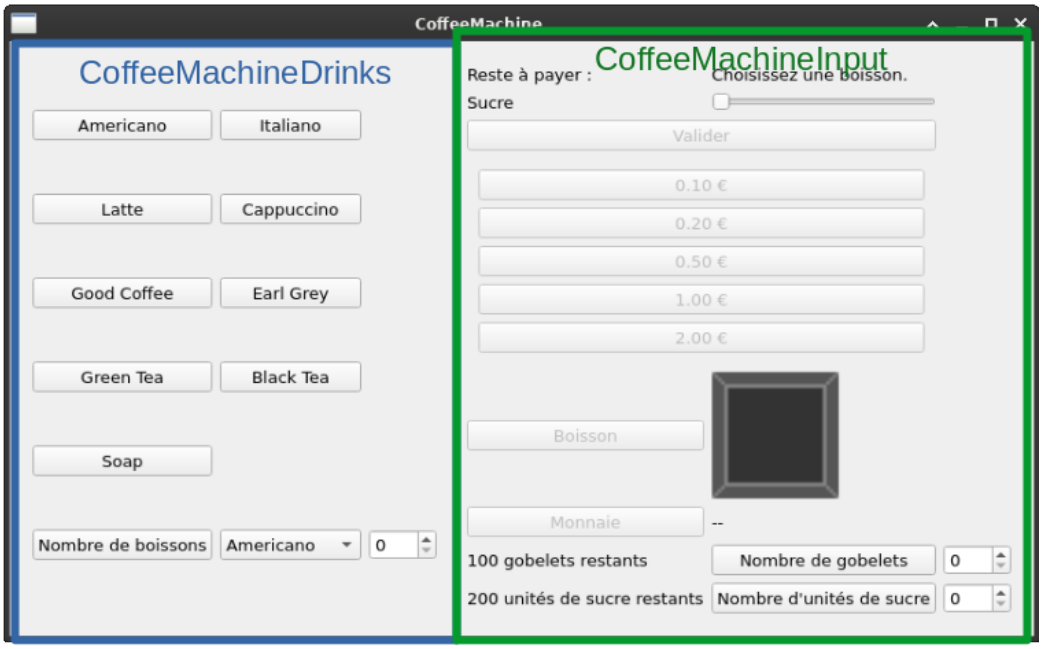


FIGURE 1 – Composants de la machine à café

fois la boisson validée, l'utilisateur peut alors payer en insérant des pièces (cf. Figure 3). Une fois la boisson payée, l'utilisateur peut récupérer sa boisson et sa potentielle monnaie (cf. Figure 4). Lorsqu'une boisson est manquante, celle-ci ne peut plus être sélectionnée. De la même manière, on ne peut pas sélectionner plus de sucre que ce qui reste dans la machine à café, et lorsqu'il n'y a plus de gobelets, les boissons sont indisponibles (cf. Figures 5 à 7). Pour chaque boisson, il n'est pas possible d'avoir plus de dix unités de sucre. Enfin, il est possible de modifier la quantité de chaque boisson, ainsi que le nombre de gobelets et d'unités de sucre dans la machine. Notez que certaines boissons ne peuvent pas être sucrées.

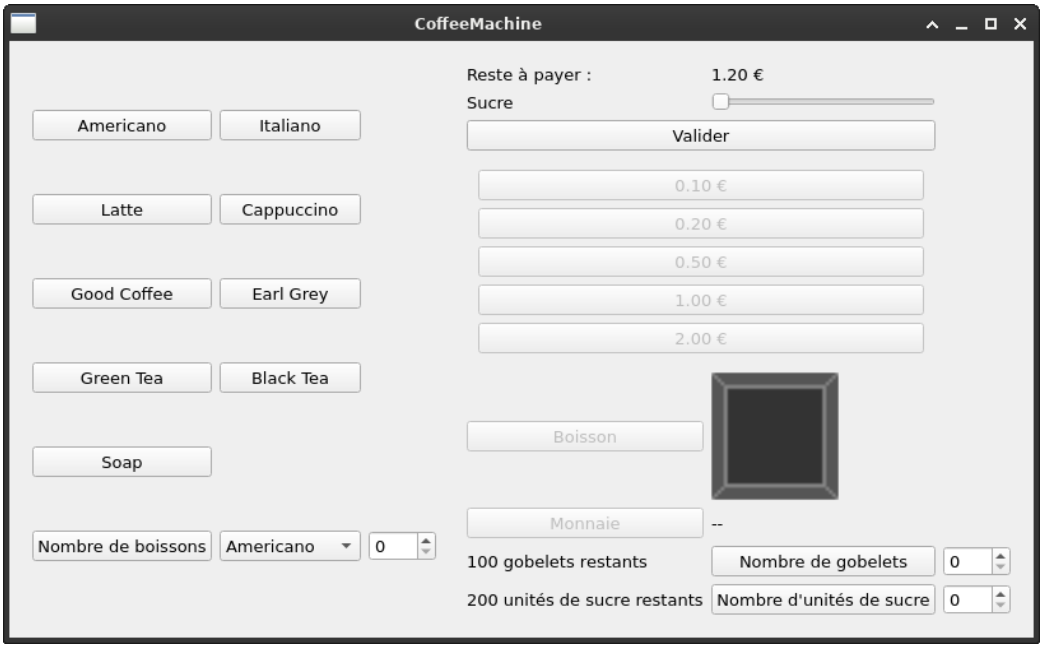


FIGURE 2 – Boisson sélectionnée

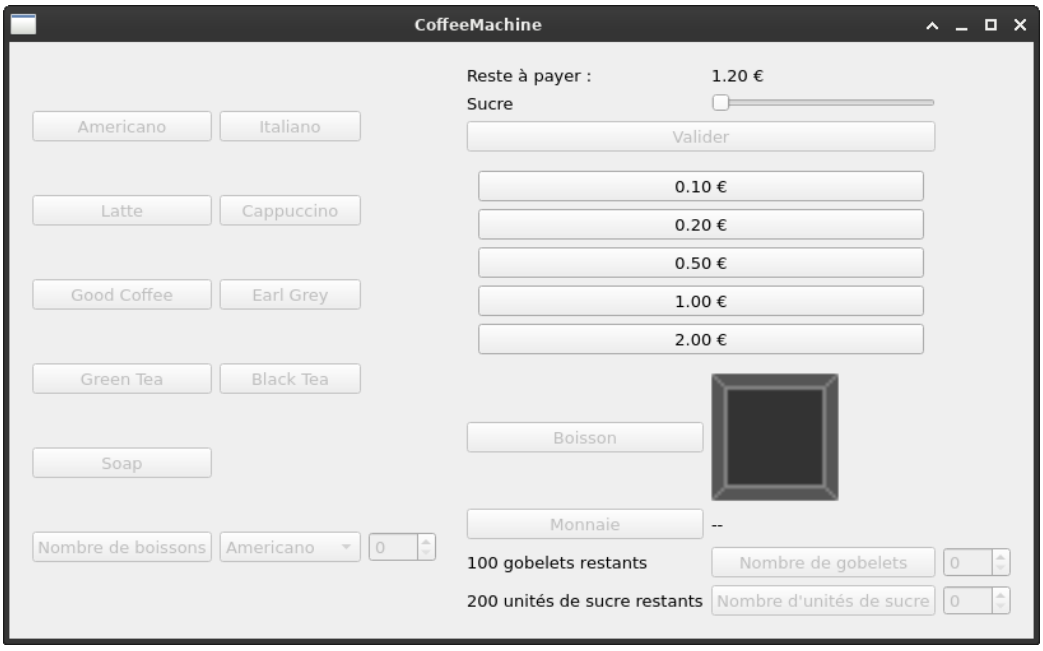


FIGURE 3 – Boisson validée

FIGURE 4 – Boisson payée

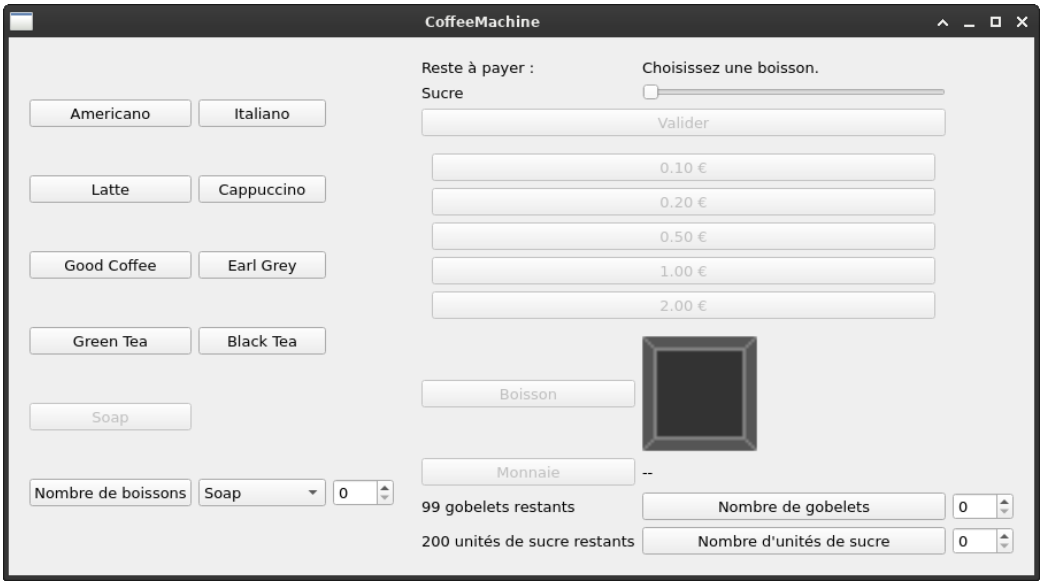


FIGURE 5 – *Soap* est indisponible

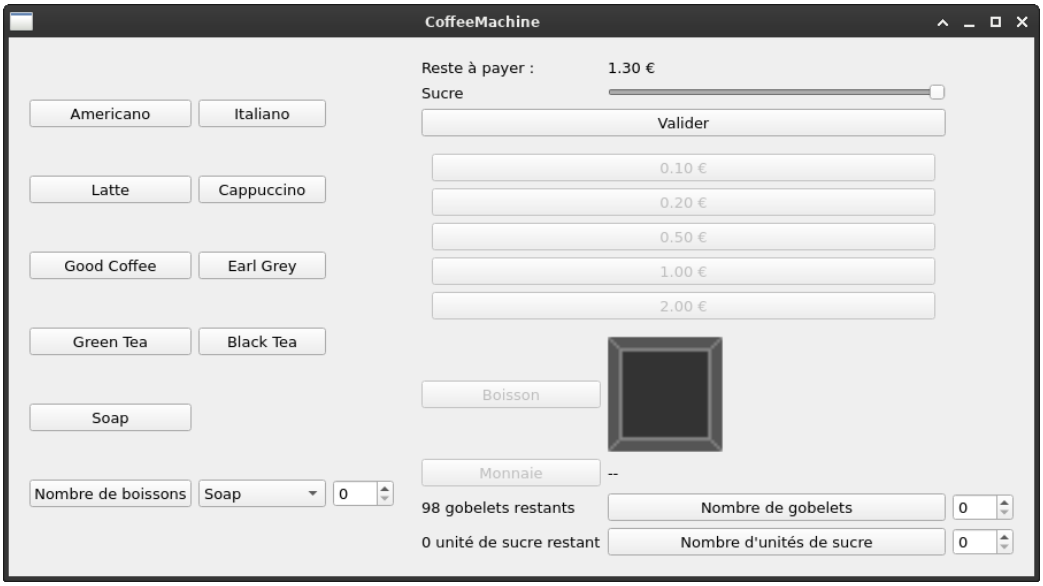


FIGURE 6 – Le sucre est indisponible

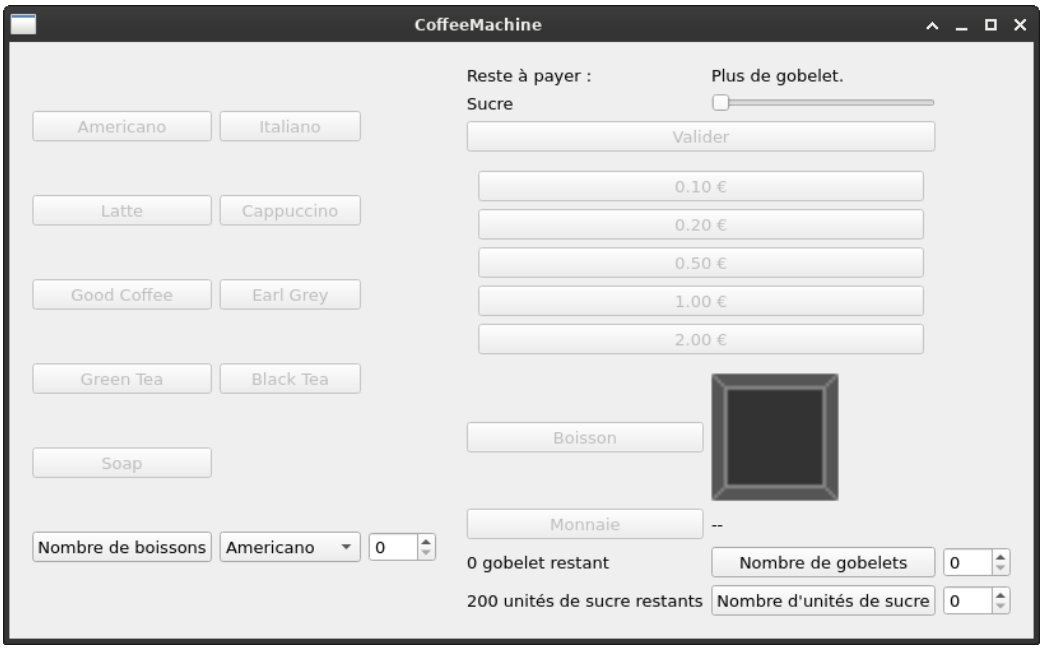


FIGURE 7 – Il n'y a plus de gobelet

1.2 Spécification technique de la machine à café

1. Tant qu'aucune boisson n'est sélectionnée, aucun prix n'est affiché.
2. Tant qu'aucune boisson n'est sélectionnée, la boisson ne peut pas être validée.
3. Lorsqu'une boisson est sélectionnée, son prix est affiché.
4. Lorsqu'on sélectionne une boisson, puis une autre, le prix affiché est mis à jour.
5. Lorsqu'une boisson est sélectionnée, il est possible de sélectionner le taux de sucre.
6. Si une boisson n'a pas de sucre, le taux de sucre ne peut pas être modifié.
7. Le nombre d'unités de sucre ne peut pas être plus grand que 10.
8. Le nombre d'unités de sucre ne peut pas être plus petit que 0.
9. Tant qu'aucune boisson n'est validée, il n'est pas possible d'insérer de pièces.
10. Tant qu'aucune boisson n'est validée, il n'est pas possible de récupérer le rendu monnaie.
11. Tant qu'aucune boisson n'est validée, il n'est pas possible de récupérer de boisson.
12. Lorsqu'une boisson est validée, il n'est pas possible de sélectionner une boisson.
13. Lorsqu'une boisson est validée, il est possible d'insérer de la monnaie.
14. Tant que la boisson n'est pas entièrement payée, il n'est pas possible de récupérer le rendu monnaie.
15. Tant que la boisson n'est pas entièrement payée, il n'est pas possible de récupérer de boisson.
16. Tant que la boisson n'est pas entièrement payée, il n'est pas possible de sélectionner de boisson.
17. Lorsqu'une pièce est insérée, le prix restant à payer affiché est mis à jour.
18. Lorsque la boisson est entièrement payée, il n'est plus possible d'insérer de pièces.

19. Lorsqu'une boisson est entièrement payée, le texte affiché invite à récupérer la boisson.
20. Lorsque la boisson est entièrement payée, la boisson est disponible.
21. Si trop de pièces ont été insérées pour payer la boisson, le rendu monnaie est disponible.
22. Tant que la boisson n'est pas récupérée, il est impossible de sélectionner la boisson.
23. Tant qu'il reste de la monnaie à récupérer, il est impossible de sélectionner la boisson.
24. Lorsque la boisson est récupérée, s'il reste de la monnaie, le texte affiché invite à récupérer la monnaie.
25. Lorsqu'une boisson et le rendu monnaie sont récupérés, le nombre de gobelets est diminué de un.
26. Lorsqu'une boisson et le rendu monnaie sont récupérés, le nombre d'unité de sucre est diminué d'autant de sucre sélectionné.
27. Lorsqu'une boisson et le rendu monnaie sont récupérés, le nombre d'unité de sucre n'est pas diminué si la boisson ne peut pas avoir de sucre.
28. Lorsqu'une boisson est manquante, on ne peut pas la sélectionner.
29. Lorsqu'il n'y a plus de gobelet, on ne peut pas sélectionner de boisson.
30. On ne peut pas sélectionner plus de sucre que le nombre d'unité de sucre restant.
31. Il est possible pour une boisson donnée de sélectionner le nombre restant.
32. Il est possible de sélectionner le nombre d'unités de sucre restants.
33. Il est possible de sélectionner le nombre de gobelets restants.

2 Objectifs de tests

Lorsque l'on fait du test de conformité à un standard ou a une spécification, il faut commencer par faire des objectifs de tests (*Test Purpose* en anglais, et TP dans le reste du document). Ces TP ont pour but de pouvoir être utilisés par tout le monde et définir ce qui doit être testé sur l'équipement (appelé

IUT pour *Implementation Under Test*). Pour écrire ces TP, il faut donc prendre le standard ou la spécification, et faire un test dès qu'une situation est évaluable. Par exemple, des **doit**, **ne doit pas** etc. Un TP s'écrit tel que le Tableau 1. Avec :

- **TP Id** l'identifiant du TP. Un identifiant prend conventionnellement la forme `TP/STD/SEC/.../TYPE-ID`
  - `STD` est à remplacer par une abréviation du standard,
  - `SEC/.../` est une succession d'abréviation de sections du standard,
  - `TYPE` peut valoir
    - `BV` pour un comportement valide,
    - `BO` pour un comportement non valide dans le temps,
    - `BI` pour un comportement invalide,
    - `TI` pour un comportement lié à un *timer*.
  - `ID` un numéro.
- **Test objective** une description brève du TP.
- **Reference** les références au standard.
- **PICS Selection** les PICS que l'équipement à tester doit valider pour passer le test.
- **Initial conditions** les conditions initiales pour faire le test.
- **Expected behavior** le corps du test.

Différence entre `BO` et `BI` : le fait de vouloir valider une boisson tant qu'aucune n'est sélectionnée est un `BO`. Le comportement (le fait d'appuyer sur valider) est non valide car la boisson n'est pas sélectionnée. Le fait de vouloir sélectionner une boisson qui n'existe pas est un `BI`, car elle n'existera jamais. Notez qu'il est aussi possible de rajouter une ligne **Config Id** si votre test nécessite certaines configurations. Ces configurations sont surtout utilisées lorsqu'il y a plusieurs équipements simulés en plus de l'IUT. Enfin, les PICS (*Protocol Implementation Conformance Statement*) sont des booléens qui correspondent à des fonctionnalités. Avant de passer les tests, le fournisseur de l'IUT devra indiquer quels PICS sont implémentés. Si un test n'a pas de fonctionnalité particulière, et donc faisable peu importe les fonctionnalités implémentées, alors le TP n'aura pas de sélection de PICS. Par convention, les noms des PICS sont en capitale.

TP Id	
Test objective	
Reference	
PICS Selection	
Initial conditions	

`with {`
`  conditions`
`}`	

Expected behaviour	

`ensure that {`
`  when{`
`  steps`
`  }`
`  then{`
`  results`
`  }`
`}`	

TABLE 1 – Objectif de test

TP Id	TP/GEONW/FDV/BEA/BV-02
Test objective	GeoNetworking address validity test
Reference	ETSI EN 302 636-4-1 [1], clauses 6.3 and 9.8.6.2
Config Id	CF01
PICS Selection	PICS_GN_BEACON_SRC
Initial conditions	

with {
the IUT being in the "initial state"
}	

Expected behaviour	

ensure that {
when{
the IUT generates a Beacon packet
}
then{
the IUT sends a GeoNetworking packet
containing SOPV field
containing GN_ADDR field
containing ST field
indicating the ITS Station type
containing SCC field
indicating the ITS Station country code
}
}	

TABLE 2 – Exemple d'objectif de test

► **1** D'après les spécifications de la machine à café, définissez les différents TP à implémenter dans votre outil de test.

3 Détails de tests

Une fois que les TP sont définis, chaque entité (que j'appelle Testeur dans la suite du document) voulant effectuer des tests va écrire des détails de tests. Ces détails de tests indiqueront donc les modalités pour pouvoir tester son IUT avec ce Testeur. Lors de l'utilisation de variables qui peuvent être

configurées dans les plans de test, nous utilisons ce que l'on appelle des PIXIT (*Protocol Implementation eXtra Information for Testing*). Ces PIXIT sont des valeurs qui sont soit indiquées par le fournisseur, mais qui peuvent être modifiées avant de lancer l'IUT, soit des variables de tests. Par exemple, la position géographique d'un IUT est un PIXIT indiqué par le fournisseur, modifiable. Un autre exemple de PIXIT est le temps accordé à l'IUT pour répondre à une requête. Par convention, les PIXIT sont en minuscules avec des _ pour séparer les mots.

3.1 `UpperTester`

Lorsque nous avons une boîte noire telle que notre machine à café, nous devons être capable de simuler l'environnement de façon automatique. Pour ce faire, nous faisons appel à un module qui s'appelle un `UpperTester` (UT). Cet UT est défini par le Testeur et implémenté par le fournisseur. Ce qui veut dire qu'un fournisseur doit implémenter les spécifications de l'UT du Testeur qui va valider son équipement. Le Testeur va donc communiquer l'état de l'environnement de l'IUT avec l'UT. Cette communication est définie par le Testeur. Pour le projet, je vous impose l'utilisation d'une communication UDP. La simulation se fait *via* ce qu'on appelle des primitives. La définition de ces primitives d'un UT se fait comme les Figures 8 à 10. Il existe trois types de primitives : les requêtes, les réponses et les indications. Les requêtes se font dans le sens Testeur vers IUT. Les deux autres se font dans le sens IUT vers Testeur. Les requêtes impliquent une réponse de la part de l'IUT. Les indications sont des primitives envoyées par l'IUT sans la demande du Testeur. Notez que l'UT doit **toujours** répondre sur le dernier port utilisé par le Testeur pour envoyer une requête. Ce qui signifie que tant que le Testeur n'a pas fait de requête, l'UT ne peut pas faire d'indication.

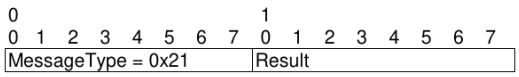
Request (UtCamTrigger_changeSpeed TS ➔ UT):

0								1								2							
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
MessageType = 0x31								SpeedVariation															

Name	Length	Value
MessageType	1 byte	0x31
SpeedVariation	2 bytes	Signed integer. Speed variation in units of cm/s

FIGURE 8 – Exemple de primitive de type requête

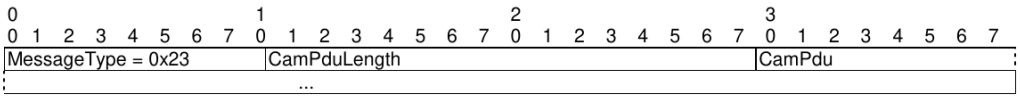
Response (UtCamTriggerResult UT ➔ TS):



Name	Length	Value
MessageType	1 byte	0x21
Result	1 byte	0x00: Failure 0x01: Success

FIGURE 9 – Exemple de primitive de type réponse

Indication (UtCamEventInd UT ➔ TS):



Name	Length	Value
MessageType	1 byte	0x23
CamPduLength	2 bytes	Length of 'CamPdu' field
CamPdu	Variable	Received CAM

FIGURE 10 – Exemple de primitive de type indication

Lorsque l'on exécute un test, celui-ci peut avoir quatre statuts

1. `pass` → le comportement attendu a eu lieu.
2. `fail` → un comportement inattendu a eu lieu.
3. `inconc` → il est impossible de conclure.
4. `error` → le test n'a pas pu aboutir (généralement un crash ou une erreur de codec).

Le cas de l'`inconc` est un cas particulier. Il n'est pas à considérer comme une erreur. Par exemple, si l'UT ne répond pas à une requête, c'est inconclusif. On ne peut pas savoir si cela vient du réseau ou de l'UT en lui-même.
Un détail de tests prendra la forme d'un algorithme. Par exemple :

```
1 : Si PICS.INFORMATION == FAUX Alors
2 :    Retourner  inconc
3 : Fin Si
4 : Envoyer UtInitialize à l'IUT
5 : Démarrer t0 d'une durée PIXIT.tc_noac
6 : alt
7 : [] UTPort.receive(UtInitializeResult.Success == FAUX)
8 :    arrêter t0
9 :    Retourner fail
10 : [] UTPort.receive(UtInitializeResult.Success == VRAI)
11 :    arrêter t0
12 : [] t0.timeout
13 :    Retourner inconc
14 : Envoyer UtGetPrint à l'IUT
15 : Démarrer t0 d'une durée PIXIT.tc_noac
16 : alt
17 : [] UTPort.receive(UtPrintResult.Value == PIXIT.no_drink_selected)
18 :    arrêter t0
19 :    Retourner pass
20 : [] UTPort.receive(UtPrintResult.Value)
21 :    arrêter t0
22 :    Retourner fail
23 : [] t0.timeout
24 :    Retourner inconc
```

Étant donné que vous allez valider ma machine à café qui est déjà implémentée, avec l'UT, nous allons procéder « à l'envers » dans le processus d'écriture des détails de tests, et utiliser les primitives indiquées en annexe (à ne pas lire pour le moment).

▶ **1** Écrivez les détails de tests, ainsi que les différentes primitives UT dont vous auriez besoin pour valider la machine à café selon vos TP.

4 Rapport de test

Le rapport de test peut être sous diverses formes (PDF, texte, tableur). Cependant, il convient d'avoir toujours certaines informations. À savoir :

— la date du test,

— l'identifiant du Testeur,
— la version de l'outil de test,
— l'identifiant de l'IUT,
— la version de l'IUT,
— lorsqu'un message est envoyé ou reçu, la nature du message (avec son contenu si possible),
— le comportement (ou la valeur) attendu,
— le comportement (ou la valeur) observé.

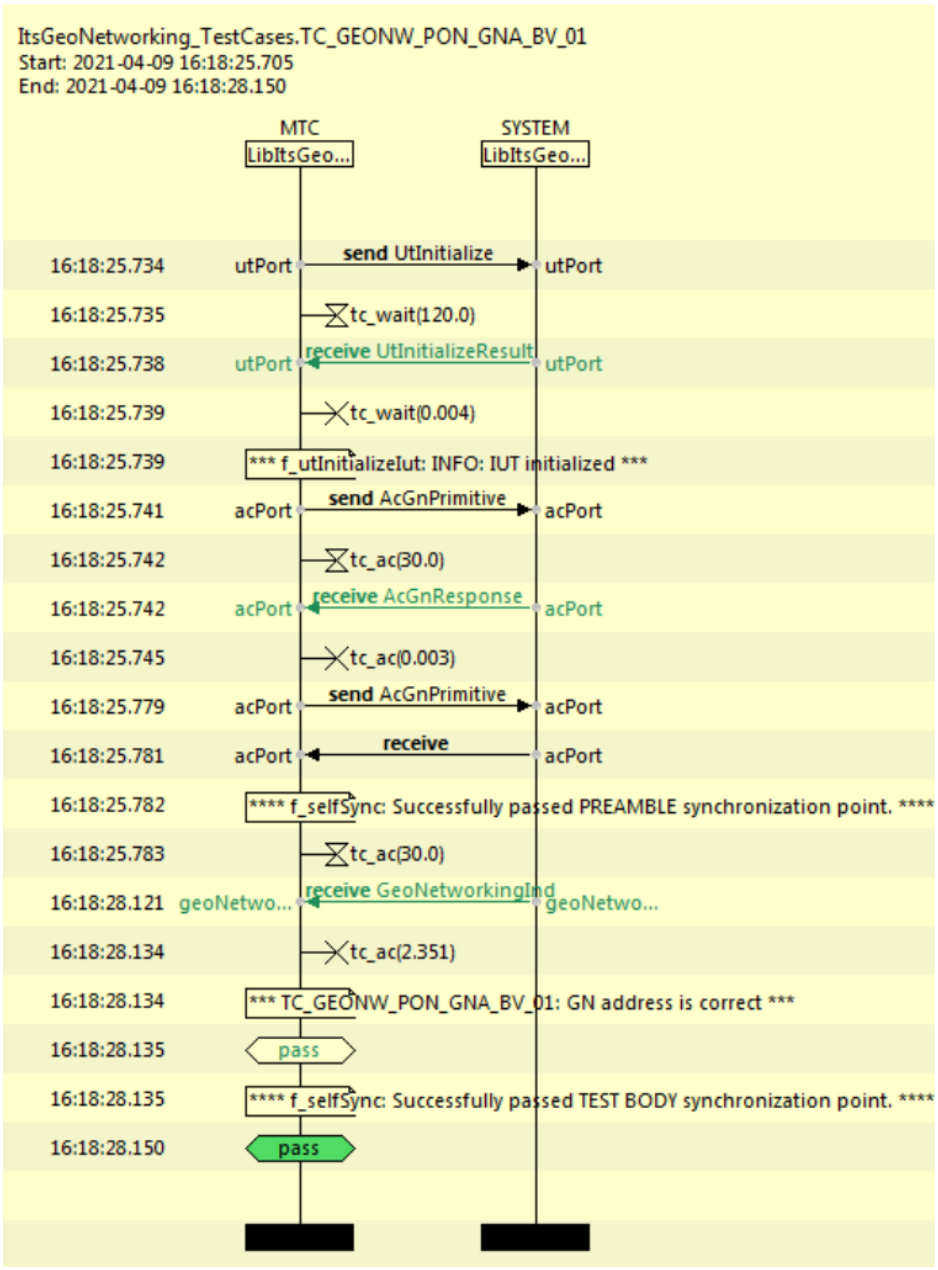


FIGURE 11 – Exemple de rapport au format PDF

```
2021/Apr/02 11:33:52.916469 ItsCam_TestCases.ttcn:55->ItsCam_TpFunctions.ttcn:89 Matching on port camPort succeeded:  matched
2021/Apr/02 11:33:52.916539 ItsCam_TestCases.ttcn:55->ItsCam_TpFunctions.ttcn:89 Receive operation on port camPort succeeded, message from system(): @LibItsCam_TestSystem.CamInd : {
    msgIn := {
        header := {
            protocolVersion := 2,
            messageID := 2,
            stationID := 3576831241
        },
        cam := {
            generationDeltaTime := 59149,
            camParameters := {
                basicContainer := {
                    stationType := 15,
                    referencePosition := {
                        latitude := 492416147,
                        longitude := 40636518,
                        positionConfidenceEllipse := {
                            semiMajorConfidence := 100,
                            semiMinorConfidence := 100,
                            semiMajorOrientation := 3601
                        },
                        altitude := {
                            altitudeValue := 1200,
                            altitudeConfidence := unavailable (15)
                        }
                    }
                },
                highFrequencyContainer := {
                    rsuContainerHighFrequency := {
                        protectedCommunicationZonesRSU := omit
                    }
                },
                lowFrequencyContainer := omit,
                specialVehicleContainer := omit
            }
        }
    },
    gnNextHeader := 2,
    gnHeaderType := 5,
    gnHeaderSubtype := 0,
    gnLifetime := 1000,
    gnTrafficClass := 2,
    btpDestinationPort := 2001,
    btpInfo := 0
} id 3
2021/Apr/02 11:33:52.916623 ItsCam_TestCases.ttcn:55->ItsCam_TpFunctions.ttcn:89 Message with id 3 was extracted from the queue of camPort.
2021/Apr/02 11:33:52.916681 ItsCam_TestCases.ttcn:55->ItsCam_TpFunctions.ttcn:90 Stop timer tc_ac: 30 s
2021/Apr/02 11:33:52.916733 ItsCam_TestCases.ttcn:55->ItsCam_TpFunctions.ttcn:91 "*** TC_CAM_MSD_FMT_BV_01: PASS: Expected CAM message received ***"
```

FIGURE 12 – Exemple de rapport au format texte

Résultat attendu	Résultat observé	Verdict	Exigence couverte	DATEX envoyé	IVI reçu	Commentaire
Le message est bien reçu par la R-ITS-S	Message reçu par la R-ITS-S	Pass	2.6.1.2_M C2 v0.20 Scénario 1 / Étape 1	<?xml version="1.0" encoding="UTF-8" standalone="no" ?><soap:E		
L'UBR traduit le C-ITS DATEX en un IVI	IVI émis par la R-ITS-S	Pass	2.4.1.4_M v4.60 2414_H_SITUATION_002		02060000C1F6BACA9C420601265E42D233AE	
1	2	Fail	2.4.1.2_M I2V v4.70 2.3 IVIM / TrafficClass			
new (0)	0	Pass	2.4.1.2_M I2V v4.70 2.3 IVIM / ivIStatus			
2 zones	2 zones	Pass	2.4.1.4_M v4.60 3 Location of events			
1 part	1 part	Pass	2.4.1.4_M ANNEXE 7 v4.30 Translation cellule D33			
applicableLanes absent	applicableLanes absent	Pass	2.4.1.4_M ANNEXE 7 v4.30 Translation cellule D40			
vehicleCharacteristics absent	vehicleCharacteristics absent	Pass	2.4.1.4_M ANNEXE 7 v4.30 Translation cellule D44			
1 part	Il ne contient qu'un seul GicPart, sans applicableLanes ni vehicleCharacteristics.	Pass				
Un seul pictogramme	1	Pass	2.4.1.4_M ANNEXE 7 v4.30 Translation cellule B48			
Pictogramme de type ISO14823	ISO14823	Pass	2.4.1.4_M ANNEXE 7 v4.30 Translation cellule B48			
ServiceCategoryCode de type trafficSignPictogram	TrafficSignPictogram	Pass	2.4.1.4_M ANNEXE 7 v4.30 Translation cellule B48			
regulatory (1)	1	Pass	2.4.1.4_M ANNEXE 7 v4.30 Translation cellule B48			
nature (5)	5	Pass	2.4.1.4_M ANNEXE 7 v4.30 Translation cellule B48			
serialNumber (57)	57	Pass	2.4.1.4_M ANNEXE 7 v4.30 Translation cellule B48			
attribut présent	attribut présent	Pass	2.4.1.4_M ANNEXE 7 v4.30 Translation cellule B48			
1 attribut	1 attribut	Pass	2.4.1.4_M ANNEXE 7 v4.30 Translation cellule B48			
Attributs de type SPE	Attribut de type SPE présent	Pass	2.4.1.4_M ANNEXE 7 v4.30 Translation cellule B48			
Valeur de SPE {0, 90}	{0, 90}	Pass	2.4.1.4_M ANNEXE 7 v4.30 Translation cellule D48			
Le pictogramme est de type 14823 avec pour paramètres: ivi type = 1 - iso14823.pictogramCode.serviceCategoryCode.trafficSignPictogram = regulatory - iso14823.pictogramCode.serviceCategoryCode.pictogramCategoryCode.nature = 5 - iso14823.pictogramCode.serviceCategoryCode.pictogramCategoryCode.serialNumber = 57 - iso14823.iso14823Attributes.SPE.spm = 90 - iso14823.iso14823Attributes.SPE.unit = 0		Pass				

FIGURE 13 – Exemple de rapport au format XLS

5 Outil de test

► **1** Faites une nouvelle version de vos plans de tests en utilisant les primitives UT tels que définis en Annexe A.

► **2** Développez l'outil de test qui déroulera les différents tests. Nous devrons pouvoir sélectionner le ou les tests que nous souhaitons exécuter, ainsi que les différents PICS et PIXIT. Pour ce faire vous pouvez utiliser le langage que vous voulez, avec un programme en ligne de commande ou graphique selon votre souhait. Votre logiciel devra fournir un rapport des tests effectués.

5.1 Remise du projet

Le projet sera à rendre par mél au plus tard le lundi 13 avril à 9 h (un petit délai sera accordé dans la journée si des problèmes d'envoi interviennent). Il devra comprendre :

- les sources du programme,
- tout fichiers permettant de compiler (Makefile, README, fichiers de projet pour les IDE etc.),
- les deux versions de vos plans de tests.

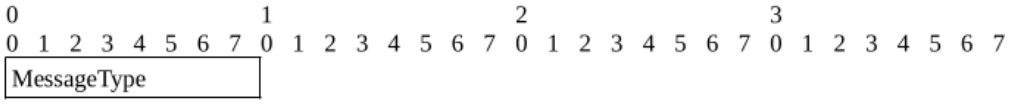
Le projet **ne devra pas** contenir le programme pré-compilé.

A Primitives UT de la machine à café

A.1 Initialisations

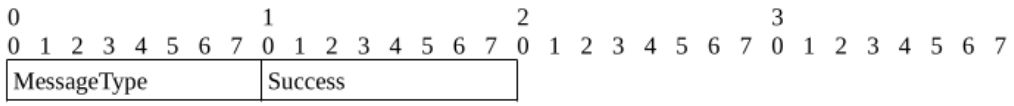
A.1.1 Initialisation

Cette primitive est utilisée pour réinitialiser la machine à café.
Requête (UtInitialize TS → UT)



Nom	Taille	Valeur
MessageType	1 octet	0x00

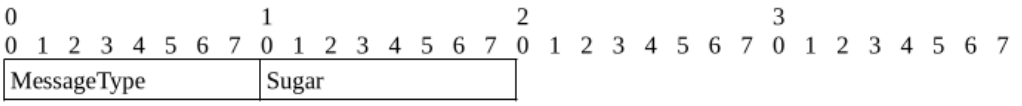
Réponse (UtResult UT → TS)



Nom	Taille	Valeur
MessageType	1 octet	0x01
Success	1 octet	0x00 : Erreur 0x01 : Succès

A.1.2 Nombre de sucres

Cette primitive est utilisée pour changer le nombre d'unités de sucre dans la machine à café.
Requête (UtSetNbSugar TS → UT)

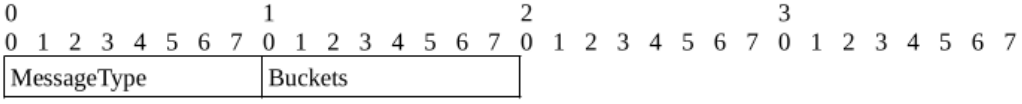


Nom	Taille	Valeur
MessageType	1 octet	0x02
Sugar	1 octet	Le nombre d'unités de sucre

Réponse (UtResult Annexe A.1.1)

A.1.3 Nombre de gobelets

Cette primitive est utilisée pour changer le nombre de gobelets dans la machine à café.
Requête (UtSetNbBuckets TS → UT)



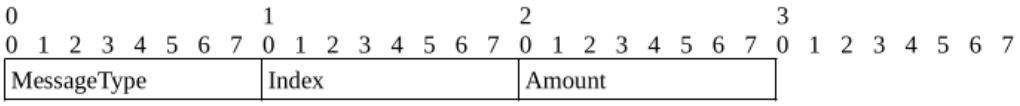
Nom	Taille	Valeur
MessageType	1 octet	0x03
Buckets	1 octet	Le nombre de gobelets

Réponse (UtResult Annexe A.1.1)

A.1.4 Nombre de boissons

Cette primitive est utilisée pour changer le nombre de boissons dans la machine à café.

Requête (UtSetNbDrinks TS → UT)



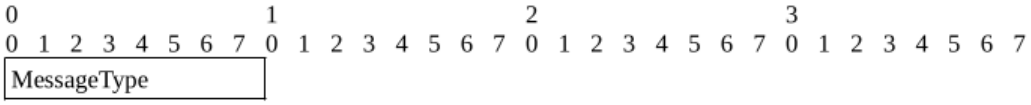
Nom	Taille	Valeur
MessageType	1 octet	0x04
Index	1 octet	Le numéro de la boisson. Attention, l'indice commence à 0.
Amount	1 octet	Le nombre de boisson.

Réponse (UtResult Annexe A.1.1)

A.2 Informations

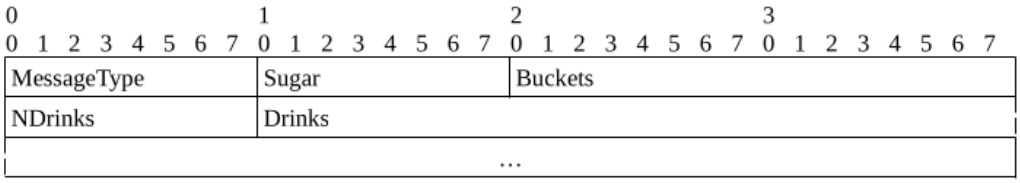
A.2.1 Informations générales

Cette requête est utilisée pour récupérer les informations globales de la machine à café.
Requête (UtGetInfos TS → UT)

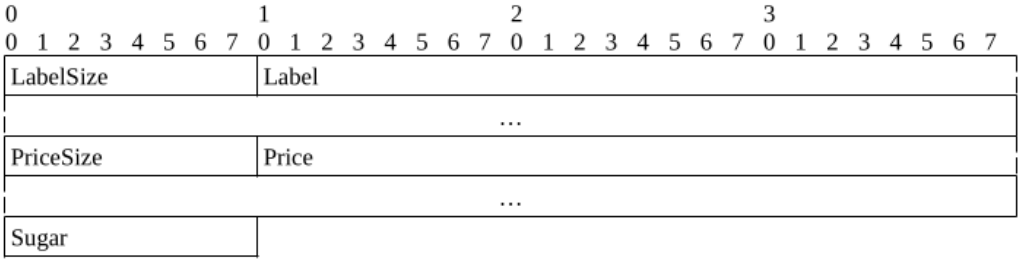


Nom	Taille	Valeur
MessageType	1 octet	0x10

Réponse (UtGetInfosResult UT → TS)



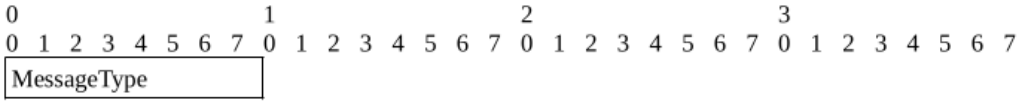
Nom	**Taille**	**Valeur**
MessageType	1 octet	0x13
Sugar	1 octet	Le nombre d'unités de sucre restants
Buckets	2 octets	Le nombre de gobelets restants
NDrinks	1 octet	Le nombre d'éléments Drinks suivants
Drinks	NDrinks × taille d'un Drink	Description de la boisson (cf. ci-dessous)



Nom	**Taille**	**Valeur**
LabelSize	1 octet	La taille du nom de la boisson
Label	LabelSize octets	Le nom de la boisson
PriceSize	1 octet	La taille du prix de la boisson
Price	PriceSize octets	Le prix de la boisson sous forme de chaîne de caractères
Sugar	1 octet	0x00 : La boisson ne peut pas être sucrée 0x01 : La boisson peut être sucrée

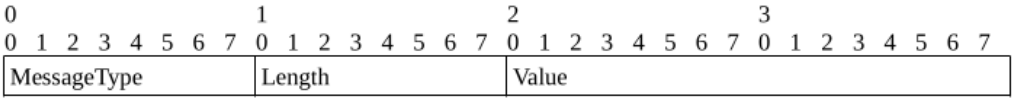
A.2.2 Affichage

Cette primitive est utilisée pour récupérer ce qui est affiché à l'écran de la machine à café.
Requête (UtGetPrint TS → UT)



Nom	Taille	Valeur
MessageType	1 octet	0x12

Réponse (UtGetPrintResult UT → TS)

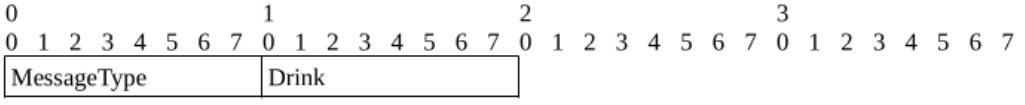


Nom	Taille	Valeur
MessageType	1 octet	0x11
Length	1 octet	La taille de ce qui est affiché
Value	Length octets	Le texte qui est affiché

A.3 Sélections

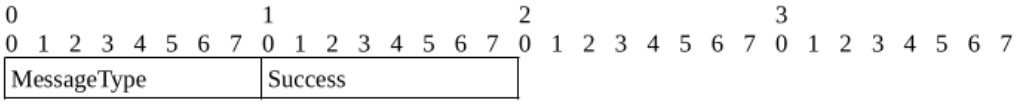
A.3.1 Sélection de boisson

Cette primitive est utilisée pour sélectionner une boisson.
Requête (UtSelectDrink TS → UT)



Nom	Taille	Valeur
MessageType	1 octet	0x21
Drink	1 octet	L'indice de la boisson désirée

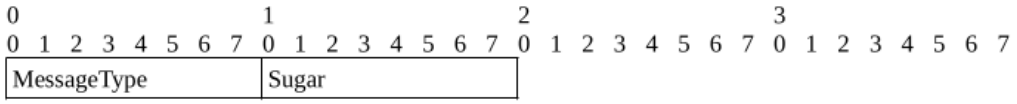
Réponse (UtSelectResult UT → TS)



Nom	Taille	Valeur
MessageType	1 octet	0x20
Success	1 octet	0x00 : Erreur 0x01 : Succès

A.3.2 Sélection du nombre d'unités de sucre

Cette primitive est utilisée pour sélectionner le nombre d'unité de sucre à mettre dans la boisson.
Requête (UtSetSugar TS → UT)

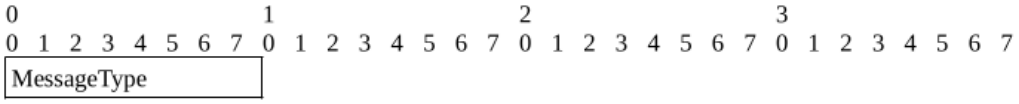


Nom	Taille	Valeur
MessageType	1 octet	0x22
Sugar	1 octet	Quantité de sucre à mettre dans la boisson

Réponse (UtSelectResult Annexe A.3.1)

A.3.3 Valider la boisson

Cette primitive est utilisée pour valider la boisson sélectionnée.
Requête (UtValidate TS → UT)

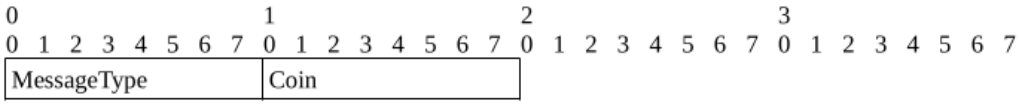


Nom	Taille	Valeur
MessageType	1 octet	0x23

Réponse (UtSelectResult Annexe A.3.1)

A.3.4 Insérer une pièce

Cette primitive est utilisée pour insérer une pièce dans la machine à café.
Requête (UtInsertCoin TS → UT)

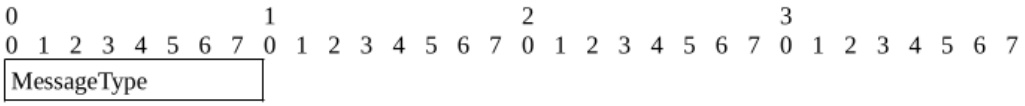


Nom	Taille	Valeur
MessageType	1 octet	0x24
Coin	1 octet	0x00 : 0,10 0x01 : 0,20 0x02 : 0,50 0x03 : 1 0x04 : 2

Réponse (UtSelectResult Annexe A.3.1)

A.3.5 Récupérer la boisson

Cette primitive est utilisée pour récupérer la boisson.
Requête (UtGetDrink TS → UT)

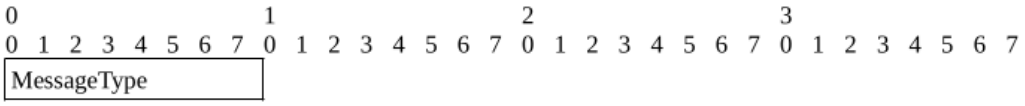


Nom	Taille	Valeur
MessageType	1 octet	0x25

Réponse (UtSelectResult Annexe A.3.1)

A.3.6 Récupérer la monnaie

Cette primitive est utilisée pour récupérer la monnaie.
Requête (UtGetChange TS → UT)



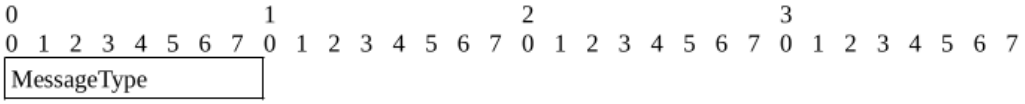
Nom	Taille	Valeur
MessageType	1 octet	0x26

Réponse (UtSelectResult Annexe A.3.1)

A.4 Indications

A.4.1 Plus de sucre

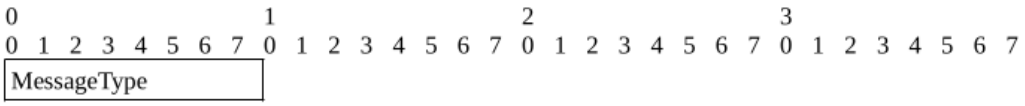
Cette requête est utilisée pour indiquer qu'il n'y a plus de sucre.
Indication (UtNoMoreSugar UT → TS)



Nom	Taille	Valeur
MessageType	1 octet	0x30

A.4.2 Plus de gobelet

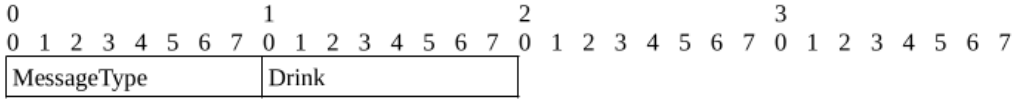
Cette requête est utilisée pour indiquer qu'il n'y a plus de gobelet.
Indication (UtNoMoreBucket UT → TS)



Nom	Taille	Valeur
MessageType	1 octet	0x31

A.4.3 Plus de boisson

Cette requête est utilisée pour indiquer qu'une boisson n'est plus disponible.
Indication (UtNoMoreDrink UT → TS)



Nom	Taille	Valeur
MessageType	1 octet	0x32
Drink	1 octet	L'indice de la boisson indisponible

B Configuration de la machine à café

La configuration de la machine à café se fait dans le fichier `config.json` qui doit être placé dans le même répertoire que l'exécutable.

- `consumable` contient les paramètres des consommables
  - `sugar` → le nombre d'unités de sucre,
  - `bucket` → le nombre de gobelets.
- `drinks` contient la liste des boissons :
  - `label` → le texte affiché sur le bouton,
  - `price` → le prix de la boisson,
  - `img` → le chemin relatif vers l'image de la boisson,
  - `nbDrinks` → le nombre de boisson.
- `ut` contient les paramètres de l'UpperTester
  - `utPort` → le port d'écoute,
  - `interface` → le nom de l'interface utilisé,
  - `reception` → pourcentage de succès de réception des primitives UpperTester,
  - `faulty` → pourcentage de risque de retourner une erreur à une requête UpperTester,
  - `indication` → pourcentage d'indications envoyés par l'UpperTester.
- `strings` contient les messages statiques
  - `chooseDrinkText` → indication de choisir une boisson,
  - `drinkMadeText` → indication de récupérer la boisson préparée,
  - `changeText` → indication de récupérer la monnaie,
  - `noBucket` → indication qu'il n'y a plus de gobelet.