Solve the Game™

Voice Game Developer Guide

TABLE OF CONTENTS

SECTION 1: Overview

Solve The Game™ is one of the most unique Alexa™ Skills, because it not only lets you play highly advanced voice adventure games with your Alexa™ device, but it also allow you to design, create, and build your own games using a simple web-based interface.

The core of Solve The Game™ is advanced game engine software, which intelligently interacts with the game player entirely using voice on the Amazon™ Alexa™ platform. This game engine uses a heuristic (probability based) approach to determining the game players' voice phrases, with a real-time dictionary and streamlined optimization. It then applies the game developer's logic to deliver the appropriate response to the game player, as well as keep track of virtual locations, objects, characters, milestones, and triggers. This delivers a most advanced and robust gaming experience to the player.

Each game developer can decide entirely how they want to build their own games, from a basic adventure game, to an escape-the-room-style game, or maybe even a virtual landscape, a murder-mystery, or even an intelligent robot-type of interaction. All this is under the developers' control through the Voice Game Developer Console at www.SolveTheGame.com.

As a game developer, you can leverage full-featured game-playing voice power with just a few clicks! Build your first game in 10 minutes with no programming languages to learn, software to buy, or servers to maintain! This is enormously powerful!

Once your games are created and completed in the Voice Game Developer Console, anyone in the world can play the games for free using the Solve The Game™ Alexa™ Skill!

In this guide, we will cover some basic definitions and concepts first. Then we'll walk you through getting started making your first game, including setting up your account, logging in, and creating a new game. In addition we'll show you how to add your own locations, objects, and characters, which are the essential elements of game play. Then we will conclude with milestones, triggers, more advanced concepts, and best practices, which will allow you to truly deliver an incredible gaming experience for your players.

<u>SECTION 2: Definitions</u>

Core Game Development Sections:

Locations: Locations are places within the game where the player may navigate. They can be connected to other locations with a North/South/East/West grid or in more advanced ways. The player will always be at some location at all times.

Objects: Objects are things within the game that the player or other characters in the game can have in their possession. Objects may also be at different locations. Each object may only be at one location at a time.

Characters: Characters are people, animals, or other entities with whom the game player can speak and interact. The player may talk to characters and will receive responses. Characters may possess objects and be at different locations.

Additional Game Development Sections:

Defaults: Defaults are initial values you specify for the way a game will start.

Global: Global phrases are things the game player can say/ask at any time during the game play, with associated action responses.

Triggers: Triggers are automatic action responses, which can occur at any time during the game play, based on a set of rules specified for each trigger.

Milestones: Hint Milestones are used to keep the player on-track and moving toward the desired solution of the game. Each milestone will have responses associated with it, which the player will hear when they ask for a hint. The game engine automatically monitors the player's progress along these milestones.

Game Development Terms:

Phrases: Player Phrases are things the game player may say, which will result in action responses by the game.

AKA (Also Known As): AKAs are alternative terms you may want to use for locations, objects, characters, and player phrases to ensure the game player won't get frustrated if they happen to use a synonym.

Action Response Terms:

Action Responses: Action Responses are a set of specific actions which the game engine will perform in response to a matching game player's phrase.

Says: Says are the responses that you want your game to say to the game player.

Moves: Moves cause objects and characters to be moved to another location, object, or character.

Go: Go specifies that the player will be moved to a new location.

Requirements: Requirements specify that certain response actions will occur only in the case that those requirements are met. They are logical conditions, in the form of: {entity} {is / is not} at {entity}.

Cases: Cases are used when you would like to specify different action responses to the same phrase, based on specific game-play conditions at that moment.

<u>SECTION 3: Getting Started</u>

Creating Your Developer Account

To get started with your developer account, you'll first need pass-phrase from the Solve The Game™ Alexa™ Skill. To do this, just enable and open the Solve The Game™ skill on Alexa™, then say "Design my Game". You will then hear a response directing you to browse to www.SolveTheGame.com on your computer. It will also include your unique initial developer pass-phrase.

Once browsing to www.SolveTheGame.com, click "Register your developer account". Enter the initial developer pass-phrase you were given by the Solve The Game™ Alexa™ Skill, choose your own username and password, and click "Submit Registration" to proceed. Your developer account will be immediately created and you will be taken to the Account page of the Voice Game Developer Console.

The pass-phrase is only used on your initial account creation. In the future, just enter your username and password to login.

Note: If you forget your pass-phrase, just ask Alexa™ "What's my pass phrase?" If you can't understand the pass-phrase you were given or would like a different one for any reason, just ask Alexa™ for a "New pass phrase."

If you forget your password or ever want to change your username/password, just get a new pass-phrase from the Solve The Game™ skill on Alexa™ and re-complete the registration. You're existing games will be retained.

Account Details

IMPORTANT NOTE: Once you're logged into the Voice Game Developer Console, you can hover over most sections to learn more about how they work and how to best use them. There is a lot of great information available this way.

The account page of the Voice Game Developer Console allows you to specify your optional Developer Handle and Email address.

If you provide a developer handle, game players will be told that your games were created by the name you provide as this handle.

If you provide an email address, you will be sent emails by the game engine when players leave comments about your game. You may find these motivational, and may also find them very helpful in determining areas of your game which you would like to revise, improve, or correct.

In addition, providing an email address will allow you to receive feedback when your game is reviewed and/or approved to go live.

The Voice Data Soft™ Voice Game Developer Console is entirely free of charge and may be used for you to create an unlimited number of voice games!

Adding a New Game

Now, try clicking "[Add New Game]" to create your first game. You will be prompted to enter a name for your game, which you can change any time. You should also choose a category for your game.

By default, all new games are in the "DEVEL" (developer) status, which means *only you* can create and play your game, while you're developing it. In fact, we designed the system so that, unless you make substantial changes to your game, you can actually open the game on your Alexa™ and talk to it WHILE you're building it in the Voice Game Developer Console! To test your game anytime, just say “Play” then the name of your game, while in the Solve The Game™ skill on your Alexa™. Remember, ONLY YOU can test your game while it’s in the “DEVEL” status.

Once you have your game to a stage where you would like your friends to try it out, you can change the status to "BETA". This will then allow your friends with an Alexa™ to enable/open Solve The Game™ on their Alexa™ and say "Play" followed by your game's name. In Beta mode, your game will not be mentioned or playable by the public. There is no charge to beta testers to play with your game.

Then, using the Design Standards in Appendix B, ensure your game is really ready for the world! When you feel you have completed your game to these standards, change the status to “REVIEW”. Your game will then undergo our review process, which can take up to 10 days. You will receive an Email either approving your game or telling you what needs to be changed in order to bring it live. *During the review process, you will not be able to edit your game in any way, so be certain you are really ready first!*

Once your game is approved, its status will change to “LIVE”. Then, anyone in the world can play your game using their Alexa™ with the Solve The Game™ skill! Your game will be in our Newest Games list, and may also appear in the Most Popular list, too, depending on how much players like it. There is no charge for anyone to play your games on Solve The Game™! We do charge for hints, though.

Once your game is in the “LIVE” status, you will not be able to make any changes to the version of your game that the world is playing. If you want to change/improve your game, change the status back to “DEVEL”, which will allow you to notice and test changes immediately. However, the public will not notice any changes you make, until the game is reviewed and approved again. In addition, a game with a live version may not have beta testers.

Also on this page, you'll see the dates you created, last modified, submitted for review, and got live approval for this game, as well as the number of unique players, sessions played, and total interactions by game players! These are fun to watch as you release your new games to the world!

Finally, you'll also see an average customer rating for your game (between 0 and 5 stars) as well as the most recent comments your game players wanted to share with you about your game! Also, the most misunderstood player phrases from the last day will appear which are helpful in adding actions for phrases you may have overlooked.

Click the Save button when you're ready to proceed.

Note: There is also a "DELETE" button at the top - CAREFUL!! THIS WILL ENTIRELY DELETE YOUR WHOLE GAME AND CANNOT BE UNDONE!!

Adding Your First Location

All games must have at least one location, so go ahead.. Click "Locations", then "[Add New Location]", then enter a name for your name game location and click "SAVE".

Note: Once again, BE CAREFUL TO ONLY CLICK THE "DELETE" BUTTON IF YOU WANT TO PERMANENTLY REMOVE A LOCATION!

You'll see an optional "AKA" (Also Known As) field for this location, as well as all objects, characters, and even player phrases themselves! You can use this in cases where you would want ease the burden on the player to remember the exact names you used. For example, you might call a room "Home Office" but players may want to simply refer to it as "Office". Just enter "Office" as one of the AKAs and the game engine will take care of the rest!

NOTE: If you use AKAs be certain to ONLY reference them within your design as their original name.

You'll now notice some required and optional phrases appear. Start by clicking "Look". You'll see several action fields appear on the right. For now, just use the "Say" actions (we'll discuss the other action fields in the next chapter). Enter at least three (but more is even better) phrases for "Say". These are the responses the game will provide the game player when they enter that new location.

The response the game player hears will be randomly chosen from your "Say" actions, but the game won't repeat the same one twice if they ask multiple times, unless you only have a single "Say" action.

You might enter things like "You're standing in the center of an office" or "You're at a strangely-familiar home office".

Now click "SAVE" and then try opening Solve The Game™ in Alexa™. Say "Play" then your game's name. You'll notice the game player is now at your initial location (you can change the starting location anytime in the Defaults section).

FUN, HUH!?

Now, click the "Inspect" phrase for your new location. Here, again, enter as many "Say" actions as you like. These will be said to the game player when they look again, look closer, inspect the location, etc.

You'll notice North, East, South, and West phrases. These are used when the player tries to move from one location to another. Don't worry; The player doesn't have to remember the directions -- They can just say "Go to the kitchen" or "Go forward" to get to another location. For now, though, you don't have any other locations yet, so skip these.

Later, when you add more locations, you'll want to specify the new location in the "Go" action. This tells the game engine that the player should go to that new location. The "Say" actions become optional when you use the "Go" action, as sometimes you'll want to transition to a new location yourself and other times you'll just let the game take them there.

You can also add new phrases here at this location. For example, you might want to say "Touch the Ceiling" and when they do it says "The ceiling seems quite normal" or “it is much smoother than most ceilings.” This phrase will ONLY be heard/matched when the game player is at this location.

Note: If you want game players to navigate around on their own, you'll want to draw out a grid of your locations and figure out the North, South, East, and West locations for each of your "Go" actions. This will let the player "walk" around your game automatically!

Adding Your First Object

Objects and characters are optional, but really useful and essential for most games. Try clicking "Objects", then "[Add New Object]". Give it a name. You'll notice it can accept AKAs, just like locations. Then specify if it is singular or plural, which is important so the game engine knows how to refer to it.

You must also specify the object's initial, starting location. You can choose any of your existing locations or "*nowhere", which will cause it to exist but not actually be anywhere the game player can get to (effectively hidden), or "*have" which means the game player will begin the game with this item in their inventory.

You'll next notice an optional "Facing Direction" field. If you don't specify one, the object will just be there at the location. If you specify the "Facing Direction", the object will be on the specified side of that location. For example, you may have a calendar in the office, but not want it just sitting on the floor, but instead be on the South wall of the office.

The "Where Phrase" can also be used to help specify more details about the objects placement in the location. For example, you might add "hanging on the wall" for the calendar. The game engine will then say that "hanging on the wall, there is a calendar here."

Of course, you might combine the "Facing Direction" and the "Where Phrase". In our calendar example it would then say "Directly in front of you, hanging on the wall, there is a calendar here."

Now, click the required "Inspect" phrase for your new object. Here, again, enter some "Say" actions, which will be spoken to the game player when they inspect this object more closely.

If the object is something that really couldn't/shouldn't be removed from the location, you can add "Say" actions to the "Can't Get" phrase. Once the object has any actions in the "Can't Get" phrase, it will automatically not be allowed to be taken from this location. For example, you might have a couch and the say action would be "The couch is just too big for you to carry around in this game," which will be spoken to the player if they try to take the couch.

Now, go ahead and add a new player phrase to the object. This phrase will only be heard/matched when the player is either at the location where this object is or if they have the object in their inventory. Add a few "Say" actions to your new player phrase.

If your phrase would have included the object's name, just leave it out. For example, if you want to let people sit on your couch, just add "Sit" as the player phrase. The game engine will know that sit only applies to this object, because you're adding the player phrase to this object.

NOTE: If you use another object name in your phrase, be certain to ONLY reference the other object with its actual name, never with an AKA. For example, if you have an object named "Mechanical Pencil" with an AKA of "pencil" and you want people to be able to put it on the couch, your phrase would be "put mechanical pencil on couch" (not "put pencil on couch"). The game engine will automatically allow AKAs for the phrase.

If you're unsure what type of verbs you might want to use, see Appendix A which shows the best-use (system) verbs. For example, you should use "get" instead of "take".

Adding Your First Character

Now, click "Characters", then "[Add New Character]" and enter a name for your new character. Just like locations and objects, characters, too, may have AKAs. Then specify the gender of the character, so the game engine can properly use pronouns. Then choose a voice for this character. Finally, specify the initial, starting location for this character when the game begins. Once again, you can use "*nowhere" to make this character exist in the game but be hidden upon startup.

The required "Introduction" phrase allows you to specify "Say" actions for what this character will say when you first greet them, perhaps with "Hi!"

The optional "How Are You" is recommended for responses to player phrases of this nature. The optional "Describe" phrase is also recommended for the game engine to describe this character to the player, if they ask for more detail on the character.

Finally, you can add more phrases, just like in locations and objects. Of course, only the phrases for this character will be heard and match when this character is in the same location as the game player.

You'll want to add many more "Say" actions for each phrase, and many more phrases for characters than locations or objects, because your game players will often want to talk a lot to the characters.

Note: If something was said that the character didn't know, it will give a generic response in the character's voice.

Global Phrases

Click in the Global section and you'll see many required phrases there. These are pre-populated with "Say" actions, which you may want to change or leave as-is. In addition, you can add any number of global player phrases, which will be responded-to regardless of the player's location, inventory, surrounding objects, or nearby characters.

SECTION 4: Creating Game Play

All game play steps are transactional, meaning the game player says something and the game responds, then nothing happens until the player speaks again. The game engine will best match the phrase heard by the player with those applicable at the current stage and conditions of play, and take the resulting action response(s) you specify.

Only the phrases you have included for the current location, objects which are at this location, objects the player has in their inventory, and characters at this location, as well as global phrases, will be considered for matching.

The action responses can be things like spoken sayings to the player, moving the player to a new location, and moving object(s) and character(s). You can specify as many response actions as you like for each matching phrase.

You may want a phrase to only match under certain circumstances, which are called Requirement Conditions. You may also want to have several different Requirement Conditions for a single phrase, where only one matches based on the current conditions, which are called Cases.

This simple combination of Player Phrases, Action Responses, Requirement Conditions and Cases allow you to build and incredibly complex, powerful, and robust voice adventure game with simple clicks on the SolveTheGame.com website.

The power of your game is how you connect everything together!

Requirement Conditions

You may choose to include optional requirement conditions for each phrase's actions responses. You can have as many requirements as you like, but they ALL must be met in order for the phrase to function. If requirement conditions are included, but none of them match, it is as though the phrase doesn't exist, and the game-engine will respond accordingly.

For each requirement condition, you must select two entities to compare with each other. The comparison can be "is/are" or "is/are not".

The left-hand entity can be any object or character, and may also be *player. You will be comparing to see if this entity (object, character or the game player himself) is or is not at a specific place.

The right-hand entity can be any location, object, or character, and may also be *nowhere, *have, or *here. The game engine will compare whether the left-hand entity is or is not at the right-hand entity place.

Some examples you might specify are:

- "Secret Bottle Is at The Dock",

- "*player Is Not at The Kitchen",
- "Stapler Is at *nowhere", or
- "Key Is at *have"

As you can see, the requirement conditions are quite intuitive. To add additional requirements for this action group, just click the "+" icon to the right of the last requirement and a new blank one will appear.

Only if all the requirements specified are satisfied, will the action response(s) occur.

*Player is used to indicate that you are going to make a requirement comparison of the player's current location.

*Nowhere is a special location which will not appear anywhere in the game play, but still allows the object or character to exist, essentially hidden.

*Have is used in requirement conditions as well as move actions, where you want to see if the game player has an object in their inventory, or move an item to their inventory.

*Here is used for requirement conditions as well as move actions, where you want to see if the game player, an object, or a character is at a location, or to move an object or character to another location.

Note: You may have noticed that objects can have objects in them. When this occurs, the inner objects are effectively hidden. The outer objects will be announced, but the game engine will not mention the inner objects. You'll need to do that with your own phrases. It can be helpful for you to determine if the player found the object yet, if it is still at its initial starting object. You might want to use this for milestones (explained later).

Moving the Player, Objects, and Characters

You can cause the game player to move to a new location by selecting the location in the "Go" action. This will immediately move the player there, even if they aren't near that location. These are used for the standard North/East/South/West system phrases. The “go” action can also be useful to move the player to a new location when they accomplish something specific, or say the right thing to a character, get the right object, etc.

You can also cause objects and characters to move locations, in the "Move" actions. Choose the object or character you want to be moved, then choose to where you would like it moved.

You can specify "*nowhere" to make the object or character become effectively hidden from the game play. You can choose "*have" for objects to add them to the player's inventory. You can choose "*here" to cause the object or character to be moved to wherever the player happens to be at that moment. Of course, you can choose an actual location, too.

Optionally, you can specify a replacement object. This is useful when you want one object to be replaced by another. For example, you could have a "Glass Bottle" move to "*nowhere" and be replaced by "Broken Pieces of Glass" when the player says "Smash" in the Bottle object.

You can enter as many "Move" actions as you like, for moving multiple objects/characters in a single player phrase/requirement condition, by simply clicking the "+" icon after each move, to add another.

Advanced Actions

If you click the "advanced" button in the actions section, you'll see some additional fields.

There are times where you want to change a player's "Facing Direction" as part of the action. For example, if the player is facing the couch which is on the East wall, you might want to specify a facing direction of "West" when they sit down, so that they can see what's in the room still, rather than still be facing the couch they are sitting it.

You may, at times, want the "Say" actions to be spoken by one of your characters. For example, in the "Mechanical Pencil" object, you may have a "Give Mechanical Pencil to Mary" player phrase, which moves the "Mechanical Pencil" to "Mary" and also says "Thank you." Just select "Mary" in the "Speaker" drop-down to have Mary's voice used for the response.

While playing the game, if there is a single character at a location or if the game player just spoke to a character, that character automatically becomes the Focused Character. Subsequent phrases made by the game player will automatically be directed to that character, until the player does something non-character-related (i.e. changes locations, looks at an object, looks around, talks to another character, etc.). If you want to override this, specify a Character Focus to make another character have the focus for the next interaction.

In addition, a specific object will also retain focus, so the game player can continue to interact with it without having to mention its name each time. To override this, specify an Object Focus. This can be useful when you have a hidden object inside another object and then want the player to be able to say "take it" after just being told that it was there hidden. Specify the hidden one as the Object Focus in the action which found it, then the "it" in the player's phrase will reference the hidden one.

Cases

You may want to have different groups of requirement conditions for a single phrase. For example, if the player says "Give the Mechanical Pencil to Mary" and they are in a location where Brent is also, maybe Mary might whisper "That's nice, but maybe when we're somewhere else". This is easily done with 2 cases: The first with a requirement that "Brent IS *here" with the whisper "Says" action, and a second case with no requirements (as long as Brent isn't here, the 2nd case will match automatically) which actually gives Mary the pencil and she says Thanks.

Note: The game engine will automatically not allow you to give an object the player doesn't have to a character who's not here, etc.

Every player phrase has an initial case. To add additional cases, just click the "+" icon at the bottom-right of an action group. Each case has its own requirement conditions and associated actions. The cases are compared in order, starting with the first one. The first matching case will be used by the game engine.

You can use the Up and Down arrows to change the order of your cases. Generally, put the most complicated ones (those with the most requirement conditions) at the top.

You may want to have a default case, which will match if none of the others do. To do this, simply add a final case at the end of the others, with no requirement conditions included. It will then always be matched if none of the others do, since they are checked in order.

If the phrase the game player spoke doesn't adequately match any of the game phrases and you don't have a default (no requirement conditions) case at the end, the game engine will automatically give a friendly response to the game player. Sometimes this response will be a simple "I don't understand" and other times it will prompt the game player to learn more about using the game or even asking for a hint.

SECTION 5: Enhancing the Player Experience

You can use Milestones, Triggers, and Default values to create hints for the game player, define actions which occur automatically, not requiring a specific player phrase, and to specify some initial values for game play.

Milestones

Milestones are steps along the way toward the game player's solving of the game, for the purpose of providing the game player with appropriate hints when they ask. Generally, they will be in order, but it is fine if the player completes a step out of order, which happens fairly often. Each milestone has "Say" actions which are spoken to the game player, as hints to help them get to the next milestone. The game engine will handle all this for you.

You start with the first milestone named "Start" which will have NO requirements. This milestone is simply the player starting the game, and should have clues you want the game player to hear first.

You should add "Say" actions for each milestone, which are actually the hints the player hears to help them get to the next milestone. Unlike regular "Say" actions, these are always said in order, so you will want to start with a more general response and move to more specific. For example, your first entry might be "I think there's something in this room to help you", followed by "Maybe it's hanging on the wall?", then "look closer at the calendar on the wall".

Each time the player hits a milestone, regardless of whether they did it in order or not, they get a nice little fun sound. This is rewarding to let them know they're on the right track. Then next time they ask for a clue, it will offer them hints toward whichever is the next milestone you have defined, in the order you defined them.

You can use the Up and Down arrows to change the order of your milestones. Keep them in the general order you expect the game to be played.

While you're developing a game, you can say "Go To Milestone" then the name of one of your milestones. This will restart the game over and take you to this milestone. For long/complex games, this will help fast-forward you to a specific portion of your game. Note: it MAY have incorrect objects in incorrect places when fast-forwarding to a milestone. To fully ensure your game is functional, say "restart" to start over at the beginning and play through the entire game at least a couple times before bringing it to "live" status.

Triggers

Sometimes, you want to have a certain combination occur, including, perhaps, an object, a location, and a character, but you don't know which will happen first. Rather than recreate numerous cases in the object, and again in the location, and again in the character, you can just create a single trigger.

The trigger works just like other action groups, with requirements and cases, but it isn't initiated by a player phrase. Instead, you just give it a name that you will use to reference it. Then, as the game player is playing the game, as soon as the requirements are met, the trigger will fire immediately causing all the actions you specify to occur.

Each trigger will only fire one time in the game play.

If you're developing your game and want to reset your trigger statuses, so that they will all be checked again, you can say "reset triggers". This is helpful while you're building and testing triggers.

Defaults

In the Defaults section, you can specify at which location and in which direction the player will be when the game first starts. These are automatically assigned when the first location is added, but can be changed any time.

You can also assign an optional alternative voice to be used when hints are spoken.

SECTION 6: Tips, Tricks, & Best Practices

Following are some of the best ways to ensure your game works as you expect it to, while continuing to improve the game player experience:

- When naming locations, sometimes you'll want to use "The" in front of the name. To determine this, ask yourself it the location can be referred to as "a" or "an" -- if not, add "The" to the beginning. For example a location named "hill stairs" wouldn't make sense if the game said "In front of you is a hill stairs". So instead, name it "The Hill Stairs" and the game will say "In front of you is the hill stairs".
- You can use all standard SSML-notation in your "Say" phrases. For example, you might want to whisper, speak softer, speak faster, or annunciate a specific word. If you get a generic error response that means your SSML is malformed.
- Sometimes you may want to create 2 similar objects, and have one object replace another when an action occurs. In this way, the user thinks one thing turned into another.
- The maximum number of items a player can have in their inventory at any time is 5. However, when you move an item to "*have", this limit is overridden, so you can always expect the item to successfully move to the player's inventory.
- If you're finding it difficult for the game to behave as you're expecting it to, especially with beta testers who say things you weren't expecting, you might try adding additional Player Phrase AKAs. Maybe sit in, while a beta tester plays your game and see what they say. You'll find cases where you'll want to add AKAs or alternative methods for accomplishing something. If a phrase doesn't quite match perfectly, add AKAs to your locations, objects, characters, and phrases themselves, to get it work as you expect.
- To cause the player to win the game, just enter a "Go" action to "*winner" location. You can include "Say" actions too, if you like. The game engine will announce to the player that they have won and ask them to rate your game and send you comments, too, all through the Solve The Game™ skill!
- We recommend making your game as hard as possible, and then add milestone hints. This way, advanced players will enjoy a challenge and anyone can ask for a hint whenever they like.
- Test your game frequently, even while building it.
- Whenever you want the game player to be able to give an object they have to a character, you must add that phrase. For example, go to the object itself (so that the phrase can only be heard when the player is with it) and add "Give cookies to Brent". Then you can apply the appropriate actions. This way, the phrase can't ever be heard when the game player isn't at/have the cookies.
- When one object is being used on another object, put the "use" phrase on the first object, that is, the one being used on the other one. For example, go into "key" and add the phrase "use key on lock".
- It's generally best to spell out all numbers.

- When naming objects which have two words in them (i.e. sun glasses, foot print), enter them with a space in between the words (i.e. “sun glasses”, NOT “sunglasses”). The game engine will then automatically understand both.
- For your “say” actions, try not to use pronouns like “it”, “them”, or “that”, because you never know what the player last did and what *the game player* thinks is “it”. Instead, use the actual phrase.
- Many of the common phrases in the game can be overridden for your specific needs. For example “drop the cookies” is an automatic phrase associated with the object cookies when they are the game player’s hands. However, if you want to change the action to make the cookies move to “*nowhere” with a customer message that dropping the cookies made them crumble, you can do this by making your own “drop” phrase on the cookies.
- You must use the best-use verbs list in the appendix, when overriding system phrases. For example, you cannot use “put down cookies”, but instead must use “drop cookies” because “drop” is a best-use verb.
- If you do re-define a system phrase with a new one of your own with a specific requirement, be sure to add another case with no requirements, which actually moves the object to “*here”. Since the old system phrase no longer exists, you must perform the action when no case requirements match.
- When adding simple phrases for objects, don’t repeat the object name. For example “move” should be used as a phrase under the “boxes” object, when you want to hear “move boxes” from the game player, or “look” under the “bottle” object, when you want to allow “look in the bottle”. This ensures that all the AKAs for the object will continue to work.
- When the player is off-track from the milestones, use “hint” phrases at other locations, to help get them back on-track. Location-based hints override milestone hints. If you include a player phrase of "hint" at any location, that will override the milestone at that location. This is especially useful if you have some "off-course" locations. For example, maybe something you say to a character lands you in jail. Then you could have a hint phrase at the jail, with cases in it for properly helping you get out of jail! You can add Requirement Conditions to location-based hints, and if none match, then the milestone hint will match.
- Name your objects most simply (i.e. "ladder" instead of "escape ladder"). You can always add an AKA for “escape ladder”, “emergency ladder”, etc.
- It is a good idea to add "yes" and "no" phrases to all characters, as people often say “yes” and “no” to them
- Any Say Actions you may add to the direction phrases (North, East, South, and West) will automatically announce the new location's "LOOK" say actions. You can use this as a transitional phrase, leaving one location and going to another.
- Triggers only fire ONCE during the player’s entire gameplay! If the circumstances occur additional times, the triggers don’t fire.
- When using one object on another object (i.e. "put sparkles in old bottle") use the exact item names, not any of their AKAs. This is important, so that all the AKAs will automatically be matched by the game engine.

- When you're ready to share your game with the world, click the game name then change the status to "Live" – it will then be ready for anyone in the world to play!
- Take your time building your game, and paint a picture with your words!
- To learn more, check out our Quick-Start Tutorial Video in your developer account or visit www.VoiceDataSoft.com.

APPENDIX A – Best-Use (System) Verbs

- inspect
- go
- forward
- back
- left
- right
- look
- "what's here"
- "look around"
- get
- drop
- use
- give
- inventory
- "where is"
- talk
- intro
- query

APPENDIX B – Design Standards

Follow these standards to ensure your game gets approved easily when you submit it:

- Ensure all Amazon™ and Alexa™ standards are met. These can be found at: https://developer.amazon.com/docs/custom-skills/policy-testing-for-an-alexa-skill.html
- Games should be family-friendly and include no swearing, sexual references, violence, illegal messages, or any other content which could be inappropriate for children playing them.
- Games may not include any reference to any actual person. They must be entirely fictitious.
- Games must include milestones to help the player through the desired winning game-play.
- Games must have more than one Say Action for each phrase, so the players don't get bored.
- Games should have comprehensive Global Say Actions, especially "Overview", so players understand what they are playing
- Games should function properly, ensuring the player is ABLE to win and free from misleading or deceptive messages causing them undue frustration.
- Games should be challenging but not impossible to win.
- Games do not need to use advanced features, such as triggers, requirements, and cases, but use of them does make for a much more intriguing and enjoyable game for the players,